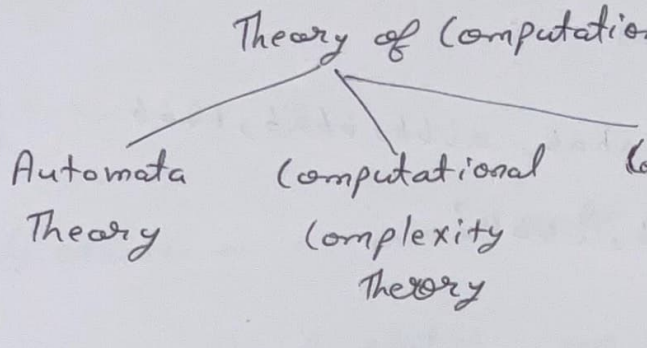
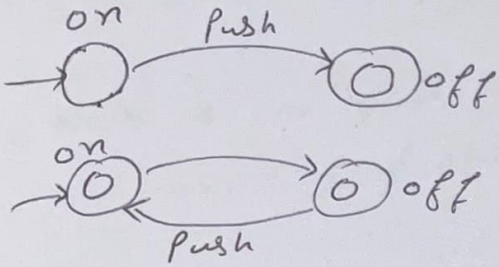
Automata Theory

Tokens :- is just like syntax





$\Sigma^0 = \{\varepsilon\}$ → Epsilon → 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 closer

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

→ It is always be a infinite set

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

$\Sigma^+ = \Sigma^* - \{\varepsilon\}$ (Positive closure)

$(\Sigma^*)^* = \Sigma^* \Rightarrow (\Sigma^*)^* \subseteq \Sigma^* \Rightarrow \Sigma^* \subseteq (\Sigma^*)^*$

$(\Sigma^+)^+ = (\Sigma^*)^+$ when $\varepsilon \notin$ belong to the set so they are True

" $\varepsilon \notin$ not belong " " " " False

$T = \{aa, ab\}$

we also do $T^* = \{\varepsilon$ → Epsilon$, aa, ab, aaaa, aaab, abaa, abab\}$

$S = \{ ab, bb \}$

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

Is $S^* = T^*$ Yes

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

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

$S^* \subseteq T^*$

$T^* \subseteq S^*$

$S^* = T^*$

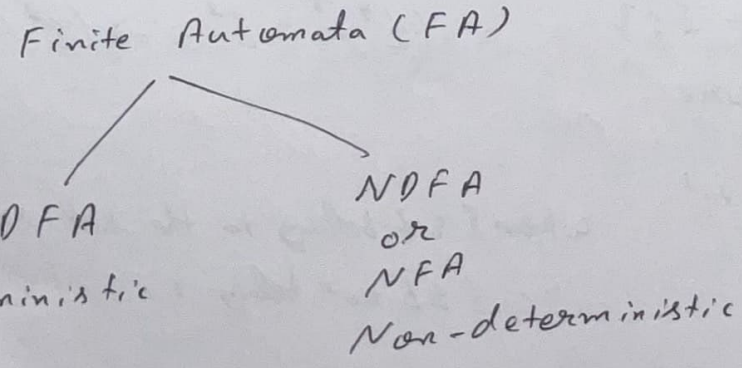
If $T = \{ ab, bbb, bb \}$

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

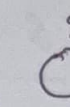
Countable infinite ?



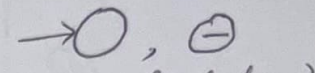
If we write FA so it is DFA

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

Q = finit set of states = $\{q_0, q_1, q_2 \dots q_n\}$



(state)


(initial state)

$s \in Q$
↓
s would be
single element
initial state
q_0 is usually choosen as an initial state

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

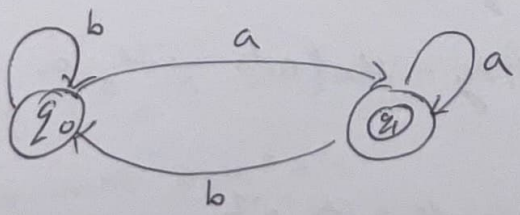
If their is no final state than we can take $q_0, q_0 q_1 \dots$
- - - as as final state

If initial state is ⊖ than final state is ⊕

δ is a transition function

$\delta(q_i, a) = q_j$
↓ ↓ → we goto q_j
initial state, after reading some element

if $q_i = q_j$ so it is loop edge
OR
$\delta : Q \times \Sigma \to Q$



$Q = \{q_0, q_1\}$
$s = q_0 \in Q$
$F = \{q_1\} \subseteq Q$

δ \ Σ	a	b
$\rightarrow q_0$	q_1	q_0
(q1)	q_1	q_0

final state

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 1-1hi entry hona sabki it is sign of Deterministic

→ if some columns are empty or if one cell contain 2 entries (q_0, q_1) like so that is non-Deterministic

$L = \{a, aa, ba, aaa, aba, baa, bba, baababa \dots\}$

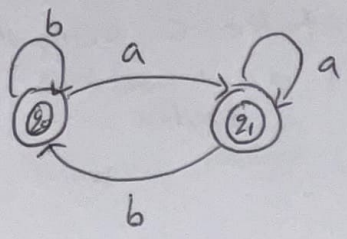
L ↑ language accepted by FA

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

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

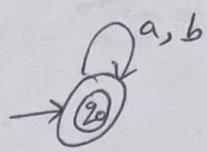


when we make every state as final state

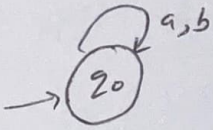
$L = \{ \varepsilon, a, b, aa, ab, ba, bb ------ \} = \Sigma^*$

so it is

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

we also achieve this by 

$L = \{ \varepsilon \}$ (empty) NULL string

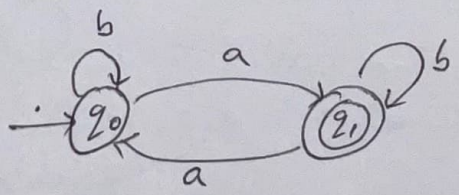


it is also finite automata because $F \subseteq \emptyset$

$\phi \subseteq \emptyset$

$L = \phi$ null language

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



This is finite automata of odd a's

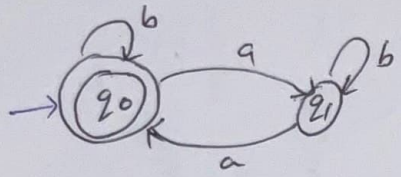
in DFA is unique transition on sigal letter

To make even' a's

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

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

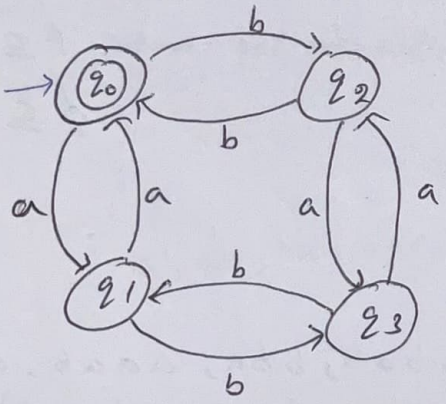
ε
Epsilon



$L = \{ \varepsilon, b, aa, bb, aab, abb, baa, bbb, \ldots \}$

It is even a's automata

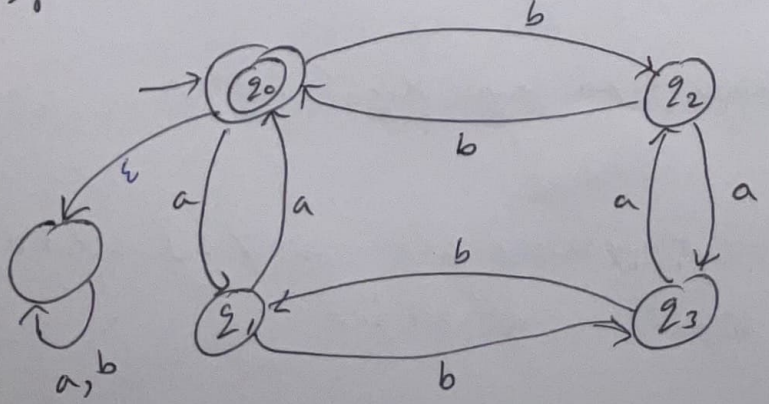
Q $L = \{ \varepsilon, aa, bb, aaaa, bbbb, abab, baba, abba, baab, bbaa, aabb, \ldots \}$



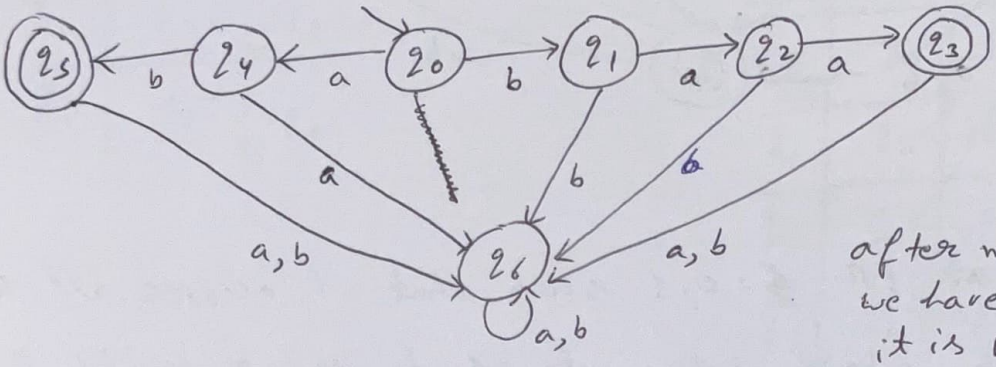
EVEN-EVEN FA

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 exepts only 2 strings which are bba, ab only (Spacilized FA)

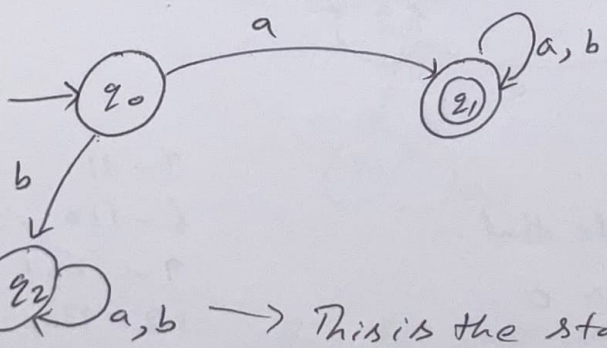


after making states (FA) we have to make sure it is DFA

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

Q All the strings which starts with A

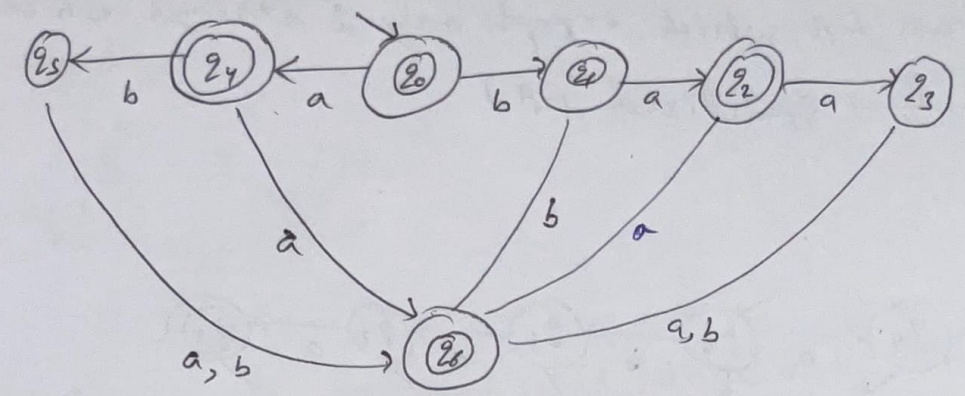
L = { a, aa, ab, aaa, aab, aba, abb, ------ }



to reject all strings start from b (dead-end)

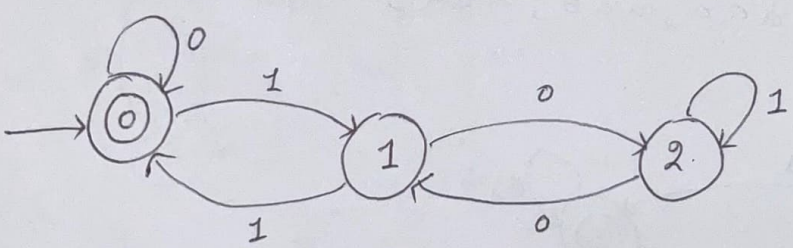
This is the state which make this automata Deterministic

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



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

Note!- states are equal to remainder



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

Step1 :- first make final state that is which give remainder 0

Step2:- Then we go to 11(3) that leads us to 0(remainder)

Step4:- Then we go to 100(4) that leads us to 1(remainder)

Step5:- Then we go to 101(5) that leads us to 2(remainder)

So Now our FA is complete.

Transition table

δ \ Σ	0	1
→ (q0)	q0	q1
q1	q2	q0
q2	q1	q2

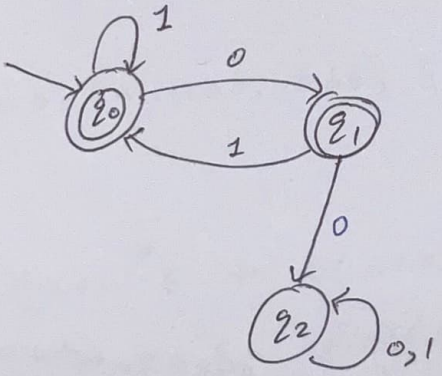
trick

for 4 states no is devide by 4

δ \ Σ	0	1
q0	q0	q1
q1	q2	q3
q2	q0	q1
q3	q2	q3

trick

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



$L = \{ \varepsilon, 0, 1, 01, 10, 11, 011, 111 \ldots$
$\ldots\ldots)$

Regular Expression

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

ε, a, b, ϕ

Regular set/ Language: $\{\varepsilon\}$ $\{a\}$ $\{b\}$ ϕ (NULL)

$\{\varepsilon\}$ ↓ 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

$a(a+b)^*$

$b(a+b)^*$

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

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

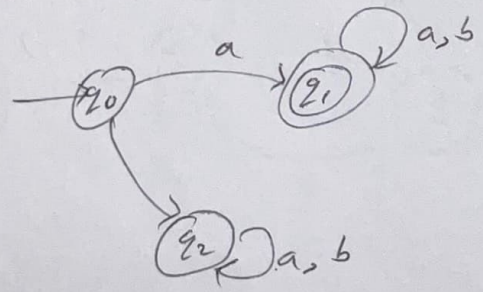
$L_1 \cdot L_2 = \{ ab, aba, abb, abbb, abba, ---- \}$

re^* =

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

$(+, \cdot, *)$ → are only allowed in Regular Expression

Q Automata that start with a



Q make a regular expression from prev automata

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

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

Ans $b^*.a^*$

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

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

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

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

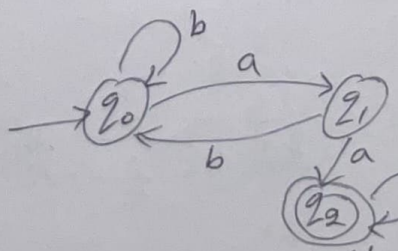
$L = \{ ab, ba, aab, abb, baa, aaab, aabb, abbb, abab, \dots \}$

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

Note :- a b ya b a first end ho bich me kuch bhi ho

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

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

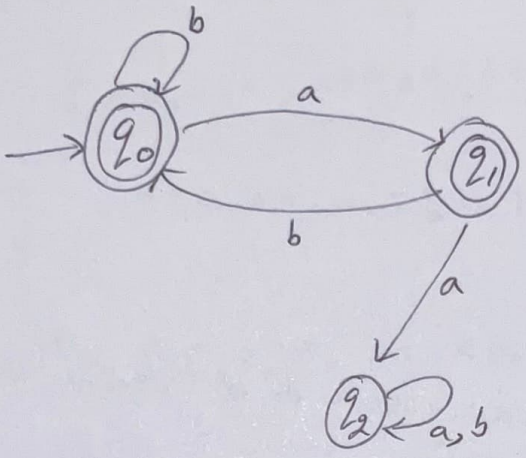


$(a+b)^* aa (a+b)^*$

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

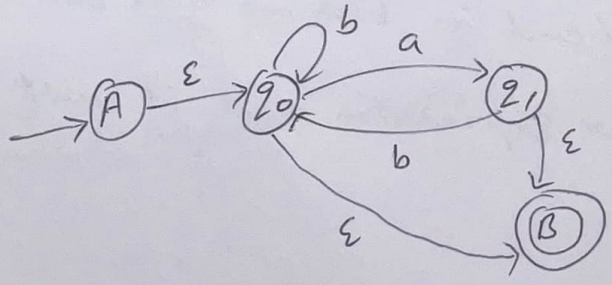
$$L = \{ \varepsilon, a, b, ab, ba, bb, aba, bba, abb, bbb, bab, \ldots \}$$



By using By pass theorem

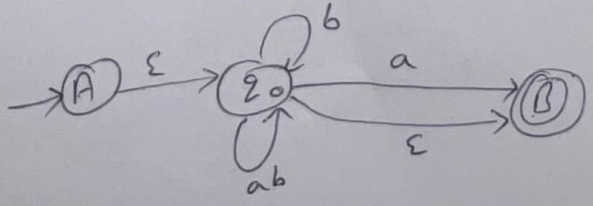
Step 1: eliminate q_2 edge because it goes to deadend state

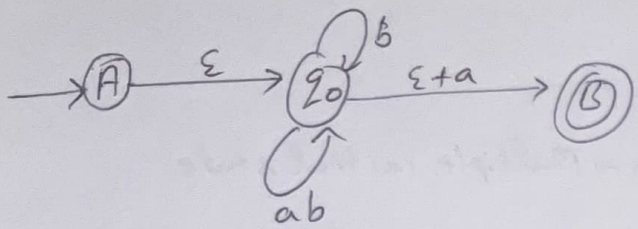
Step 2: Check is ^Initial state is unique or not ~~xxx~~ reachable

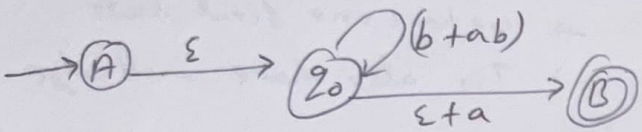


Now it is TG

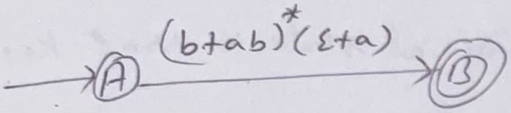
Now By pass q_1







By pass q_0



we also write it like this

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

TG → Transition graph

FA → Has one initial state

→ Every FA is TG but every TG is not FA

→ FA has no null edge

TG → Multiple initial state

→ one initial state

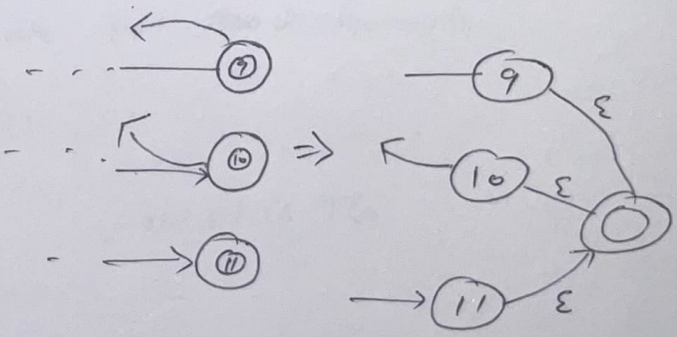
→ we have one final state

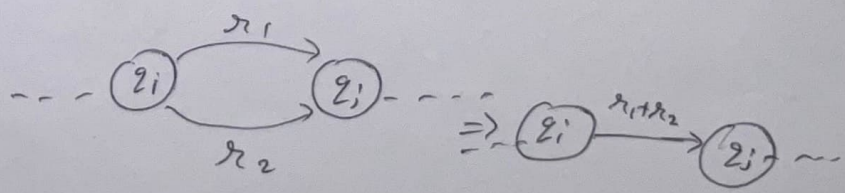
→ TG also have null edge

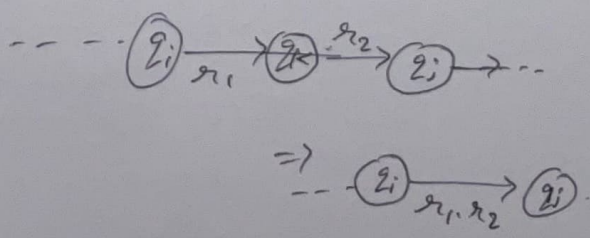
→ TG should have non-reenterable unique initial state

initial state ki traf koi edge nahi aane chyea → ○ ke alava

→ we should have unleavable final state (final state se koi edge bhar nahi jaae chyea)

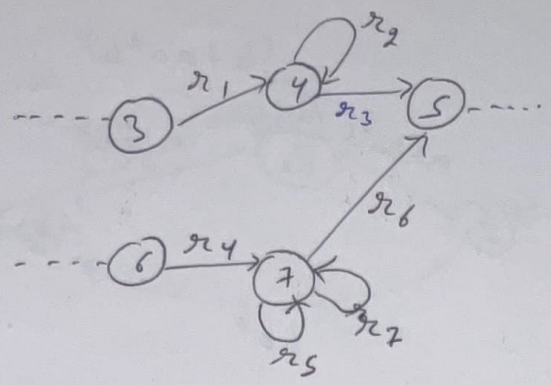




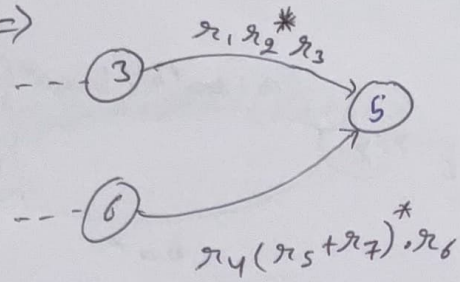


$r_1 \cdot r_2 \neq r_2 \cdot r_1$

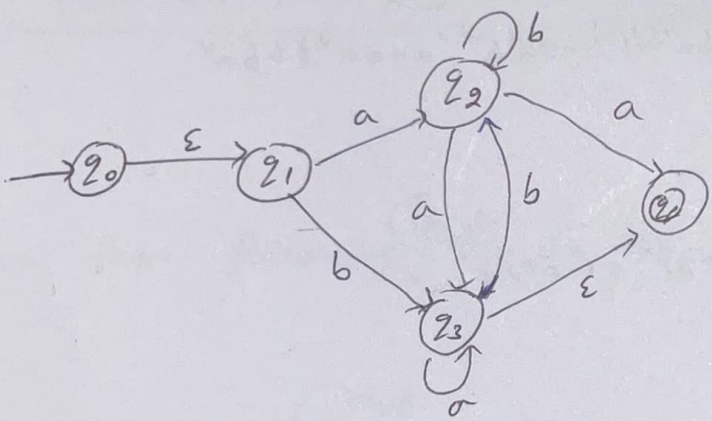
$r_1 + r_2 = r_2 + r_1$



$\Rightarrow$



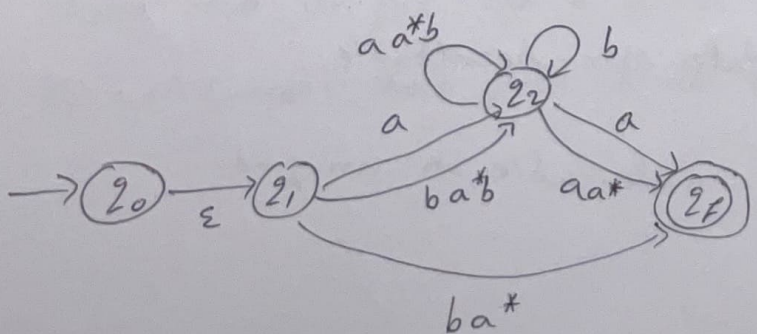
(9)



Eliminate q_3, q_2, q_1

Step 1:- Their is non-reentral state or not

Step 2:- Is final state is unleavable or unique

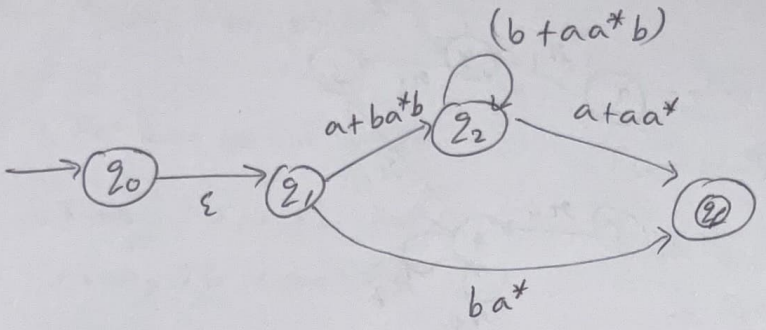


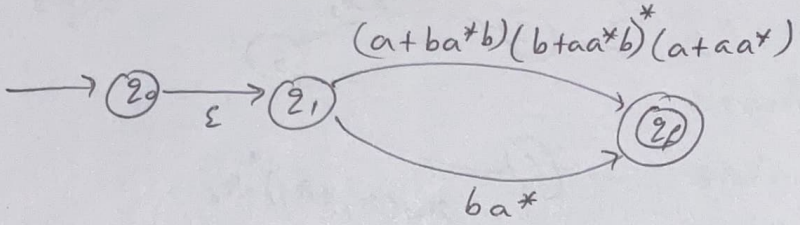
$q_1 \xrightarrow{via\, q_3} q_f \quad ba^*$

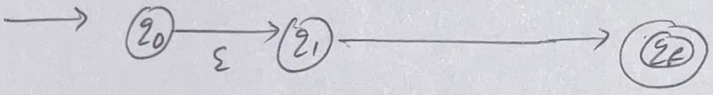
$q_2 \xrightarrow{via\, q_3} q_2 \quad aa^*b$

$q_1 \xrightarrow{via\, q_3} q_2 \quad ba^*b$

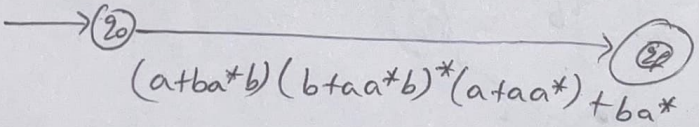
$q_2 \xrightarrow{via\, q_3} q_f \quad aa^*$



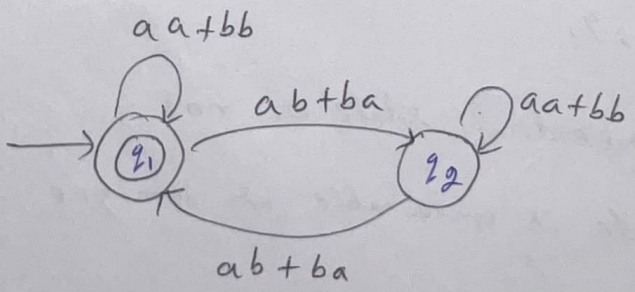




$(a+ba^*b)(b+aa^*b)^*(a+aa^*)+ba^*$

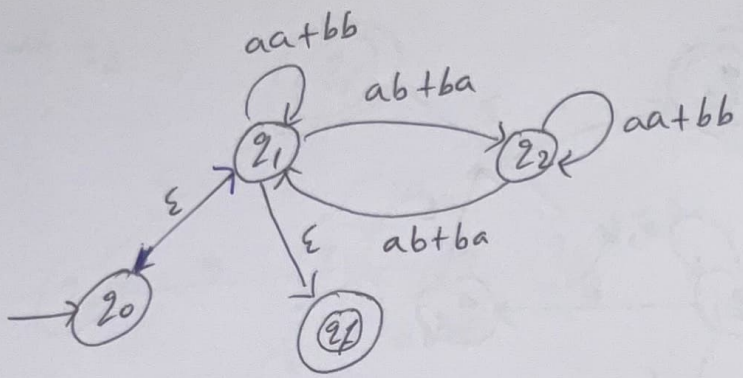


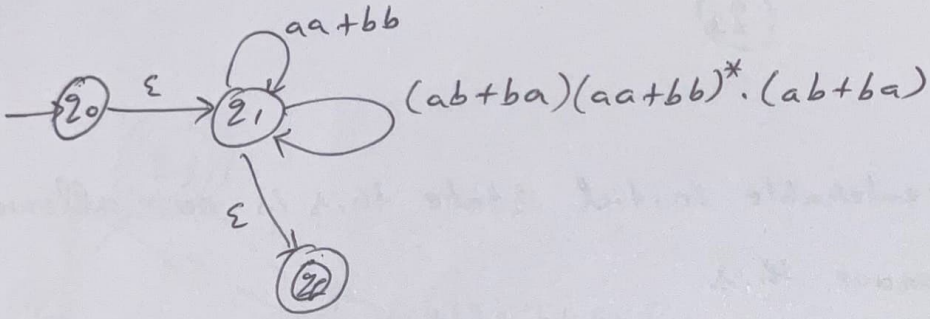
Q

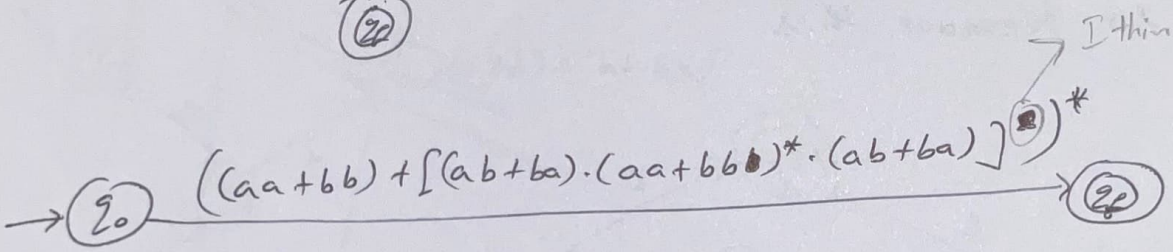


Step 1:- Their is initial state non-reentrable

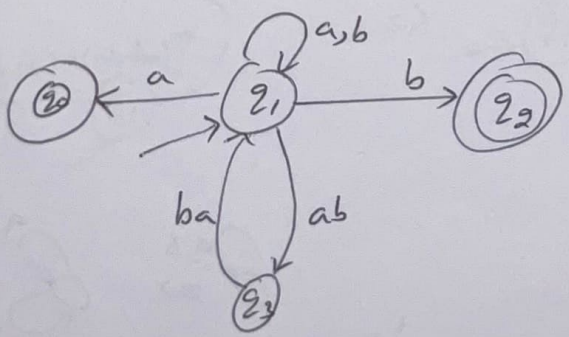
Step 2:- Is final state is unleavable or unique







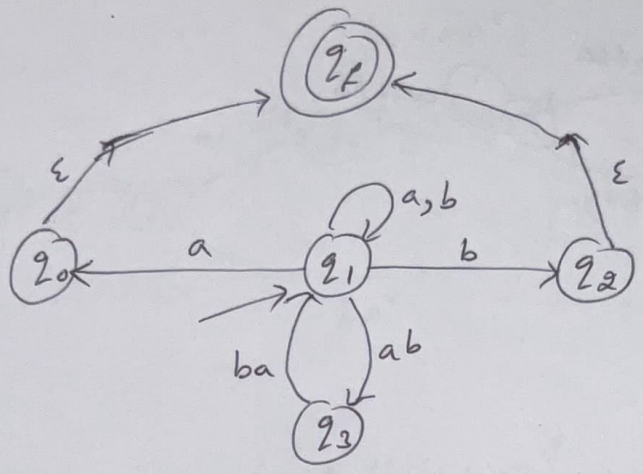
8 Convert then following TG to RE



There are unique but 2 final state that not allowed in RE

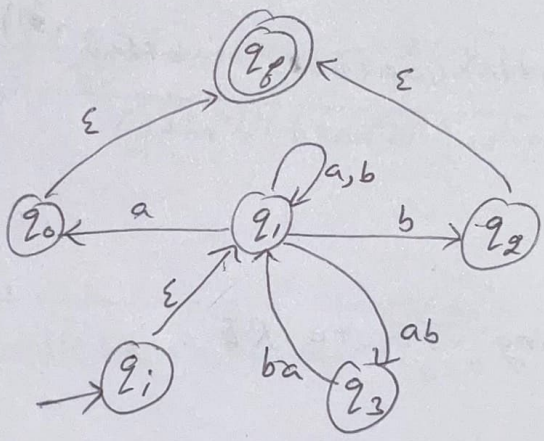
So we convert these final states to unique final stale

Step 1



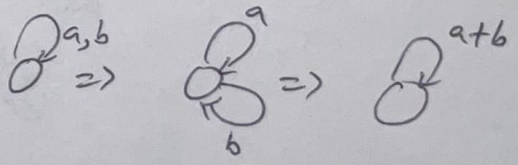
we have Reenterable initial state this is not allowed in RE so remove this

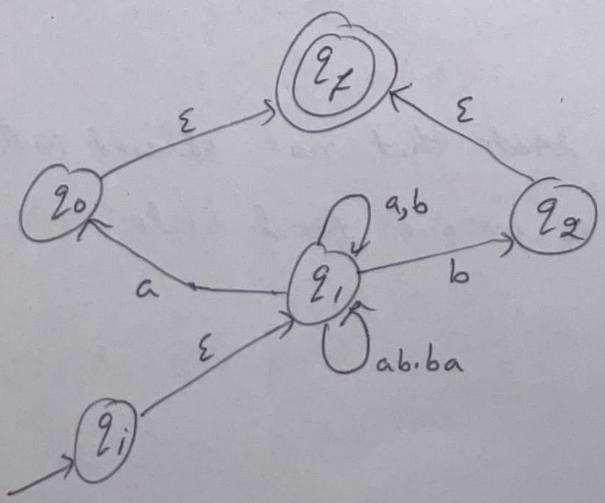
Step 2



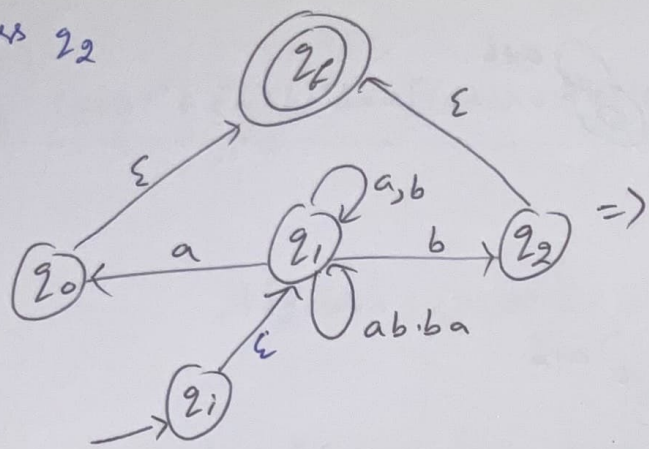
Step 3

By Pass q_3

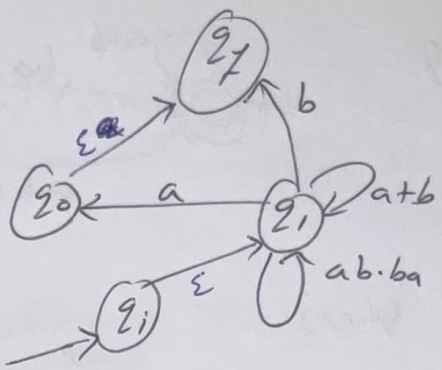


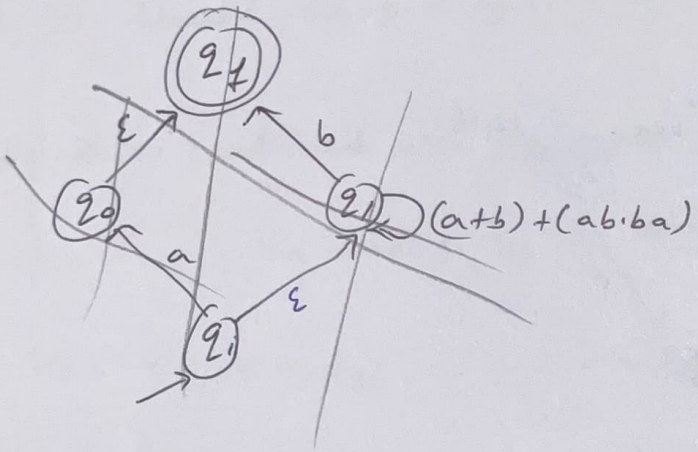


Bypass q_2

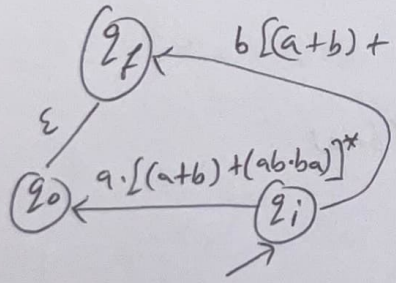


=>

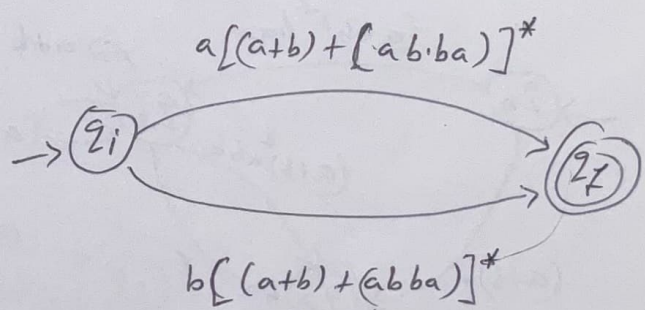


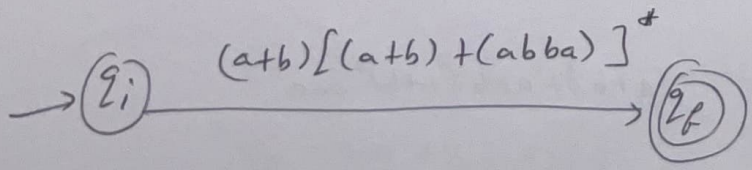


Step 4 :- Bypass q_1

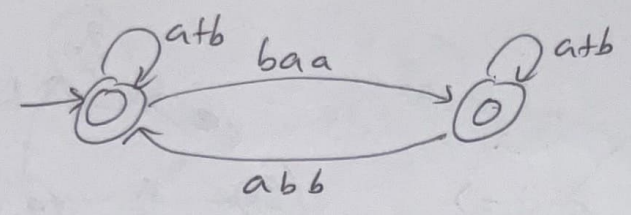


=>

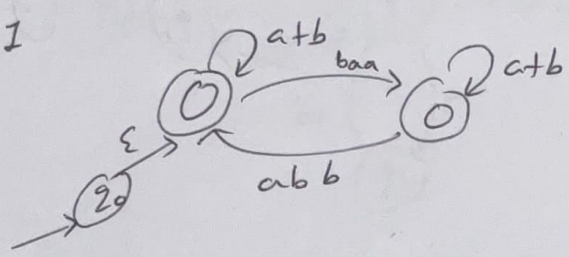




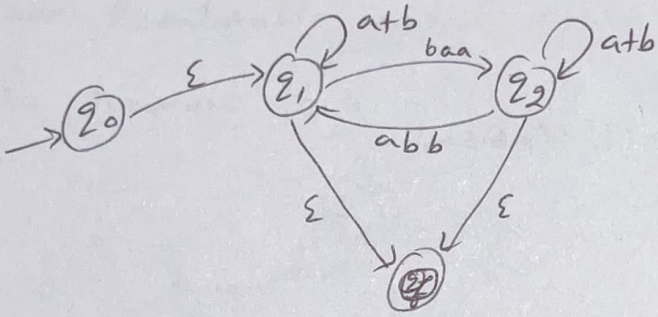
0



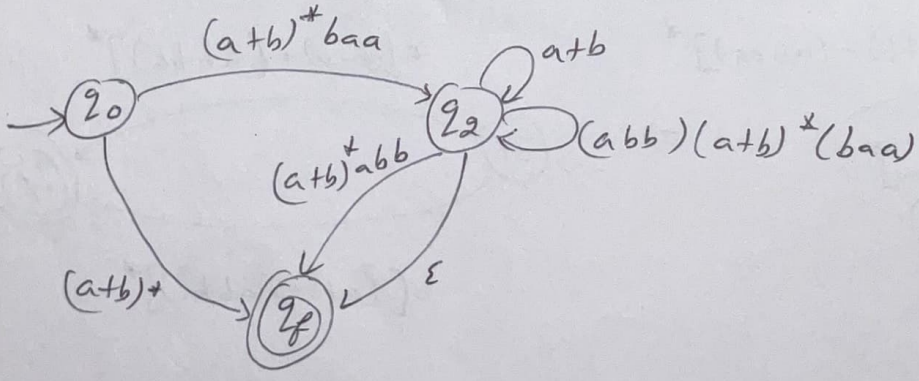
Step 1

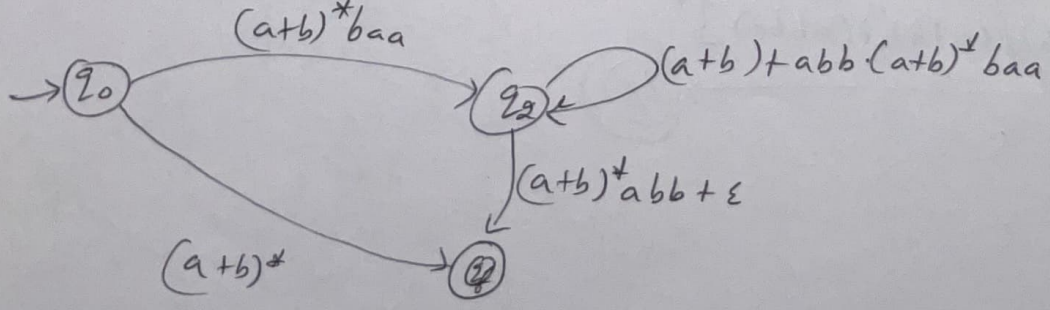


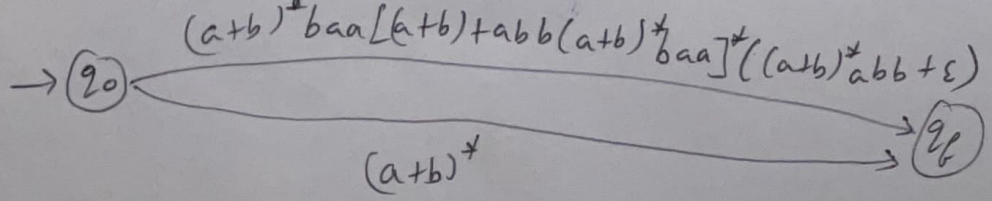
Step 2



Bypass q_1







q_0 ——— $(a+b)^* + (a+b)^*baa[(a+b) + abb(a+b)^*baa]^*((a+b)^*abb+\varepsilon)$ ——— q_f

Algebra Expression using :- Ardan's Thearem

(1) The graph should not have null more

(2) It should have single initial state

If these conditions are true, then we can take vertises $v_1 - - - - - v_n$ where v_1 is the initial state then

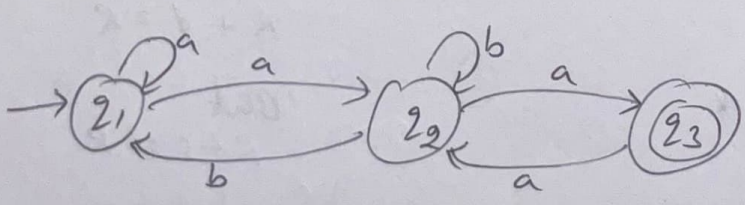
$$v_1 = v_1\alpha_{11} + v_2\alpha_{21} - - - - - v_n\alpha_{n1} + \varepsilon$$

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

$$\vdots$$

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

Q



$$q_1 = q_1a + q_2b + \varepsilon$$

$$q_2 = q_1a + q_2b + q_3a$$

$$q_3 = q_2a$$

Property :-

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

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

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

Prove that the language represented by the regular expression above 1 is the set of strings in which every zero is imidiatly followed by at least two one's

Let $R_1 = 1^*(011)^*$

$R = \varepsilon + R_1 R_1^*$ from above

From property

$R = R^*$

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

$E + RR^* = R^*$

$R + \phi = R$

But

$R + \varepsilon \neq R$

from (P1)

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

Remember Properties

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

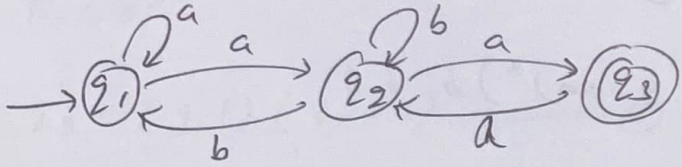
2. $\varepsilon + RR^* = R^* = \varepsilon + R^*R$

3. $R + \phi = R$ But $R + \varepsilon \neq R$

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

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

Q



$q_1 = q_1 a + q_2 b + \varepsilon$ — (1)

$q_2 = q_1 a + q_2 b + q_3 a$ — (2)

$q_3 = q_2 a$ — (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 Ardam's theaorem:-

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

$$R = Q + RP$$

has a unique Solution given by

$$R = QP^*$$

Q

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

$$R \quad = \quad Q \quad + \quad R \quad P$$

$$q_2 = (q_1 a)(b + aa)^* \quad \text{——} (4)$$

Substituting (4) in (1)

$$q_1 = q_1 a + (q_1 a (b + aa)^*) b + \varepsilon$$

$$q_1 = \varepsilon + q_1 (a + a(b + aa)^* b)$$

$$R = Q + R \quad P$$

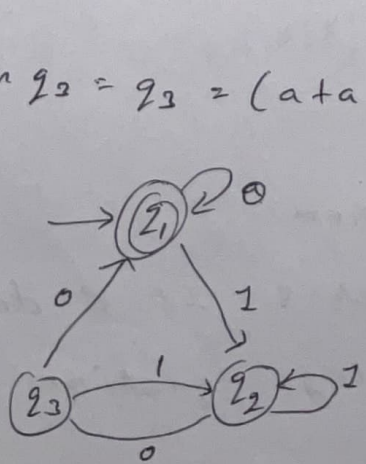
Using Arden's theorem

$$q_1 = \varepsilon \cdot (a + a(b + aa)^* b)^* \quad \text{——} (5)$$

Putting q_1 in q_2 $\quad q_2 = (a + a(b + aa)^* b)^* a (b + aa)^*$

Putting q_2 in $q_3 = q_3 = (a + a(b + aa)^* b)^* a (b + aa)^* a$ Ans

Q



$q_1 = q_{10} + q_{30} + \varepsilon$ — ①

$q_2 = q_{11} + q_{21} + q_{31}$ — ②

$q_3 = q_{20}$ — ③

from eqn 1 & 3

$q_1 = q_{10} + (q_{20}) 0 + \varepsilon$ — ④

$q_2 = \underbrace{q_{11}}_{R} + \underbrace{q_2}_{R}\underbrace{(1+0.1)}_{P}$

using: $R \to Q$, $R \to R$, $P \to P$

$R = QP^*$

$q_2 = (q_{11})(1+0.1)^*$ — ⑥

Put in eqn 6 in eqn 4

$q_1 = q_{10} + (q_{11})(1+0.1)^* \cdot 00 + \varepsilon$ — ⑦

$\frac{q_1}{R} = \frac{\varepsilon}{Q} + \frac{q_1}{R}\frac{(0+1(1+0.1)^* 00)}{P}$

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

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

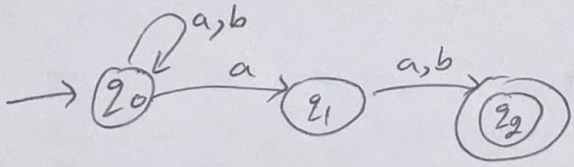
$L = \{ aa, ab, aaa, aab, bab, baa, \ldots \}$

we have to restrict a ke baad sirf 1 single letter aae $a(a+b)$

a se phale kuch bhi aae

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

NFA



Contruct NFA to FA

δ / Σ	a	b
→ q_0	$[q_0, q_1]$	q_0
q_1	q_2	q_2
q_2	ϕ	ϕ

→ this ϕ say 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_1, q_3

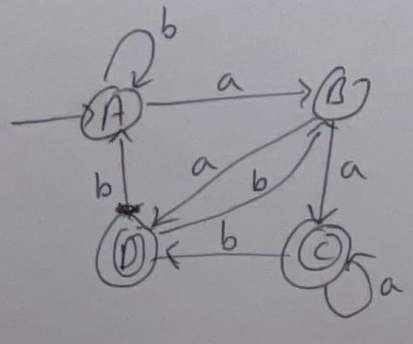
if q_3 not go anywhere so can't take ϕ in our set

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

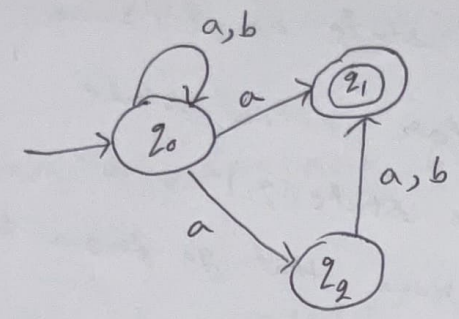
step 8:- So at last one make final state

final state is that state which contain alw q_2

Σ / Q	a	b
→ 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

δ / Σ	a	b
→q_0	$[q_0, q_1, q_2]$	q_0
(q_1)	ϕ	ϕ
q_2	q_1	q_1

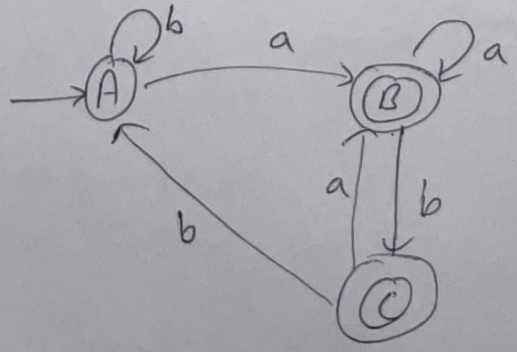
Transition table for DFA

δ / Σ	a	b
→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

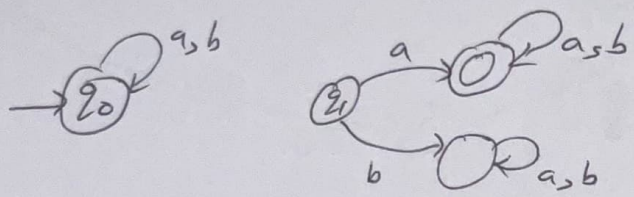
Here for q_0 it go to q_0
q_1 it go nowhere
q_2 it go to q_1
so we add q_0, here q_1

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

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



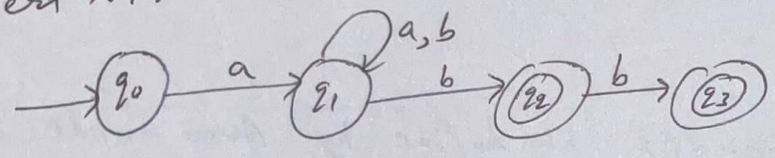
when their is no way to reach to initial to final state so our language is null but it is NFA



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 Determnistic NDFA or NFA is define by 5 tuple $(Q, \Sigma, \delta, s, F)$

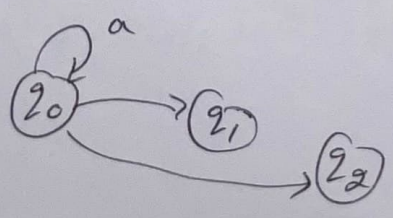
Q = finite ~~set of~~ non empty set of states

Σ = finite non empty set of symbols (letters)

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

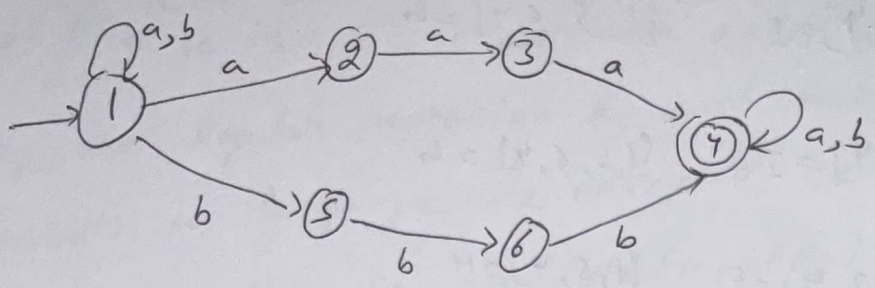
s = $s \in Q$

F = 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 sates DFA me hogi us se kam nahi hogi

Σ \ Q	a	b
1	[1,2]	[1,5]
2	3	ϕ
3	4	ϕ
4	4	4
5	ϕ	6
6	ϕ	4

Transition table to DFA

Σ \ Q	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] = F
[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

$[1,5,6,4]=G$	$[1,2,4]=I$	$[1,5,6,4]=G$
$[1,5,4]=H$	$[1,2,4]=I$	$[1,5,6,4]=G$
$[1,2,4]=I$	$[1,2,3,4]=F$	$[1,5,4]=H$

Algorithm which compute $FA_1 + FA_2$ 5 marks (41)

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 their is an $FA(FA_1 + FA_2)$ that accepts the language $l_1 + l_2$ represend by the regular expression

$r_1 + r_2$

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 $z_1, z_2 \dots$ where z something equal to x something or y something (zk (k would be greater than or equal m, n)

step2:- the combination state x_{start} / y_{start} (→ or) 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_{new} after reading letter p in FA_1]

or [y_{new} after reading letter p in FA_2]

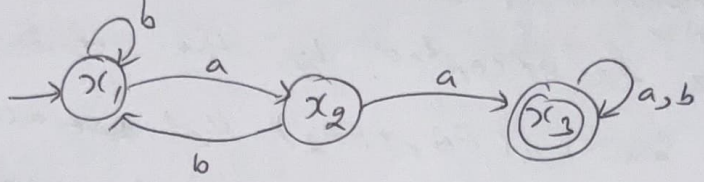
Note:- This algorithm work only in deterministic FA so fistly check FA is deterministic or not

Both FA's have diff state names must x_1, x_2, x_3 y_1, y_2, y_3

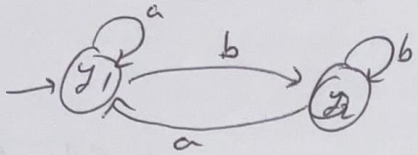
Construct a FA that have substring aa

① Either string have aa, end with b or both

FA1:



FA2:



→ End's with b

Transition table for FA_1

	Σ	a	b
→	x_1	x_2	x_1
	x_2	x_3	x_1
(final)	x_3	x_3	x_3

Transition Table for FA_2

	Σ	a	b
→	y_1	y_1	y_2
(final)	y_2	y_1	y_2

Max states are FA_1 states * FA_2 states

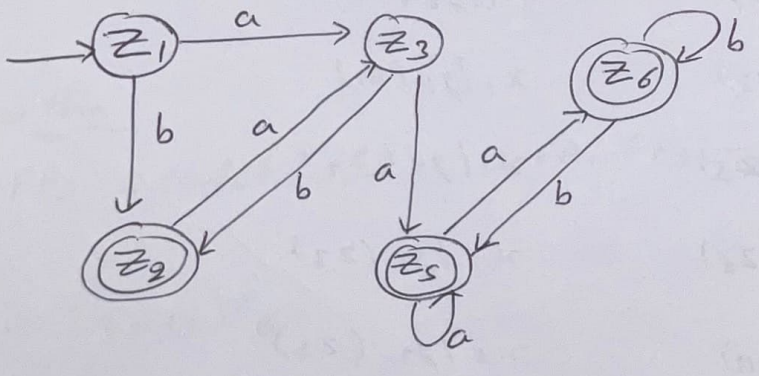
Transition Table for $FA_1 + FA_2$

	Σ	a	b
→	$z_1 = x_1 \mid y_1$	$x_2 \mid y_1 = z_3$	$x_1 \mid y_2 = z_2$
(final)	$z_2 = x_1 \mid y_2$	$x_2 \mid y_1 = z_3$	$x_1 \mid y_2 = z_2$
	$z_3 = x_2 \mid y_1$	$x_3 \mid y_1 = z_5$	$x_1 \mid y_2 = z_2$
(final)	$z_4 = x_2 \mid y_2$	$x_3 \mid y_1 = z_5$	$x_1 \mid y_2 = z_2$
(final)	$z_5 = x_3 \mid y_1$	$x_3 \mid y_1 = z_5$	$x_3 \mid y_2 = z_6$
(final)	$z_6 = x_3 \mid y_2$	$x_3 \mid y_1 = z_5$	$x_3 \mid y_2 = z_6$

finital state is where both FA finital state are there final state is where FA_1 final or FA_2 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 $FA_1 + FA_2$

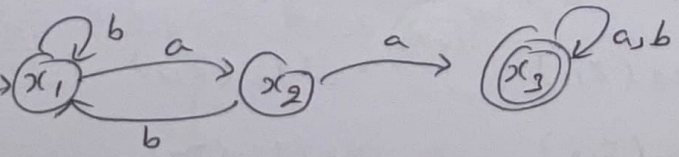
Note!- Remove that row where we can't go but don't remove inital state row
here we can't reach to z_4 state so we remove z_4 row completely
In a & b column we don't have z_4



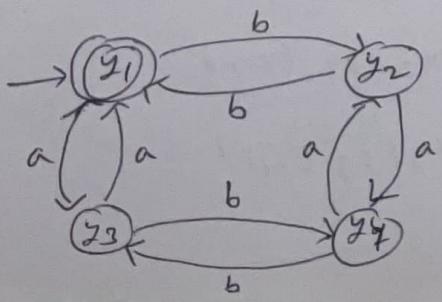
Q Either string have even no of a or even no of b or both

find FA1 + FA2

FA_1 :



FA_2 :



Transition table for FA_1

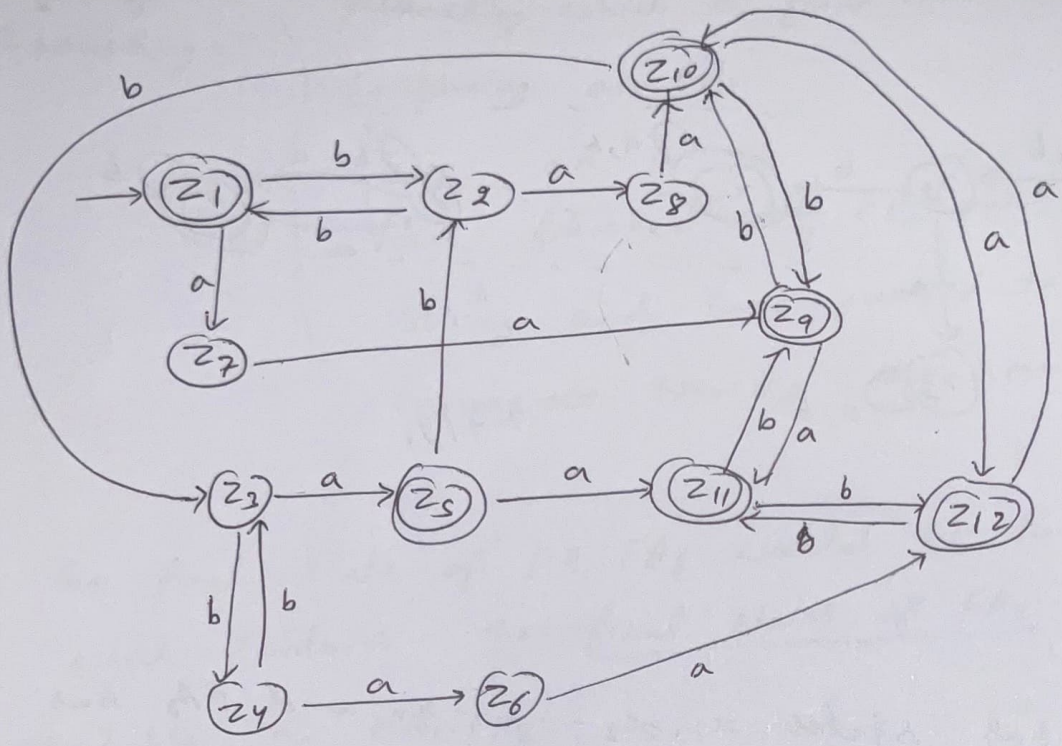
Σ / Q	a	b
$\rightarrow x_1$	x_2	x_1
x_2	x_3	x_1
(*) x_3	x_3	x_3

Transition table for FA_2

Σ / Q	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 $FA_1 + FA_2$

Σ / Q	a	b
$\rightarrow$ (*) $z_1 = x_1/y_1$	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 (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)
(*) $z_5 = x_2/y_1$	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)
(*) $z_9 = x_3/y_1$	x_3/y_3 (z_{11})	x_3/y_2 (z_{10})
(*) $z_{10} = x_3/y_2$	x_3/y_4 (z_{12})	x_3/y_1 (z_9)
(*) $z_{11} = x_3/y_3$	x_3/y_1 (z_9)	x_3/y_4 (z_{12})
(*) $z_{12} = x_3/y_4$	x_3/y_2 (z_{10})	x_3/y_3 (z_{11})



Algorithm

$FA_1 \cdot FA_2$ => concatination

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

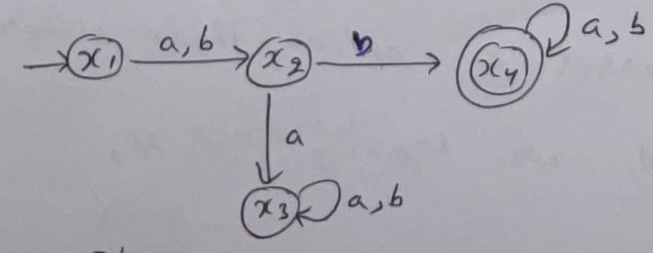
r_1 : $(a+b)^* a$

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

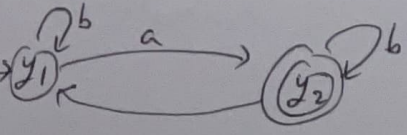
↓

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

FA_1 :



FA_2 :



FA_1

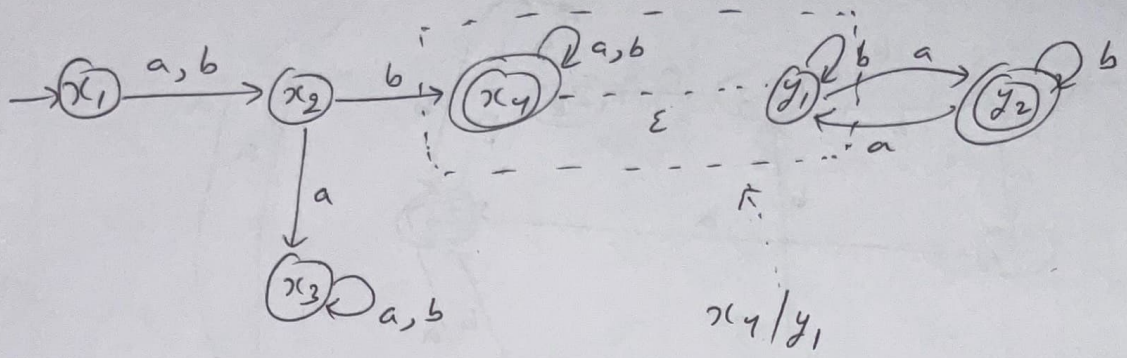
$(a+b)\,b\,(a+b)^*$

second letter must be b

FA_2

odd no of a's

we take initial state of FA_1 and final state of FA_2



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

$z_1, z_2 \dots z_k$

1) The initial state of $FA_1 \cdot FA_2$ will be the initial state of FA_1

2) we make a z state for every non-final state in FA_1 reached before ever hitting a final state in FA_1

3) for every final state in FA_1 we establish a z state that expresses an option that either we are continuing in FA_1 or begining with FA_2

$Z_{something}$ = { $x_{something}$ which is final state and still continuing on FA_1

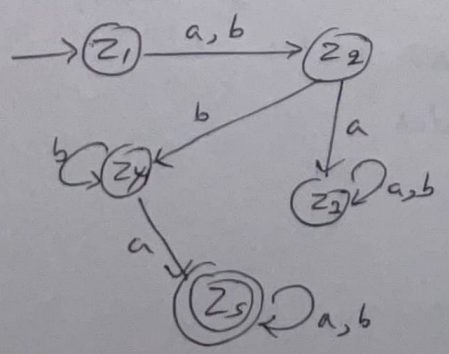
OR

have finished with FA_1 part of the input string and have jumped to FA_2 to commence tracing the remainder of string

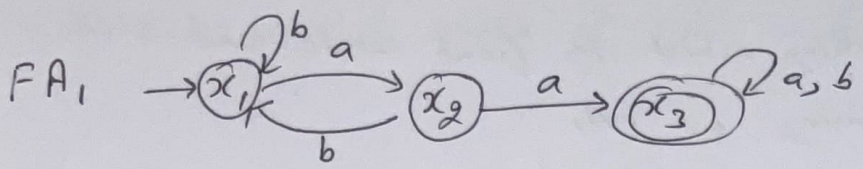
4) the final state of $FA_1 . FA_2$ would be those states which contains the final states of FA_2. ~~transition~~

Transition table of $FA_1 . FA_2$

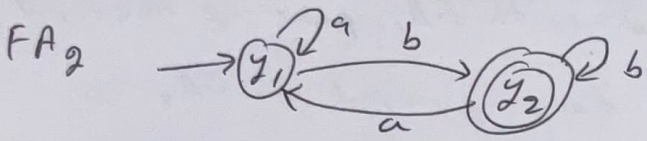
	ε	a	b
$z_1 = x_1$		$x_2 (z_2)$	$x_2 (z_2)$
$z_2 = x_2$		$x_3 (z_3)$	$x_4 \mid y_1 (z_4)$
$z_3 = x_3$		$x_3 (z_3)$	$x_3 (z_3)$
$z_4 = x_4 \mid y_1$		$x_4 \mid y_1 \mid y_2 (z_5)$	$x_4 \mid y_1 \mid y_1 = x_4 \mid y_1 (z_4)$
$(z_5) = x_4 \mid y_1 \mid y_2$		$x_4 \mid y_1 \mid y_2 (z_5)$	$x_4 \mid y_1 \mid y_2 (z_5)$



8



aa is mandetory



ended with b

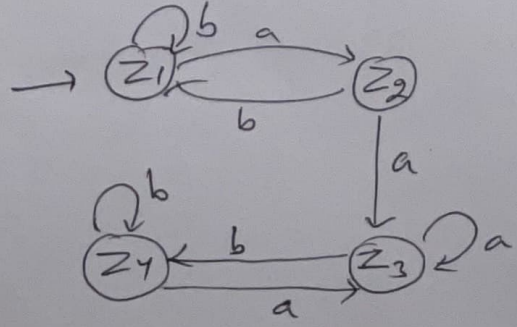
Transition table for $FA_1 \cdot FA_2$

	ε	a	b
→ $Z_1 = x_1$		$x_2 (z_2)$	$x_1 (z_1)$
$Z_2 = x_2$		$x_3/y_1 (z_3)$	$x_1 (z_1)$
$Z_3 = x_3/y_1$		$x_3/y_1/y_1 = x_3/y_1 (z_3)$	$x_3/y_1/y_2 (z_4)$
(Z4) $= x_3/y_1/y_2$		$x_3/y_1 (z_3)$	$x_3/y_1/y_2 (z_4)$

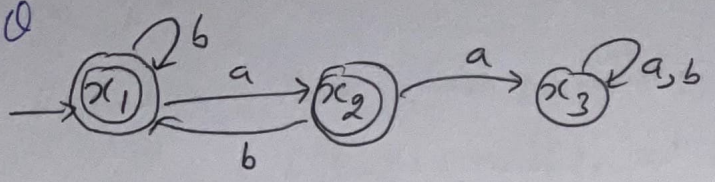
when you go to the final state of FA_1 so it give two option just on FA_1 final state of jump to the initial state of FA_2

Note!- final state is that which contain y_2

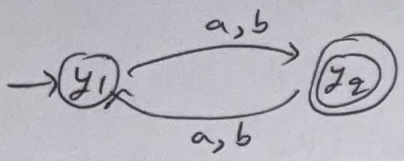
Note!- New states are depend upon how many states on FA_2 if FA_2 have more states so $FA_1 \cdot FA_2$ have more states



Q



not accept aa



odd no of a & b

we have x_1 is inital and final state also so make FA_1 and FA_2 initial state as initial state (x_1/y_1)

	ε	a	b
→ $z_1 = x_1/y_1$		$x_2/y_1/y_2\,(z_2)$	$x_1/y_1/y_2\,(z_3)$
$z_2 = x_2/y_2/y_2$		$x_3/y_1/y_2\,(z_4)$	$x_1/y_1/y_2\,(z_3)$
$z_3 = x_1/y_2/y_2$		$x_2/y_1\,(z_2)/y_2$	$x_1/y_1/y_2\,(z_3)$
$z_4 = x_3/y_1/y_2$		$x_3/y_1/y_2\,(z_4)$	$z_3/y_1/y_2\,(z_4)$
$z_5 = y_1$		~~$y_2\,(z_4)$~~	~~$y_2\,(z_4)$~~

Construction of Minimum state Finite Automata

K.L.P Mishra, N. Chandra

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 Q

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

As it is difficult to construct $\delta(q_1, x)$ and $\delta(q_2, x)$ 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 element 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 $a \in \Sigma$

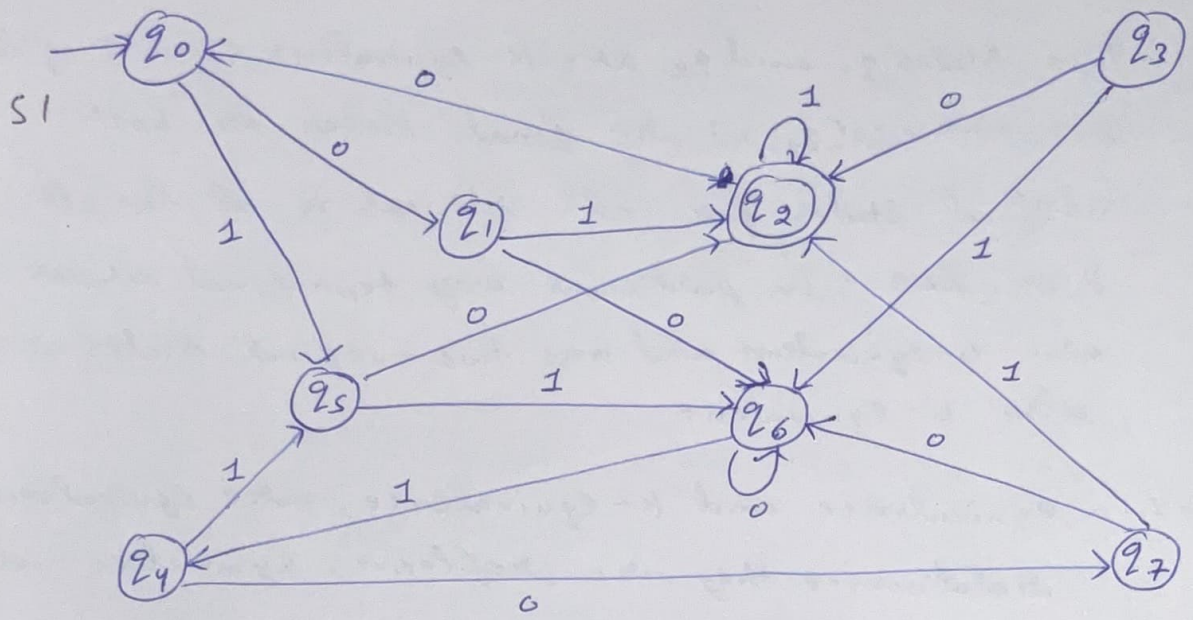
step 1:-(Construction of Π_0) By defination of 0-equivalence $\Pi_0 = \{Q_1^0, Q_2^0\}$
$Q_1^0 \rightarrow$ set of final states and $Q_2^0 = Q - Q_1^0$

step 2:- (Construction of Π_{k+1} from Π_k). Let Q_i^k be any subset in Π_k. If q_1 and q_2 are in Q_i^k they are $(k+1)$ equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are k-equivalent. find $\delta(q_1, a)$ & $\delta(q_2, a)$ are ~~be equivalent~~ class in Π_k

step 3:- Construct Π_n for $n = 1, 2 - - - -$ 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 step3 i.e the elements of Π_n. The state table is obtained by replacing a state q by the corr equivalence class $[q$

Q



S1

State/Σ	a	b
→ q_0	q_1	q_5
q_1	q_6	q_2
(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

$Q_1^\circ = F = \{q_2\}$ $\quad Q_2^\circ = Q - Q_1^\circ$

set of final states → Q_1°, set of non final states → Q_2°

$[Q_1^\circ, Q_2^\circ]$

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

in set Q_1°

q_2 can't be further partation.

Consider

$\{q_0$ and $q_1\}$

$q_0 \xrightarrow{a} q_1$ $\quad q_1$ and $q_6 \in Q_2^\circ$

$q_1 \xrightarrow{a} q_6$

$q_0 \xrightarrow{b} q_5$ $\quad q_5 \in Q_2^\circ$ but $q_2 \in Q_2^\circ$

$q_1 \xrightarrow{b} q_2$ $\quad$ Therefore $\{q_0$ and $q_1\}$ is not 1-equivalent

$\{q_0$ and $q_4\}$ → they are on Q_2'

$\{q_0, q_6\}$ → they are on Q_2'

$\{q_1, q_7\}$ → they are on Q_2' with a and Q_1' with b

$\{q_3, q_5\}$ "

$\{q_4, q_6\}$

→ जो तीनों हमने ऊपर ही set में रख लिए हैं

क्योंकि $\{q_4, q_6\}$ आ $\{q_1\}$ आ।

$Q_1 = \{q_0, q_4, q_6\}$

$Q_2 = \{q_1, q_7\}$

$Q_3 = \{q_3, q_5\}$

$\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

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

$\{q_2\}$

$\{q_0, q_4\}$

$\{q_1, q_7\}$

$\{q_3, q_5\}$

$\{q_6\}$

→ 2-equivalent

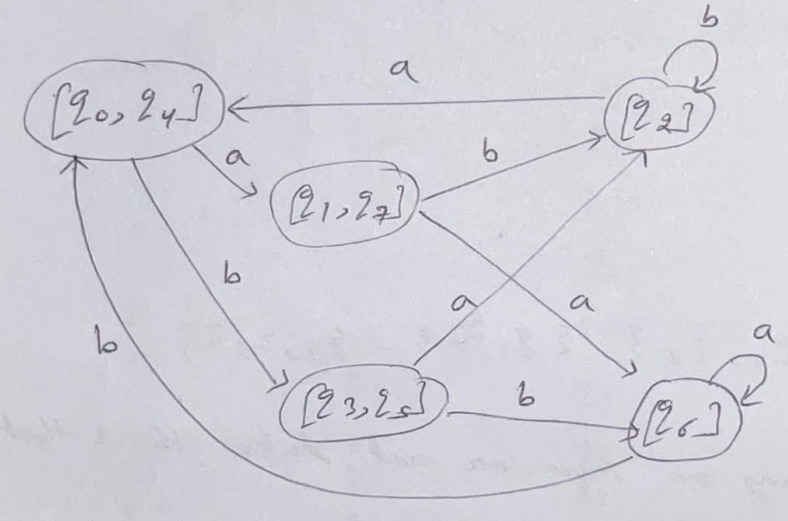
$\{q_0, q_6\}$ ✗ because q_0 with a and q_6 with a go to diff states

$\{q_0, q_4, q_6\}$ $\{q_1, q_7\}$ → they go to diff

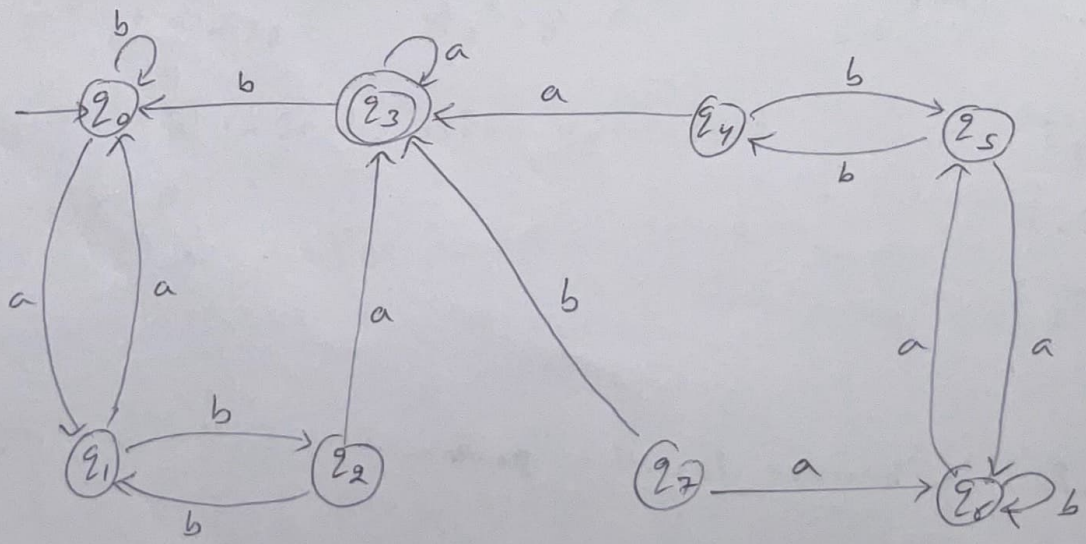
$\pi_3 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

δ	a	b
$\to [q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
(q_2)	$[q_0, q_4]$	$[q_2]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$

$q_0^{q_4}$ with b go to q_5 so we take q_5 and q_5 comes with q_3 in Π_3



Q



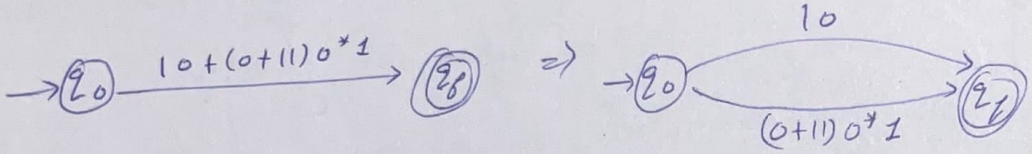
109 page

Transition System containing ε Moves

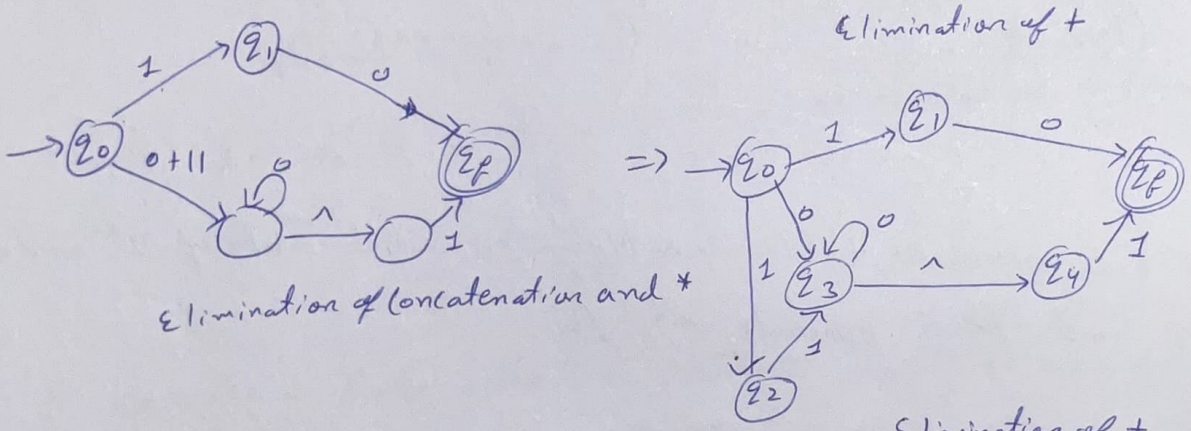
Q Construct a DFA with reduced states equivalent to the r.e $10+(0+11)0^*1$ | 69 page

K L P Mishra & N Chandra

Construction of NDFA The NDFA is constructed by eliminating the operation + (concatenation and * and the $\wedge$-moves in successive steps

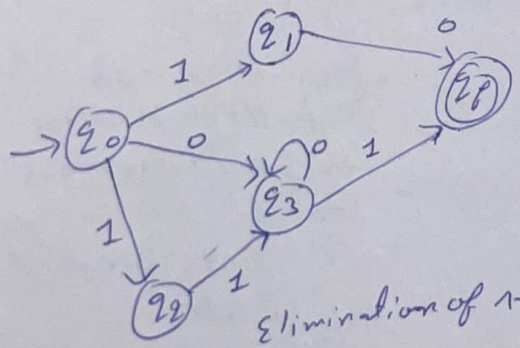


Elimination of +



Elimination of concatenation and *

Elimination of +

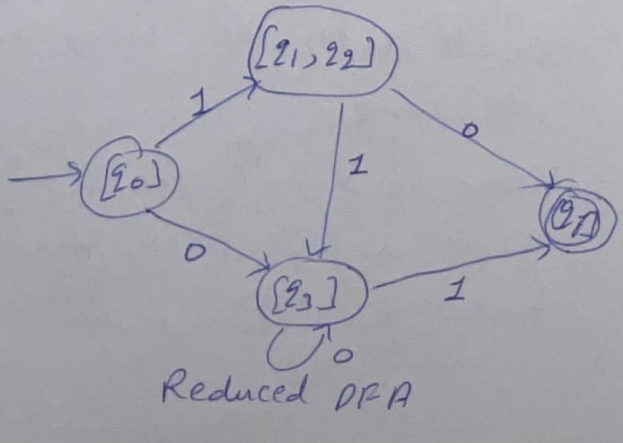


Elimination of $\wedge$ moves

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

(DFA)

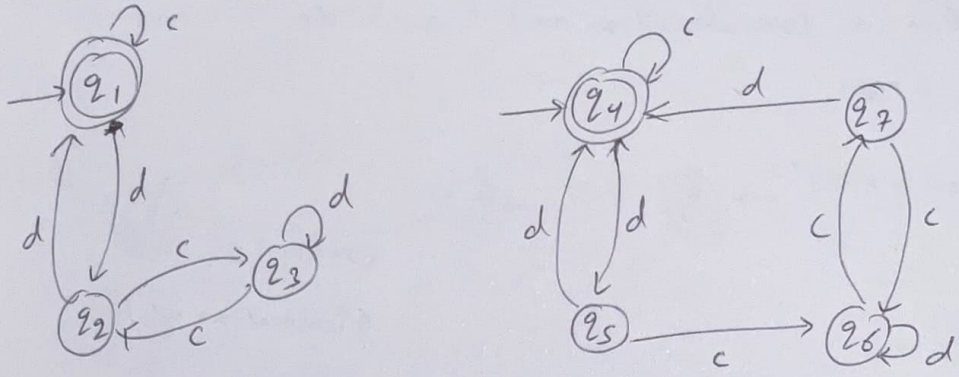
Q	Q_0	Q_1
$\rightarrow[q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1, q_2]$	$[q_f]$	$[q_3]$
$[q_f]$	ϕ	ϕ
ϕ	ϕ	ϕ



Reduced DFA

Equivalence of two FA

Determines whether m and m' are equivalent



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

q_1, q_1'	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_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

after making each state check it belong 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 categorie 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 viceversa) at every row so m and m' are equivalent

Transition system containing ε moves

Part 6

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

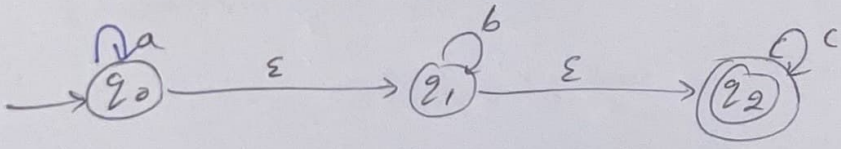
Step1:- find all the edges starting from v_2

Step2:- Duplicate all these edges starting from v_1 without changing the edge labels

Step3:- If v_1 is an initial state make v_2 also as an initial state

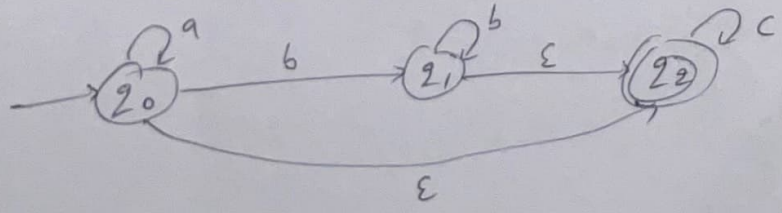
Step4 If v_2 is a final state make v_1 also as a final state

Q Consider a finte automaton with ε moves obtain an equivalent automaton 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

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



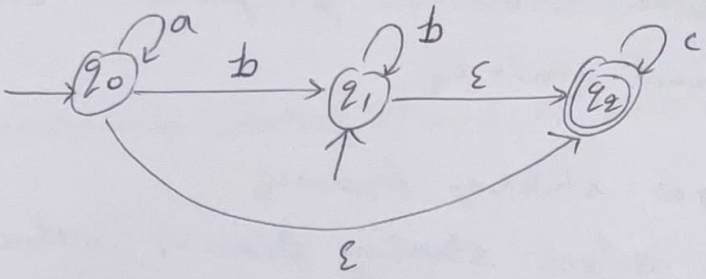
$q_1 \xrightarrow{b} q_1$

therefure, we will duplicate this edge have q_0 as a source state

$q_0 \xrightarrow{b} q_1$

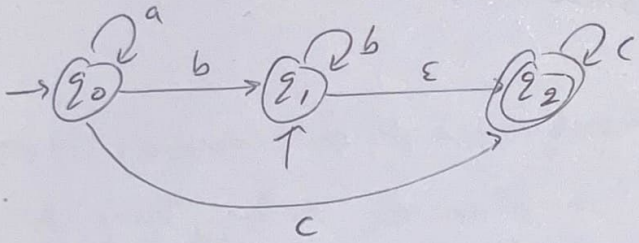
4. if v_1 is initial state make v_2 also as initial state

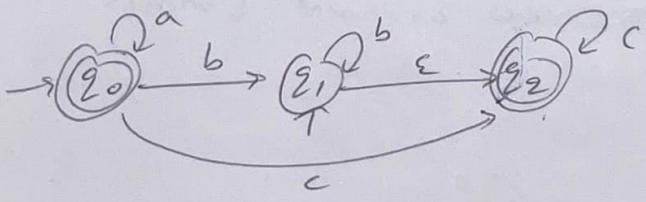
(q_0) (q_1)



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

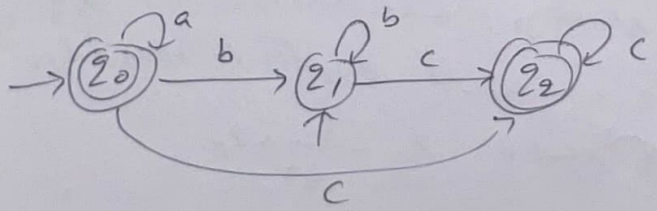
$q_2 \xrightarrow{c} q_2$ so $q_0 \xrightarrow{c} q_2$

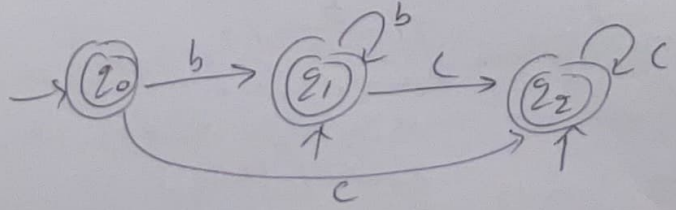




$q_1 \xrightarrow{\varepsilon} q_2$

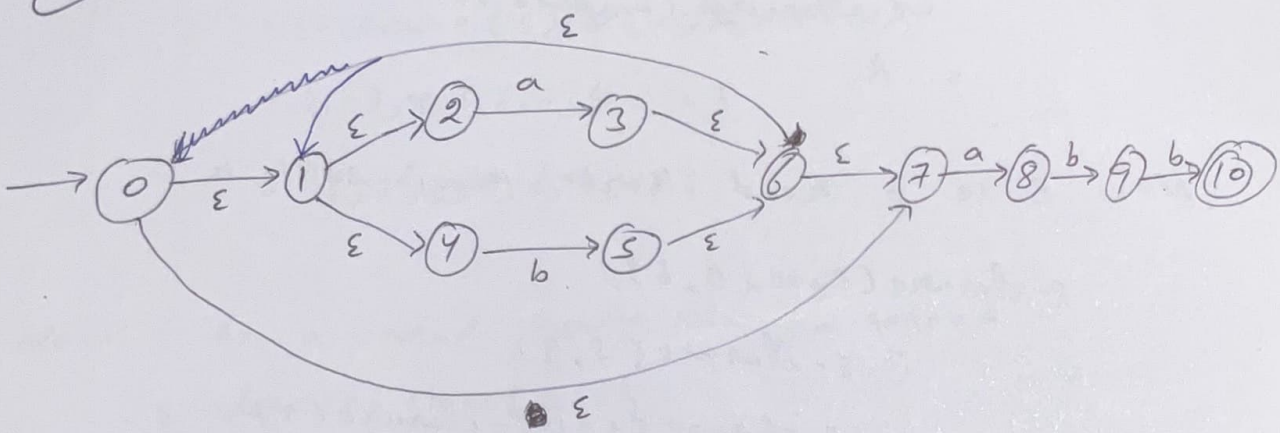
$q_2 \xrightarrow{c} q_2$ so $q_1 \xrightarrow{c} q_2$





this algorithm fails because we have many initial states

NFA-ε to DFA



firstly find ε-closer of (0) → Initial state

A = ε-closer(0) = {0, 1, 2, 4, 7}

Now A se a read karke kha ja rahe h

ε-closure(Move{A, a})

= ε-closure(3, 8)

= ε-closure(3) ∪ ε-closure(8)

= {3, 6, 7, 1, 2, 4} ∪ {8}

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

Now A se b read karke kha ja sakte h

ε-closure(Move{A, b})

= ε-closure(5)

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

Now B se a read karke kha ja sakte h

ε-closure (Move{B, a})

= ε-closure(3, 8)

we already compute it

= B

Now B se b read karke kha ja sakte h

ε-closure (Move{B, b})

= ε-closure(5, 9)

= ε-closure{5) $\cup$ ε-closure(9)

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

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

Now C se a read karke kha ja sakte h

ε-closure(Move{C, a})

= ε-closure (3, 8)

= B

Now C se b read karke kha ja sakte h

ε-closure (Move{C, b})

= ε-closure{5}

= C

Now D se a read karke kha ja sakte h

ε-closure (Move{D, a})

= ε-closure (3, 8)

= B

Now D se b read karke kha ja sakte h

ε-closure (Move {D, b})

$= \varepsilon$-closure (5, 10)

$= \varepsilon$-closure (5) $\cup$ ε-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})

$= \varepsilon$-closure (3, 8)

$= B$

Now E se b read karke kha ja sakte h

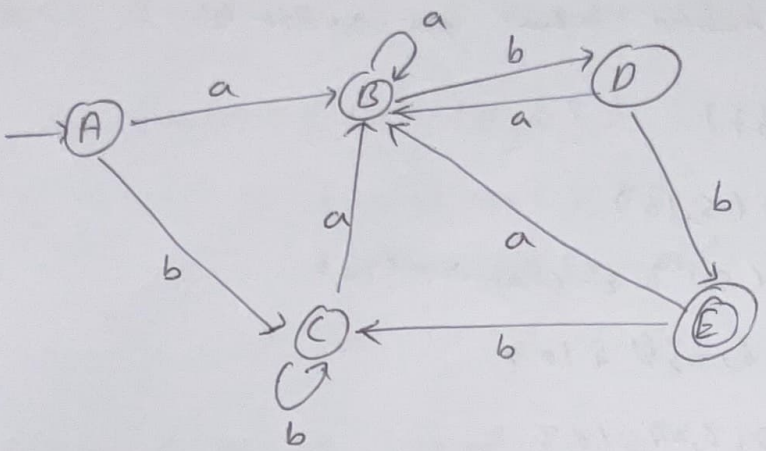
ε-closure (move {E, b})

$= \varepsilon$-closure (5)

$= C$

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

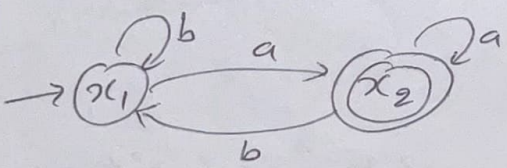
	a	b
→A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C



when we run min-state finite automata algorithm on it than we have 4 states

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

Q Find FA* for FA



$(a+b)^*a$

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

$\{\phi, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$

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

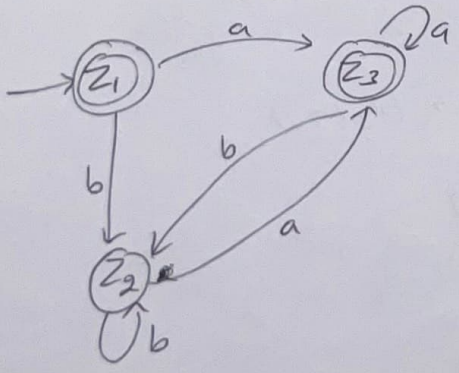
$\{\phi, \{x_1\}, \{x_1, x_2\}\}$

$\rightarrow (z_1) = \phi$ [Initial as well as final] → we see it later

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

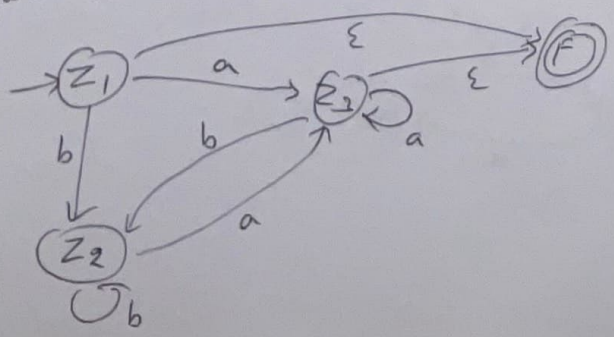
	ε	a	b
$z_2 = x_1$		$x_2/x_1 = z_3$	$x_1 = z_2$
$z_3 = [x_1, x_2]$		$x_1/x_2 =$ $x_2/x_1 =$ $= x_1/x_2 = z_3$	$x_1 = z_2$
$\rightarrow (z_1) = \phi$		$x_2/x_1 = z_3$	$x_1 = z_2$

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

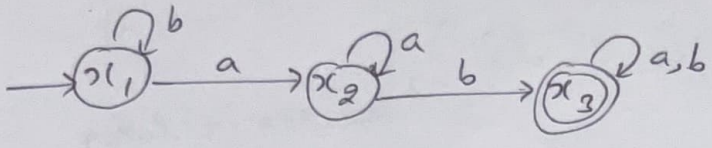


Ans

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



Q find FA* for FA



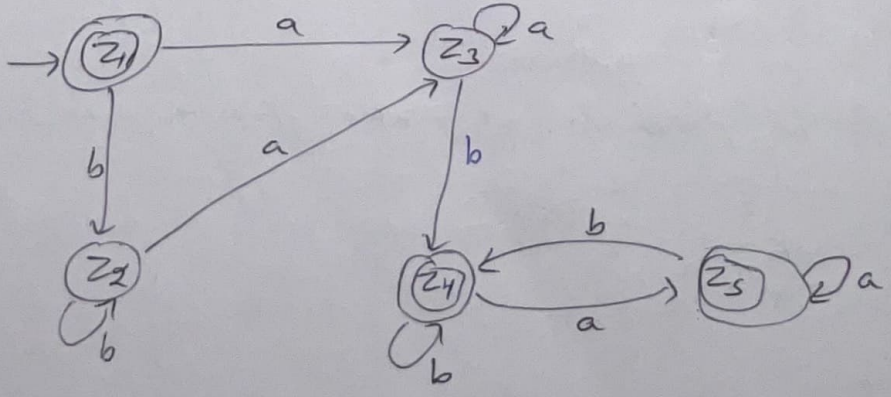
Possible subsets :- $\{\phi, \{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_2, x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$

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

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

transition table for FA*

Σ / δ	a	b
→ $z_1 = \phi$	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

1. 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

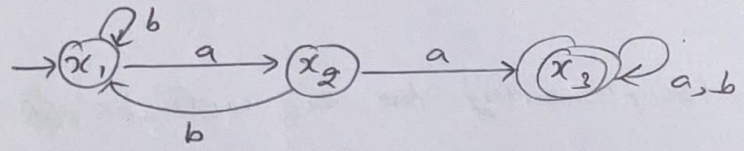
2. Complimentation property

Q find $L_1 \cap L_2$ two languages defined as follows over $\Sigma = \{a, b\}$

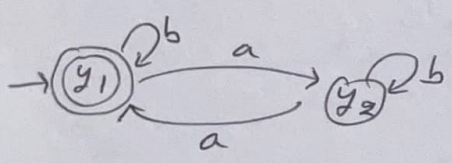
L_1 = all the string with aa

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

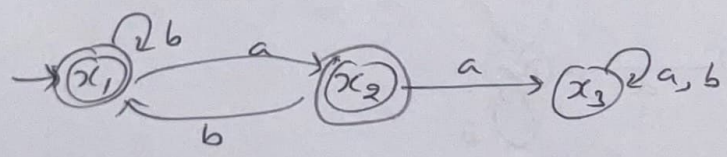
FA_1 :



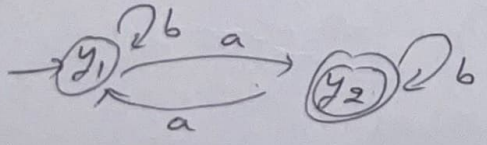
FA_2 :



FA_1 :



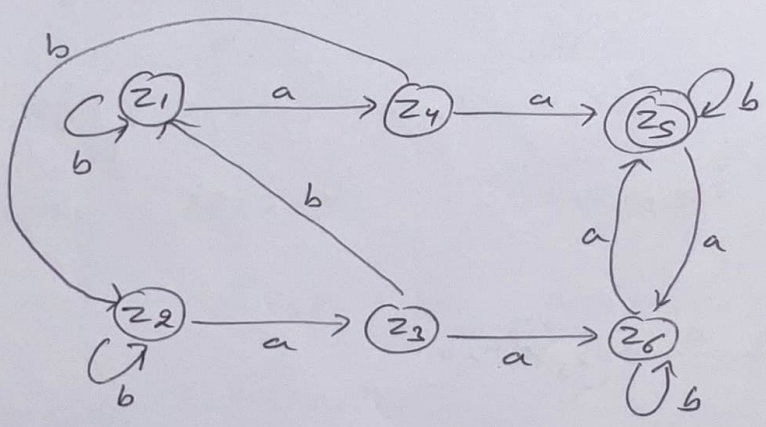
FA_2 :



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

Σ / Q	a	b
→ (z1) = x1/y1	x2/y2 (z4)	x1/y1 (z1)
(z2) = x1/y2	x2/y1 (z3)	x1/y2 (z2)
(z3) = x2/y1	x3/y2 (z6)	x1/y1 (z1)
(z4) = x2/y2	x3/y1 (z5)	x1/y2 (z2)
z5 = x3/y1	x3/y2 (z6)	x3/y1 (z5)
(z6) = x3/y2	x3/y1 (z5)	x3/y2 (z6)

now again take complementry so z5 will be final state

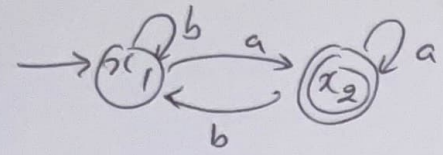


① L_1 $(a+b)^*a$

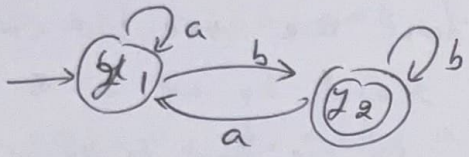
L_2 $(a+b)^*b$

find $L_1 \cap L_2$?

FA_1 :

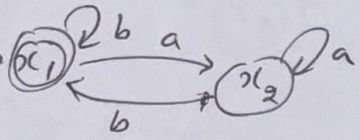


FA_2 :

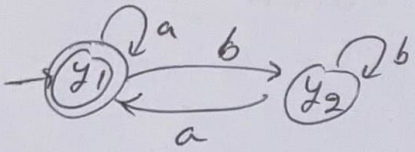


$FA_1 \cap FA_2 = (FA_1' \cup FA_2')'$

FA_1' :



FA_2'



Transition Table for FA_1'

δ	a	b
$\rightarrow x_1$	x_2	x_1
x_2	x_2	x_1

Transition Table for FA_2'

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

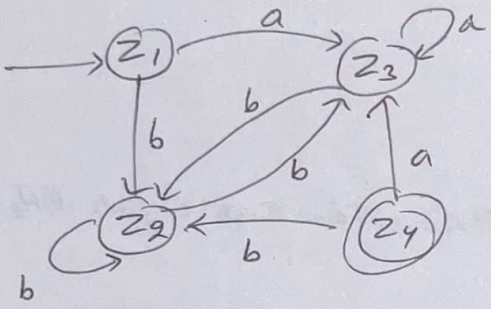
Transition Table for $FA_1' + FA_2'$

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

Transition Table for $(FA_1' + F_2')'$

	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

$L_1 \cap L_2 = \phi$

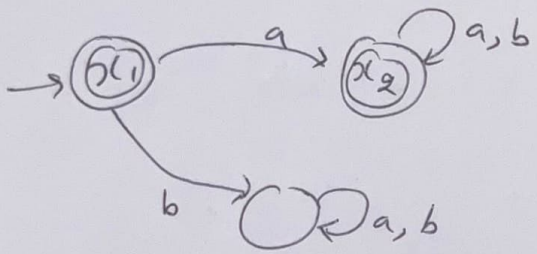
chapter 9 denial I.A cohine 1st Q 17 part one in exam

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

$(ab^*)^*$ chapter-9 14th Q

$= \{ a, ab, abb, abbb, aab ---- \}^*$

$= \{ \varepsilon, aab, a, ab, abb, abbb ---- \}$



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

Minimization to DFA ☆

z_2 +
z_4 +
z_5 +

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:

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

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:

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

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 + Λ)	(a + b)*aa(a + b)*
8.	(b + ab)*(a + Λ)	(b + ab*a)*ab*
9.	(b + ab)*(a + Λ)	(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	Strings with an even number of a's
	(ii) Even-length strings	Strings with an odd number of a's
19.	(i) Even-length strings	Strings with an odd number of a's and an odd number of b's
	(ii) Even-length strings	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 *not* under complement.

Pumping Lamma ✱ 5 marks Doubt

Theorm 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 $n = 1\ 2\ 3 \ldots$ are words in L

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

Let us assume a language:-

$L = \{ \varepsilon, ab, aabb, aaabbb \ldots \}$

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

Prove that $L = \{ a^n b^n \mid n \geq 0 \}$ 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 $\omega \in L$ and

$|z| > K$

which can be expressed as the concatination of 3 strings

x, y, z

$\omega = xyz$ ω is any string

then

xy^iz for $i = 1, 2, 3 \ldots$

must be in L

Let $\omega = a^4 b^4$ for $n = 4$

aaaabbbb is in L

Case I

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

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

Pump $|y| >= 0$ two times

But $aaaabbbb \notin L$

therefore our assumption was wrong so L is nonregular

$|y| > 0$

$|xy| \leq n$ $|w| > K$ ⟶ states

↓

length of string

Let $x = a^p, y = a^q, z = b^r$

$|x| = p$

$|y| = q$ } Length

$|z| = r$

If xy^iz must be in L

$i = 1$ $a^p (a^q)^i b^r$ must be in L

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

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

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

$q \geq 0$

→ Hence $a^p (a^q)^i b^r \notin L$

therefore our assumption was wrong L is non-regular

show that the language pallindrom is not regular

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

or

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

Assume this is regular language

$x\ y^i\ z$ → Pump it i times

$y \neq \varepsilon$

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

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

using pumping lamma

must write this before start Question

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

$w = xyz$

$x = a^p \ , y = a^q \ , z = b^r$

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

then $a^p (a^q)^i b^r$ must also in L

$a^p a^{qi} b^r$ must also in L

$a^{p+qi} b^r$ must also in L

for $i = 2$

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

Therefore $x y^i z$ is not in L

Hence our assumption is wrong the language is non-regular

Q a^p where p is prime is non-regular

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

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

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

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

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

$xy^{30}z$ 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 $y = a^m (m \neq 0)$ $|y| > 0$ — 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

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

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

$\omega = a^{n^2}$ 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

$|x| + |y| + |y| + |z| > n^2$ since $|y| > 0$ ——— (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 (Contex free language) represented by CFG (Context free grammer)

Lexical Analysis

int identifier = 20;

$[a-z, A-z, -][0-9, a-z, A-z, -]^*$

Lexical Analysis

↓

Lexical Analyzer

Non terminating

Production Rules

① $NT = \{S, A, B\}$

② $T = \{a, b\}$

③ $S \rightarrow AB$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

Starting $NT = S$

Combination of all four things is called grammer

$S \Rightarrow AB \Rightarrow$

↓

generates to

OR

Derivates to

$S \Rightarrow aAB$

$S \Rightarrow aaAB$

$S \overset{*}{\Rightarrow} aaad \text{ --- } \varepsilon B$

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

$S \overset{*}{\Rightarrow} a^+ b^*$

CFG → RL, Somepart of NRL, CFL, PDA

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

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

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

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: $S \to aS$

P2: $S \to \varepsilon$

Production Rules

$S \to aS/\varepsilon$

$S \Rightarrow$ (generates to) ε belong to CFL

$S \Rightarrow aS$

$\Rightarrow aaS$ (Put S to aS)

$\vdots$

$\Rightarrow a^n S$ ($S \to \varepsilon$)

$\Rightarrow a^n \varepsilon$ CFL

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

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

Q find the CFL for the give CFG

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

Production Rules

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

$S \Rightarrow a$

$a \in CFL$ —— ①

$S \Rightarrow b$

$b \in CFL$ —— ②

$S \Rightarrow aS$

$\Rightarrow aaS$

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

$\Rightarrow a^n$

$a^n \in CFL$

$S \Rightarrow bS$

$\Rightarrow bbS$

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

$\Rightarrow b^n$

$b^n \in CFL$

$S \Rightarrow aS$

$\Rightarrow abS$

$\Rightarrow abaS$

$\Rightarrow ababS$

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

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

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

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

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

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

$V_T = \{a, b\}$

$S \rightarrow XaaX$

$X = aX \mid bX \mid \varepsilon$

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

$S \Rightarrow XaaX$

$\Rightarrow aaX$

$\Rightarrow aa$

$aa \in CFL$ —(1)

here we have 2 non-Terminal

there are 2 types of derivation leftmost and rightmost

In leftmost we start from left and move to right

In Rightmost we start from right and move to left

At one time we change one non-termial

find one

Q Find CFL from given CFG

$V_N = \{S, X, Y\}$ $V_T = \{a, b\}$

$S \to XY$

$X = aX \mid bX \mid a$

$Y = Ya \mid Yb \mid a$

$X \Rightarrow aX$ $X \Rightarrow bX$

$\Rightarrow abX$ $\Rightarrow baX$

$(a+b)^* a$

$Y \Rightarrow Ya$ $Y \Rightarrow Yb$

$\Rightarrow Yba$ $\Rightarrow Yab$

$a(a+b)^*$

$S \Rightarrow XY$

$(a+b)^* a\, a\, (a+b)^*$ Ans

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

Q find CFL from given CFG

$V_{NT} = \{S, B, U\}$

$V_T = \{a, b\}$

$S \to SS \mid BS \mid SB \mid \varepsilon \mid USU$

$B \to aa \mid bb$

$U \to ab \mid ba$

B = Balanced

U = unBalanced

$S \Rightarrow BS$ $S \Rightarrow SB$

$\Rightarrow aaS$ $\Rightarrow$

$\Rightarrow aa$

$aa \in CFL$

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

Q find CFL from given CFG

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$

$S \Rightarrow aSa$
$\Rightarrow aaSaa$
$\Rightarrow aabaa$
$\Rightarrow aaaaa$

$S \Rightarrow aSa$
$\Rightarrow abSba$
$\Rightarrow abbSbba$
$\Rightarrow abbabba$
$\Rightarrow abbbbba$

this is palendrom

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

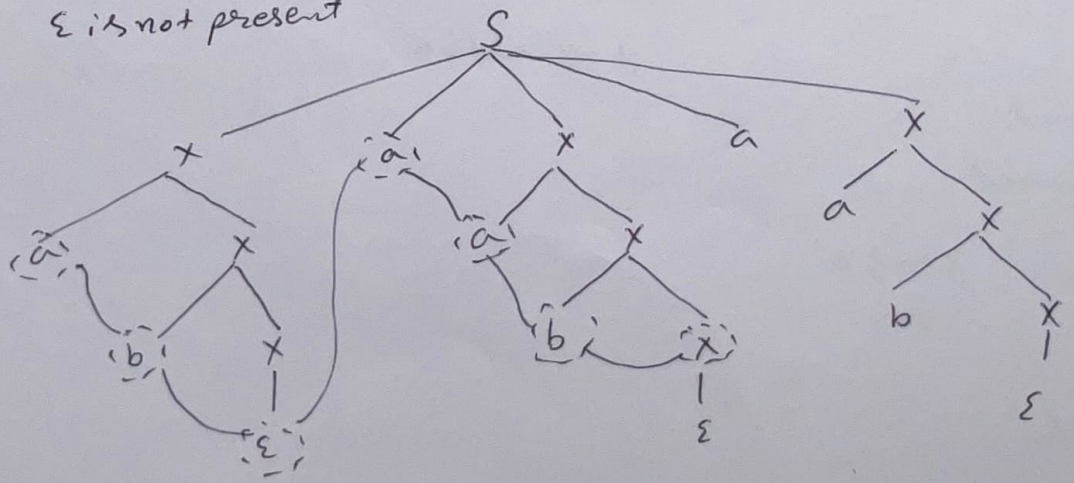
$a^n b a^n , a^m a a^n , b^n a b^n , b^n b b^n$

Q Consider the CFG

$S \rightarrow XaXaX$
$X \rightarrow aX \mid bX \mid \varepsilon$

what is the language this CFG generates? find CFL

min string length → 2
ε is not present



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

Q Consider the CFG

$S \to aX$

$X \to aX / bX / \varepsilon$

what is the language this CFG generates? find CFL

$a(a+b)^*$ Ans

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

(i) ab^*

$S \to aX$

$X \to bX / \varepsilon$

$\{a, ab, abb, abbb ------ \}$

Q Find CFG's $\Sigma = \{a, b\}$;

All words that have exactly 2 or 3 b's

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

$S \to XbXbX / XbXbXbX$

$X \to aX / \varepsilon$

give all
Terminal
Non-Terminal
starting nonterminal

d Find CFG

(i) All words that have diff first and last letter

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

$S \rightarrow aXb \mid bXa$

$X \rightarrow aX \mid bX \mid \varepsilon$

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

~~$S \rightarrow XX \mid abb \mid baaX$~~

$S \rightarrow baaS \mid abbS \mid \varepsilon$

(iii) All words that do not have substring ab

b^*a^*

$S \rightarrow XY$

$X \rightarrow bX \mid \varepsilon$

$Y \rightarrow aY \mid \varepsilon$

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 → capital letter

Terminal → small letter

③ Eliminate unit productions

left one nonterminal — Right one nonterminal

unit production

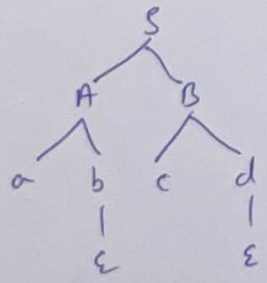
$S \xRightarrow{*} xAy \xRightarrow{*} w$

S → useful Symbal

w → terminal string

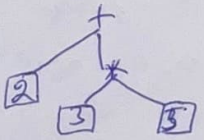
S that generate terminal string in some steps so that is useful symbal

Parse Tree



Internal nodes are non-terminal

Syntax Tree



Internal nodes are Symbols

① we say X is generating if $X \overset{*}{\Rightarrow} w$ for some terminal string w

And

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

Q
$S \to AB$
$A \to aA/\varepsilon$
$B \to bB/\varepsilon$
$C = cC/\varepsilon$

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

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

$S \to A$

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

B → not reachable from S
C → not generating
so we have
$S \to aS/A$
$A \to a$

Q
$S \to aAa/aBC$
$A \to aS/bD$
$B \to aBa/b$
$C \to abb/DD$
$D \to aDa$
$E \to Fb$
$F \to c$

E & F are generating but they are not reachable
D is not generating because it
so we have
$S \to aAa/aBC$
$A \to aS$
$B \to aBa/b$
$C \to abb$

If $G = (V_{NT}, V_T, S, P)$ be a CFG and $L(G) \neq \phi$ then $G_1 = (V'_{NT}, V_T, S, P')$ (Terminal ↓ It change)

by removing useless symbols/productions will lead to a condition $L(G) = L(G')$

↓

Language generating by grammer G

Remove Nullable production

$A \to \varepsilon$ → It goes directly to ε

$A \overset{*}{\Rightarrow} \varepsilon$ → after some generate it goes to ε so it is nullable

ε → directly nullable
ε nullable → indirectly nullable

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

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

$L(G) \neq L(G')$ $\quad L(G') = L(G) - \{\varepsilon\}$

$S \to AB \to \not{\varepsilon} | B | A | AB$

$A \to Aa/\varepsilon \to a | Aa$

$B \to Bb/\varepsilon \quad b | Bb$

$S \to AB$ here $A \to \varepsilon$

$S \to \varepsilon B \quad B \to \varepsilon$

$S \to \varepsilon\varepsilon \quad S \overset{*}{\Rightarrow} \varepsilon$

$\varepsilon \in L(G)$

AB	
0 0	∈
0 1	B
1 0	A
1 1	AB

Aa	
0	a
1	AA

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 $S \to \varepsilon$

$S \to AB$
$A \to aAA/\varepsilon$
$B \to bBB/\varepsilon$

Since
$A \to \varepsilon$
$B \to \varepsilon$
} null productions

so $S \to AB/A/B/\varepsilon$
$A \to a/aA/aAA$
$B \to b/bB/bBB$
} $L(G') = L(G) - \{\varepsilon\}$

How we write aAA and bBB

Q Eliminate NULL productions from the following CFG

S → ABaC
A → BC
B → b/ε
C → D/ε
D → d

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

Sol:-

B → ε
C → ε } are ε - productions

A *⇒ ε is a nullable production

S → ABaC / ABa / AaC / Aa / BaC / Ba / aC / a
A → BC / B / C
B → b
C → D
D → d

ABC	
000	⇒ a
001	⇒ aC
010	⇒ Ba
011	⇒ BaC
100	⇒ Aa
101	⇒ AaC
110	⇒ ABa
111	⇒ ABaC

it terminates here that why we not all ε here

BC	
00	→ ε X
01	→ C
10	→ B
11	→ BC

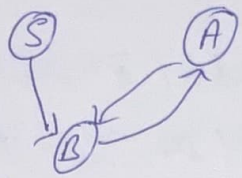
Eliminate unit production

Q S → Aa / B
B → A / bb
A → a / bc / B

The CFG without unit productions can be written as

S → Aa
B → bb
A → a / bc

Plus the new production rules



$A \to bb$ ⟶ Replace B with B production rule

$B \to a/bc$ → Replace A with A production

$S \to a/bc/bb$ → $S \to B \to A$ so $S \to A$ Replace B with S production rules

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

$B \to a/bc/bb$ // not reachable from S → useless symbol

$A \to a/bc/bb$

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

$F \to I/(E)$. () → Terminal

$T \to F/T*F$

$E \to T/E+T$

Soln Here starting non terminal is E

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

$V_T = \{a, b, 0, 1, (,), *, +\}$

$S = E$

The CFG without unit productions can be written as

$E \to E+T$

$T \to T*F$

$F \to (E)$.

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

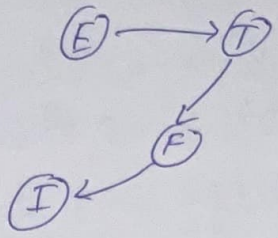
- Plus the new production rules

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

$F \rightarrow a/b/Ia/Ib/Io/I1/(E)$

$T \rightarrow a/b/Ia/Ib/Io/I1/(E)/T*F$

$E \rightarrow a/b/Ia/Ib/Io/I1/(E)/T*F/E+T$



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

CNF

- Eliminate the NULL production

Step1:-

Step2:- Eliminate any unit production

Step3:- Eliminate all the useless symbol in the grammer

Step4:- Convert the resulting grammer CNF (chomsky normal form)

$S \rightarrow OAO/1B1/BB$

$A \rightarrow C$

$B \rightarrow S/A$

$C \rightarrow S/E$

(AB) Bade → non-terminal note

(a) Chote → terminal note

Chomsky Normal Form (CNF)

Every CFL (without null) is generating by CFL 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 CFG has no useless symbol such a grammer 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 bodys of 3 or more into a cascade of production, each with a body containg 2 variables

step3:- for every terminal a that appers in a body of length 2 or more create a variable, say (A). This variable has only one production $A \to a$

step2:- for step2 if the production $A \to B_1 B_2 \cdots B_K$ $K \geq 3$ then introduce new variable $(K-2)$ no of new variables

$A \to B_1 C_1$

$C_1 \to B_2 C_2$

$C_2 \to B_3 C_3$

$\vdots$

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

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

we only want these type of production

$P \to AB$

or

$P \to a$

Q Convert the following grammer into CNF

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

Soln a $S \to a$, $S \to b$ } already in chomsky normal form

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

$\{a \to A, b \to B\}$ -> we change this

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

$A \to a$

$B \to b$

Now

$S \to AA$, $S \to BB$ } in chomsky normal form

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

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

If length of body is n so time complexity

$A \rightarrow a$
$B \rightarrow b$
$X \rightarrow SA$
$Y \rightarrow SB$

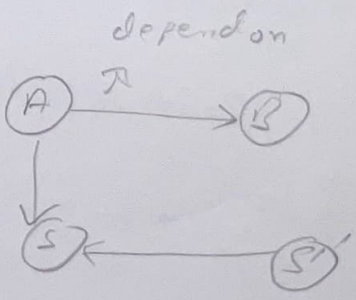
Example CFG to CNF Simplification of CFG

$S \rightarrow ASA / aB$
$A \rightarrow B / S$
$B \rightarrow b / \varepsilon$

$S' \rightarrow S$ 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 :-

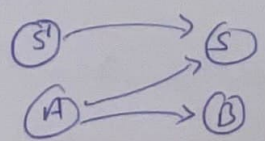
$S' \rightarrow S$
$S \rightarrow S$
$A \rightarrow B$
$A \rightarrow S$

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

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

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

$A \rightarrow ASA/a/aB/AS/SA/b$

$B \rightarrow b$

Step3:- find the production that has more then 2 Non-Terminalin

RHS

$S' \rightarrow ASA$

$S \rightarrow ASA$

$A \rightarrow ASA$

$S' \rightarrow AX/a/aB/AS/SA$

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

$A \rightarrow AX/a/aB/AS/SA/b$

$B \rightarrow b$

$X \rightarrow SA$.

Step4:- 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)

$E \rightarrow E+T \mid T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$T \rightarrow T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Soln

$P \rightarrow +$

$M \rightarrow *$

$L \rightarrow ($

$R \rightarrow)$

$A \rightarrow a$

$B \rightarrow b$

$Z \rightarrow 0$

$N \rightarrow 1$

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

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

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

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

we have to convert

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

$F \rightarrow LER$

$E \rightarrow EP_1 \mid TM_1 \mid LE_1 \mid a \mid b \mid IA \mid IB \mid IZ \mid IN$

$T \rightarrow TM_1 \mid LE_1 \mid a \mid b \mid IA \mid IB \mid IZ \mid IN$

$F \rightarrow LE_1 \mid a \mid b \mid IA \mid IB \mid IZ \mid IN$

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

$P_1 \rightarrow PT$

$M_1 \rightarrow MF$

$E_1 \rightarrow ER$

$P \rightarrow +$

$M \rightarrow *$

$L \rightarrow)$

$A \rightarrow a$

$B \rightarrow b$

$Z \rightarrow 0$

$N \rightarrow 1$

Example: Consider the CFG

$S \rightarrow aaaaS / aaaa$ which generates

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

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

Convert the above CFG to CNF

① $S \rightarrow aaaaS / aaaa$

② $S \rightarrow AAAAS / AAAA$

$A \rightarrow a$

$S \rightarrow AA_1 / AAAA$

$A_1 \rightarrow AAAS$

$S \rightarrow AA_1 / AAAA$

$A_1 \rightarrow AA_2$

$A_2 \rightarrow AS$

$S \rightarrow AA_1 / AA_3$

$A_1 \rightarrow AA_2$

$A_2 \rightarrow AS$

$A_3 \rightarrow AAA$

$A_4 \rightarrow AA$

Example 3:

$S \rightarrow Na$

$N \rightarrow a / b$

$S \rightarrow NA$
$N \rightarrow a / b$
$A \rightarrow a$
} valid

$S \rightarrow NN$
$N \rightarrow a / b$
Note valid

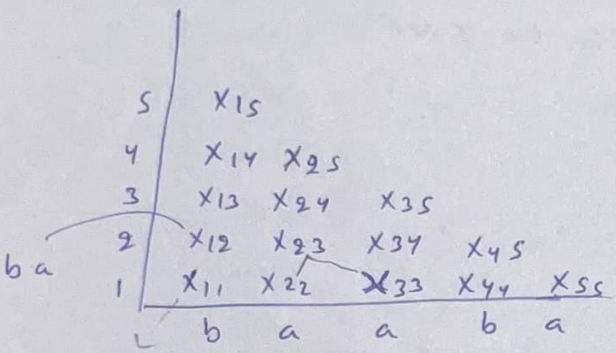
Find whether the string exist in CFG

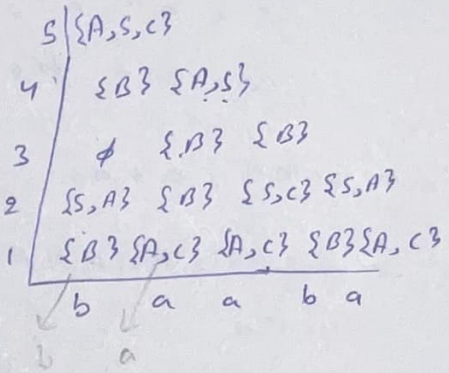
start from 1

String :- b a a b a
1 2 3 4 5

CFG = S → AB/BC
A → BA/a
B → CC/b
C → AB/a

CYK Algorithm





$X_{12} = \{B\} \cdot \{A, C\}$

$= \{BA\} \cup \{BC\} \Rightarrow \{S, A\}$

$X_{23} = aa$

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

$\{\{AA\} \cup \{AC\} \cup \{CA\} \cup \{CC\}\}$

$= \{B\}$

$X_{34} = ab$

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

$\{\{A.B\} \cup \{CB\}\}$

$\{S, C\}$

$X_{24} = aab$

a/ab

aa/b

$a = \{A, C\}$

$ab = \{S, C\}$

$aa = \{B\}$

$b = \{B\}$

$\{A, C\}\ \{S, C\} = \{AS, AC, CS, CC\}$

$\{B\}\ \{B\} = BB$ ↓ B

$x_{24} = B$

$x_{45} = ba$

$\{S, A\}$

$x_{13} = baa$

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

$ba/a \rightarrow \{S, A\} \cdot \{A, C\} = \{SA\} \cup \{SC\} \cup \{AA\} \cup \{A, C\} \cup \{AC\} \cup \{B\}$

$\rightarrow \phi$

$x_{35} = aba = a/ba$ (→ devide) $= \{A, C\}\ \{S, A\}$

$= AS, AA, CS, CA$

ab/a (→ devide) $= \{S, C\}\ \{A, C\}$

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

$x_{14} = baab$

$baa/b = \{\phi\} \cdot \{B\} = \{B\}$

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

$b/aab = \{B\}\ \{B\} = \{B\}$

$= \{B\}$

x_{25} = aaba

a/aba = {A, C} {B} = {AB}, {CB} = S

aa/ba = {B} {S, A} = BS, BA = A

aab/a = {B} . {A, C} = BA, BC = A, S

{A, S}

x_{15} = baaba

b/aaba = {B} ∪ {A, S}

{BA} ∪ {BS}

{A}

ba/aba = {S, A} ∪ {B}

{SA} ∪ {AB}

{S, C}

baa/ba = ϕ ∪ {S, A}

{S, A}

baab/a = {B} ∪ {A, C}

{BA} ∪ {BC}

{A, S}

= {A, S, C}

string is exist

PDA

PDA is used to define the contex free grammer

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

gama → those character go to stack are part of gama

PDA is made up of 7 tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

↳ always on the stack if stack is empty so have to put Z_0 to there

Q → finite set of states (like FA)

Σ → input symbol (like FA)

Γ → stack alphabate (Symbol push into stack)

δ → Transition function

$$\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow Q \times \Gamma^*$$

Deterministic PDA

q_0 → initial state

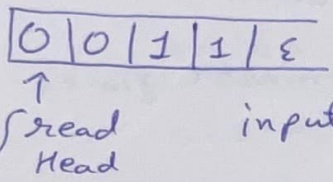
Z_0 → stack start symbol it always on stack

F → set of final states

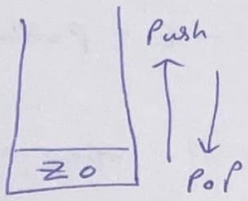
δ → for No deterministic PDA

$$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \Gamma \rightarrow 2^Q \times \Gamma^*$$

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



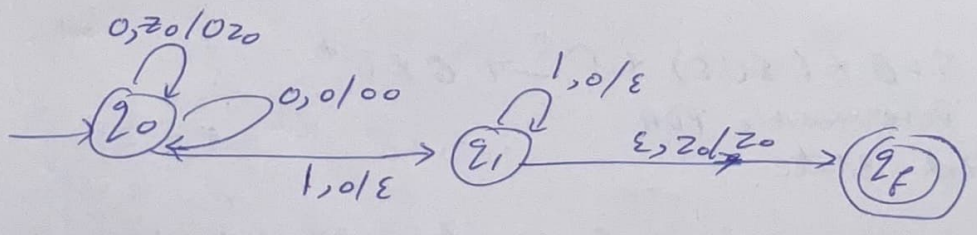
Finite Control (it decide our stack grow or shrink)



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

Notes:- Too version of PDA exist 1st that entring on accepting accept by state (like FA do)

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

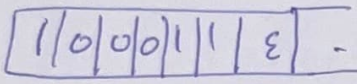


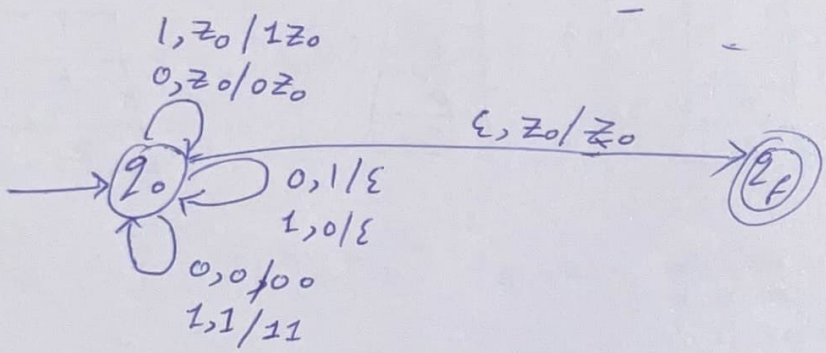
in initial state firstly we have to pop from the stack than again push that on stack than push read Head value when we are not in initial state so pop from the stack and we have 1

Q $L = \{ w \in \{0,1\}^* \mid n_0(w) = n_1(w) \}$

$|w| > 1$

$= \{ 01, 10, 0011, 0101, 1010, 1100, 0110 \ldots \}$





Linear way to represent

$\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$

$\delta(q_0, 1, Z_0) = (q_0, 1Z_0)$

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

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

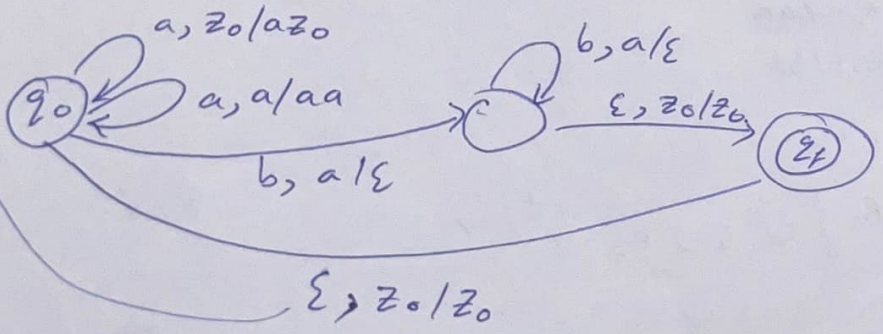
$\delta(q_0, 0, 0) = (q_0, 00)$

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

$\delta(q_0, \varepsilon, Z_0) = (q_f, Z_0)$

If we want to accept NULL string

Q $\{ a^n b^n \mid n \geq 0 \}$



$\delta(q_0, \varepsilon, Z_0) = (q_1, Z_0)$

↳ Accept ε_0

$\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon)$

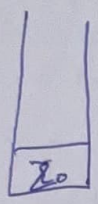
↳ Accepted by empty stack (NULL store)

Q $\{ w c w^R \mid w \in \{a,b\}^+ \}$

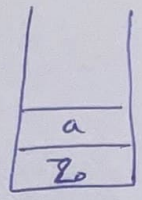
→ Guiding principle
split

a	b	b	c	b	b	a	ε

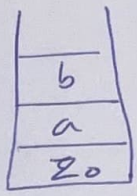
step1



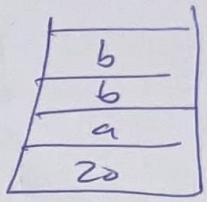
step2

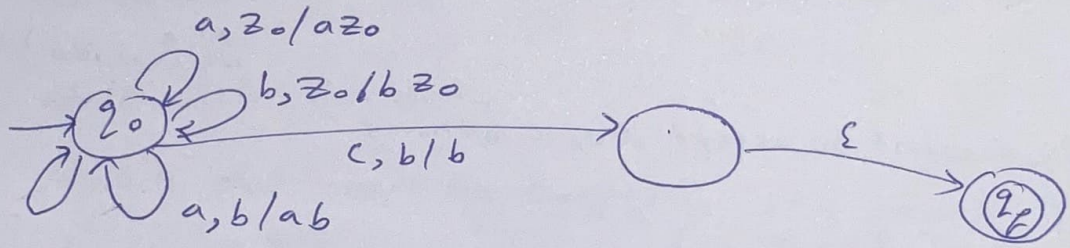


step3



step4



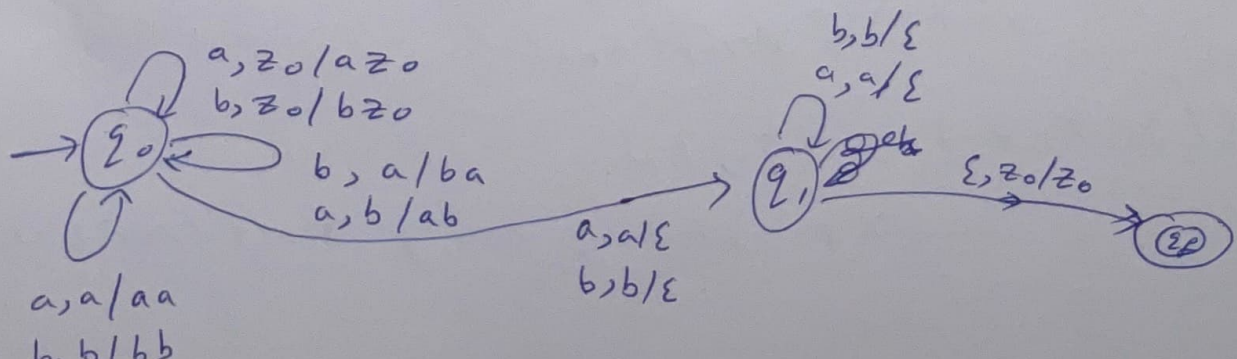


Q $L = \{ \omega \omega^R \mid \omega \in \{a,b\}^+ \}$

a b a a b a

a b b b b a

b a a a a b



$(q_0, abba, z_0) \Longrightarrow (q_0, a z_0)$

$(q_0, bba, a) \Rightarrow (q_0, ba z_0)$

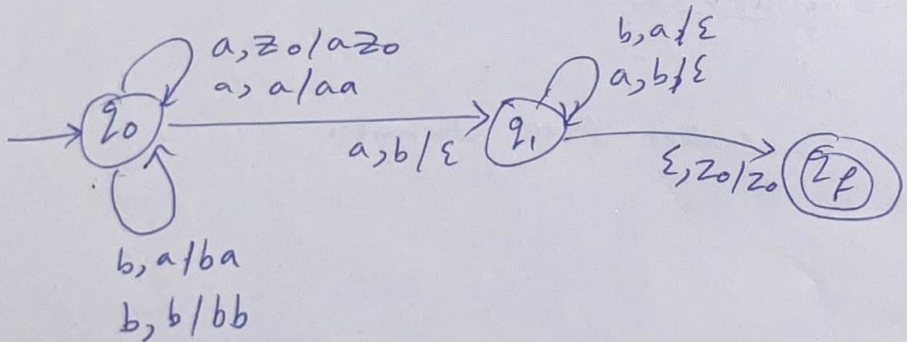
$(q_0, ba, b) \Rightarrow$

$(q_0, bba z_0)$ $\quad$ $(q_1, a z_0)$

$(q_0, a, b) = (q_0, abba z_0)$ $\quad$ $(q_1, a, a) \Rightarrow (q_1, z_0)$

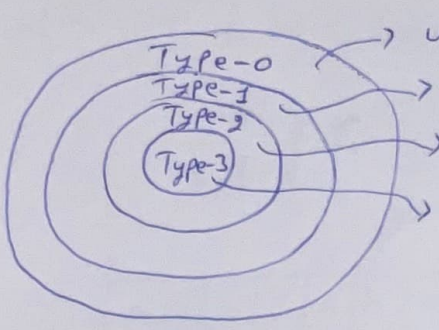
Q $L = \{ a^n b^m a^m b^n$, where ~~n, m~~ and $n, m \geq 1 \}$

m, n are independent



abbaab

Q $\{0^n 1^{2n}$, where n should be greater than 0 $\}$



→ unrestricted grammer (TM)

→ context-sensitive grammer (LBA)

→ context-free grammer (PDA)

→ Regular Grammer (FA)

1. $\alpha \in (V_{NT} \cup V_T)^*$ → unristricted grammer

$\alpha \to \beta$

2. $|\alpha| \leq |\beta|$ → context-sensitive grammer

$\alpha \to \beta$

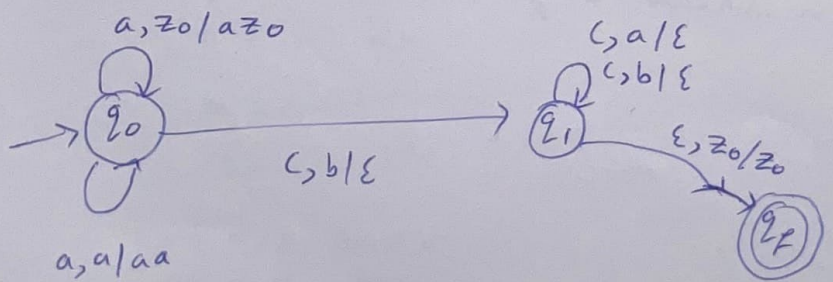
eg:- $\underline{a}\underline{B}\underline{b} \to \underline{a}\,\underline{aBBA}\,\underline{b}$

3. $|\alpha| \leq |\beta|$ → context-free grammer

$\alpha \to \beta$

$A \to \beta B$

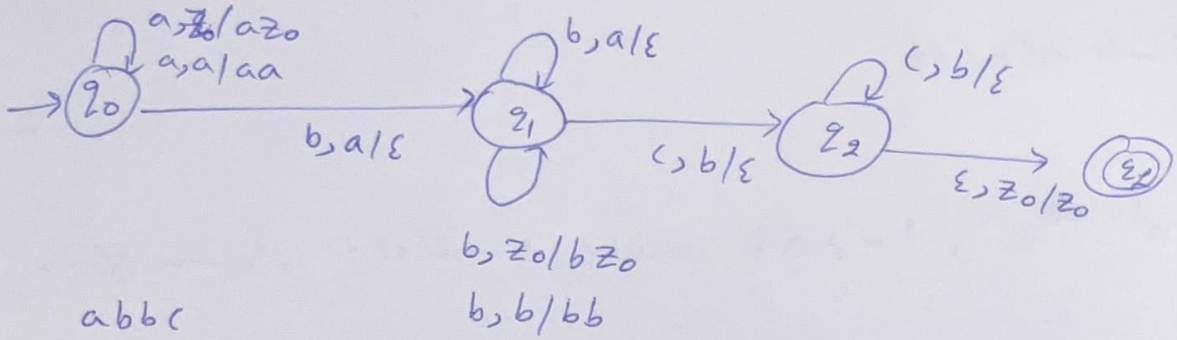
Q $a^n b^m c^{(n+m)}$, $n, m \geq 1$ make PDA of it



a, a / aa
b, a / ba
b, b / bb

abcc
aabbcccc
aabccc
⋮

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



abbc
aabbbc
aabbbbbccc

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 gramer of it

$S \longrightarrow aSb / \varepsilon$

Q $\{ a^m b^n / n \geq 1 \}$ make smallest grammer of it

$S \rightarrow aSb / ab$

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

$S \rightarrow aSb/bb$

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

$S \rightarrow aasb \mid \varepsilon$

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

$S \rightarrow aasb/aaa$

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

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

$S \rightarrow AS_1$

$S_1 \rightarrow aS_1 b \mid \varepsilon$

$A \rightarrow aA/a$

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

$S \rightarrow AS_1$

$S_1 \rightarrow aS_1 b \mid \varepsilon$

$A \rightarrow aA \mid \varepsilon$

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

PDA (Pushdown Automata)

RG → RL → FA ⇄ NFA / DFA

1) FA used to Recognize regular language
2) FA can't Recognize more Powerfull language than regular language.

Like $a^n b^n$ / $n \geq 1$

It can't accepted by FA because memory not sufficient

CFG → CFL → PDA → DPDA (There is only one move in every situation)

→ ND PDA (There could be multiple moves under a situation)

ND PDA is more powerful than DPDA

Every CFL is can't be recoganize by DPDA
Every CFL is can be recoganize by ND PDA

Palindrom can be accepted by ND PDA

FA + Stack = PDA

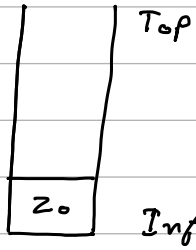
$a^n b^n$ $n \geq 1$ accepted by PDA because of stack

Why stack?

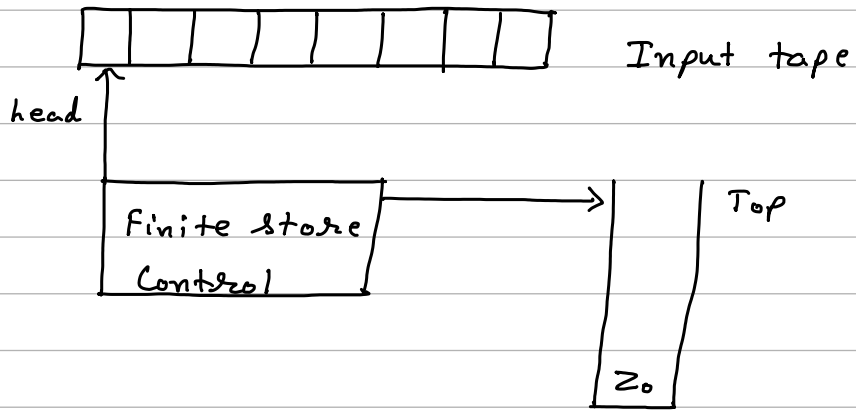
1) It is a zero address DS push & pop operation are performed on Top

2) Infinite size, so no overflow

3) Z_0 is initial stack symbol so no underflow

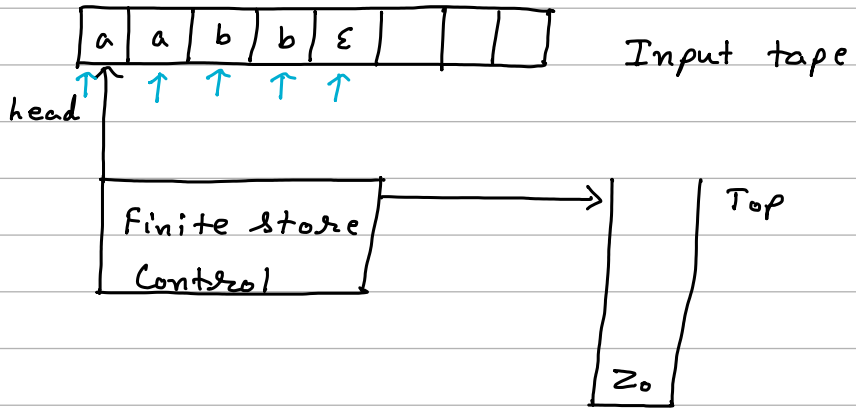


Infinite size stack

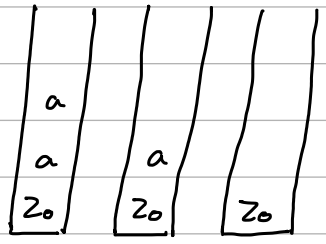


FA + stack = PDA

Q $a^m b^n$ $n \geq 1$



FA + stack = PDA



It is empty at end of tape so language is accepted

formal Defination of PDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q = set of states

Σ = Input alphabets

Γ = (upper case gama (to)) Stack symbols

eg:- z_0, a, b ------

$\delta = Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to $Q \times \Gamma^*$

- δ → transition fun
- Q → current state
- $(\Sigma \cup \varepsilon)$ → Input alphabat
- Γ → top most stack symbol
- Q → next state
- Γ^* → operation on stack
  - push
  - pop
  - no operation

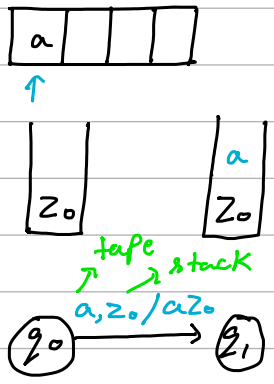
$q_0 = q_0 \in Q$ is the initial state

F = final state

z_0 = Inital stack symbol

Operation on PDA

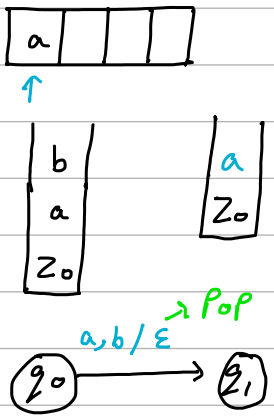
Push



$\delta(q_0, a, z_0) = (q_1, aZ_0)$

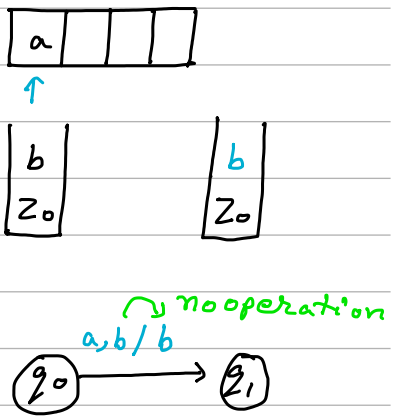
state, on tape, on stack

Pop



$\delta(q_0, a, b) = (q_1, \varepsilon)$

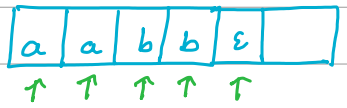
Skip / No-operation

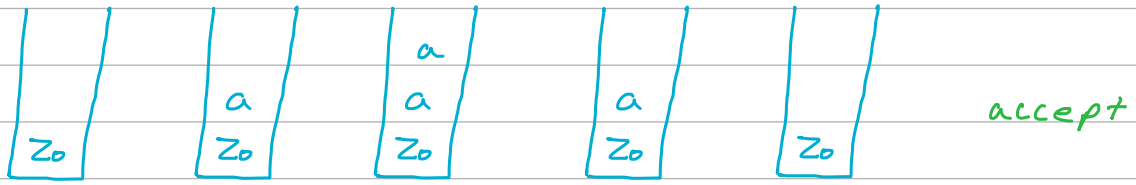


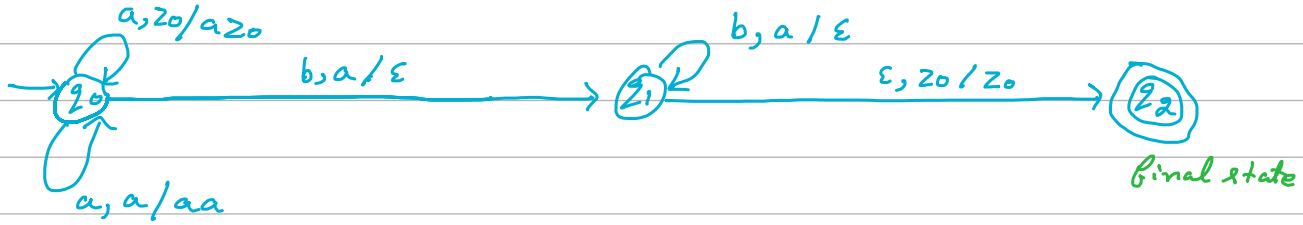
$\delta(q_0, a, b) = (q_1, b)$

① $a^n b^n$, $n \geq 1$

Consider $n = 2$







$\delta(q_0, a, Z_0) = (q_1, aZ_0)$ push

$\delta(q_0, a, a) = (q_1, aa)$ push

$\delta(q_0, b, a) = (q_1, \varepsilon)$ pop

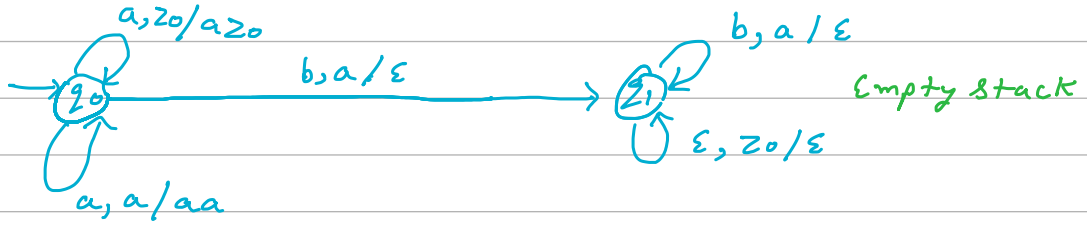
$\delta(q_1, b, a) = (q_1, \varepsilon)$ pop

$\delta(q_1, \varepsilon, Z_0) = (q_2, Z_0)$ skip / No-operation

↳ final state acceptance

This is DPDA

DPDA — final state
— empty stack



$\delta(q_0, a, z_0) = (q_1, az_0)$ push

$\delta(q_0, a, a) = (q_1, aa)$ push

$\delta(q_0, b, a) = (q_1, \varepsilon)$ pop

$\delta(q_1, b, a) = (q_1, \varepsilon)$ pop

$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$ pop

↓

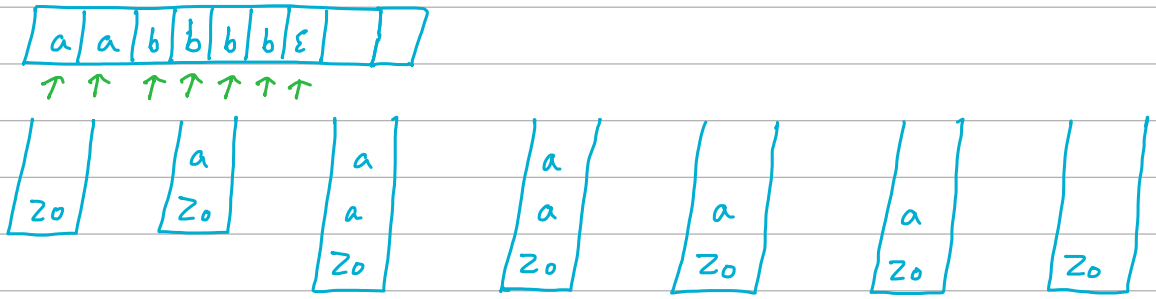
Empty stack acceptance

Q Design PDA for CFL $a^n b^{2n}$, $n \geq 1$

$n = 1$ abb

$n = 2$ aabbbb

$n = 3$ aaabbbbbb



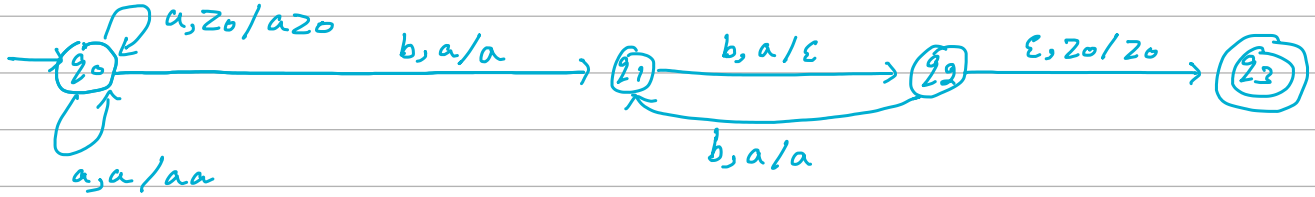
push a push a skip b pop a skip b pop a

a on tape z_0 on stack → push

a on tape a on stack → push

got one b → skip

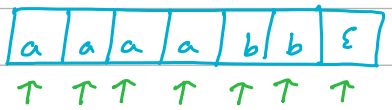
got two b → pop

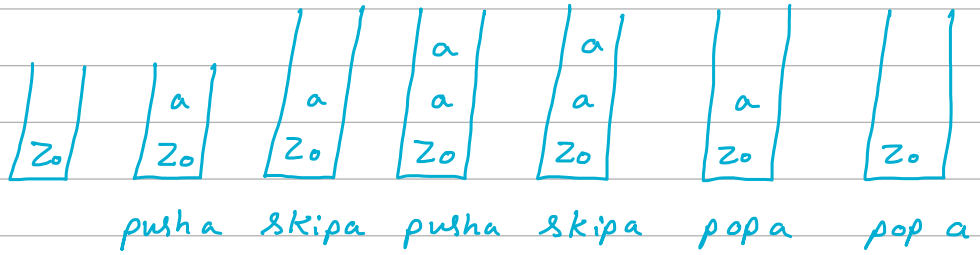


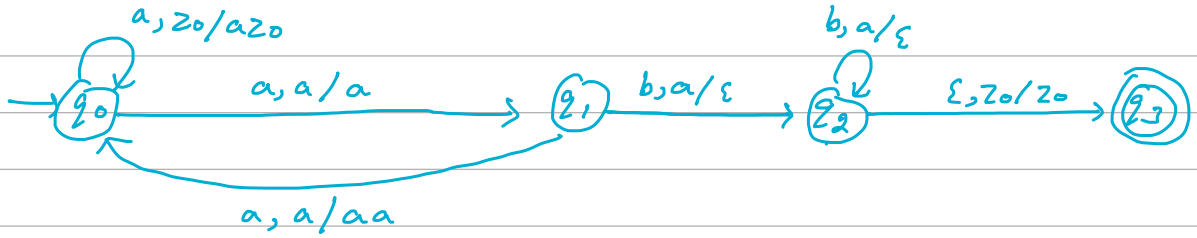
$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ push
$\delta(q_0, a, a) = (q_0, aa)$ push
$\delta(q_0, b, a) = (q_1, a)$ skip / No operation
$\delta(q_1, b, a) = (q_2, \varepsilon)$ pop
$\delta(q_2, b, a) = (q_1, a)$ skip
$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$ skip

Q Design PDA for CFL $a^{2n}b^n$, $n \geq 1$

$n=1$ aab
$n=2$ aaaabb
$n=3$ aaaaaabbb







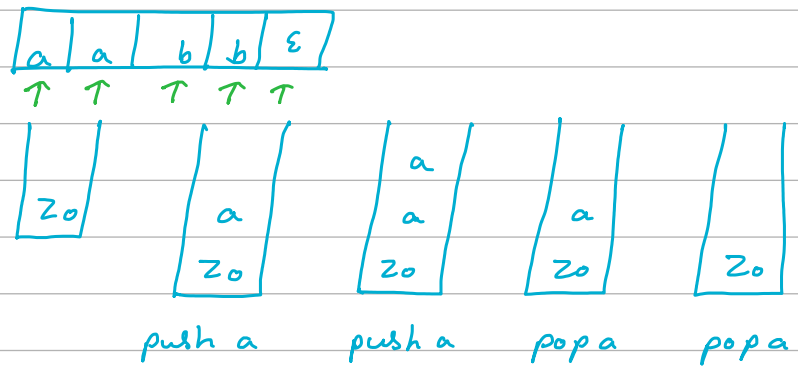
$\delta(q_0, a, Z_0) = \delta(q_0, aZ_0)$ push a
$\delta(q_0, a, a) = \delta(q_1, a)$ skip
$\delta(q_1, a, a) = \delta(q_0, aa)$ push a
$\delta(q_1, b, a) = \delta(q_2, \varepsilon)$ pop a
$\delta(q_2, \varepsilon, Z_0) = \delta(q_3, Z_0)$ skip

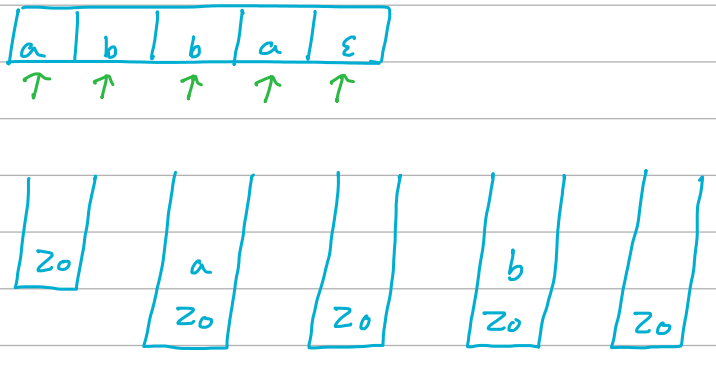
Q Design PDA for CFL

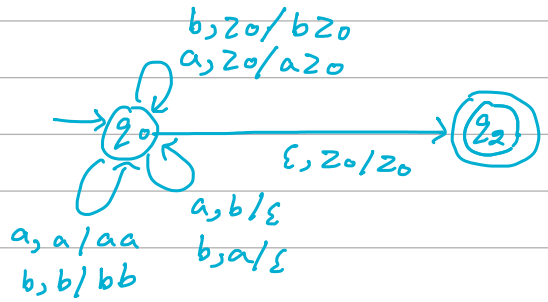
$L = \{ w \in (a,b)^* \mid n_a(w) = n_b(w) \}$

Equal no of a's and b's

ab, ba, aabb, bbaa, abab, baba, abba, baab, ----







$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ push

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$ push

$\delta(q_0, a, a) = (q_0, aa)$ push

$\delta(q_0, b, b) = (q_0, bb)$ push

$\delta(q_0, a, b) = (q_0, \varepsilon)$ pop

$\delta(q_0, b, a) = (q_0, \varepsilon)$ pop

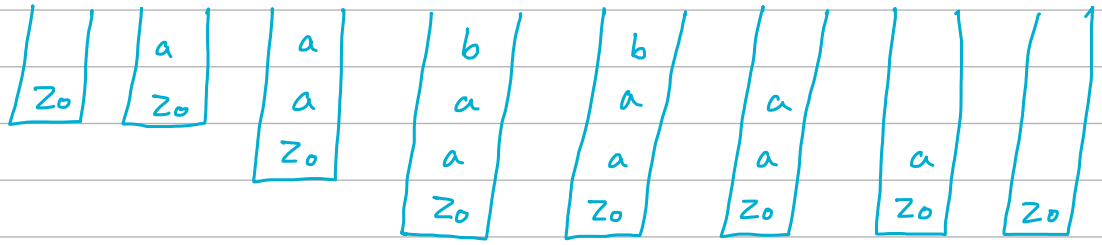
$\delta(q_0, \varepsilon, z_0) = (q_1, z_0)$ skip

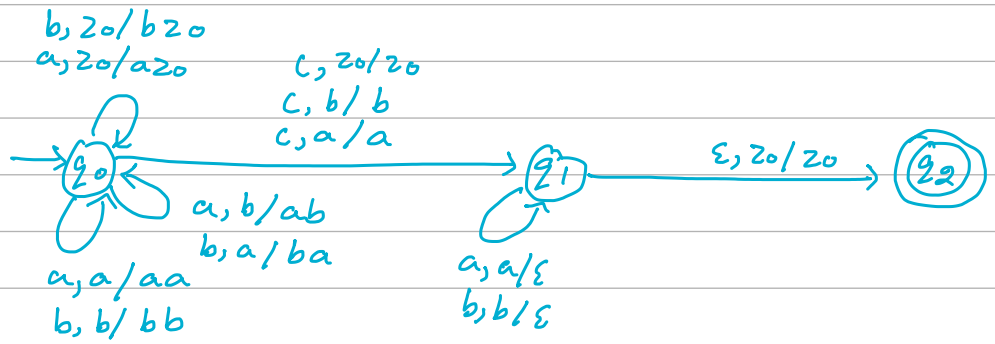
① $L = \{ w c w^R \mid w \in (a,b)^* \}$

$w = aab$

$w^R = baa$

aabcbaa





$\delta(q_0, a, z_0) = (q_0, a z_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, a, b) = (q_0, ab)$
$\delta(q_0, a, z_0) = (q_0, b z_0)$
$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, b, a) = (q_0, ba)$ } push

$\delta(q_0, c, z_0) = (q_1, z_0)$
$\delta(q_0, c, a) = (q_1, a)$
$\delta(q_0, c, b) = (q_1, b)$ } skip

$\delta(q_1, a, a) = (q_1, \varepsilon)$ pop

$\delta(q_1, b, b) = (q_1, \varepsilon)$ pop

$\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$ skip

Q Design PDA for CFL

$L = \{ a^n b^m \mid n \le m \}$ or $\{0^n 1^m \mid n \le m\}$

$n=1\ m=1$ ab

$n=1\ m=2$ abb

$n=1\ m=3$ abbb

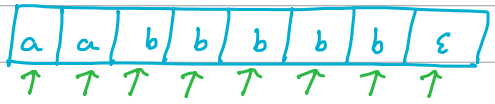
$n=2\ m=2$ aabb

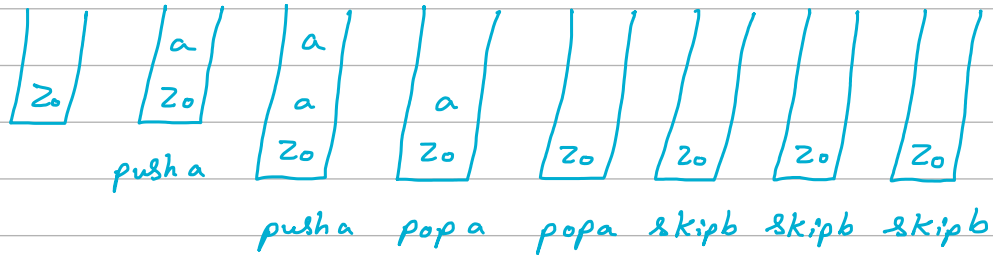
$n=3\ m=3$ aaabbb

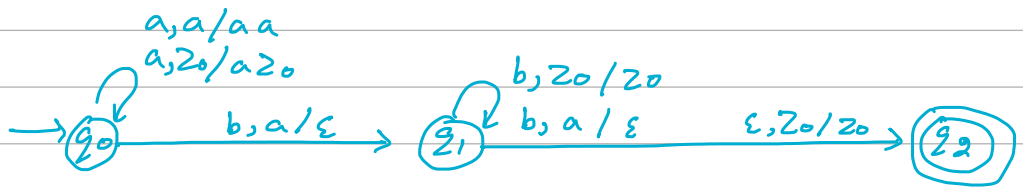
$n=2\ m=3$ aabbb

$n=2\ m=4$ aabbbb

$n=2\ m=5$ aabbbbb







$\delta(q_0, a, z_0) = (q_0, az_0)$ } push

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \varepsilon)$ } pop

$\delta(q_1, b, a) = (q_1, \varepsilon)$

$\delta(q_1, b, z_0) = (q_1, z_0)$ skip

$\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$ skip

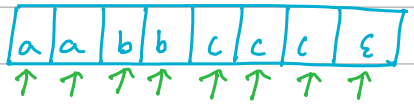
Q Design PDA for CFL

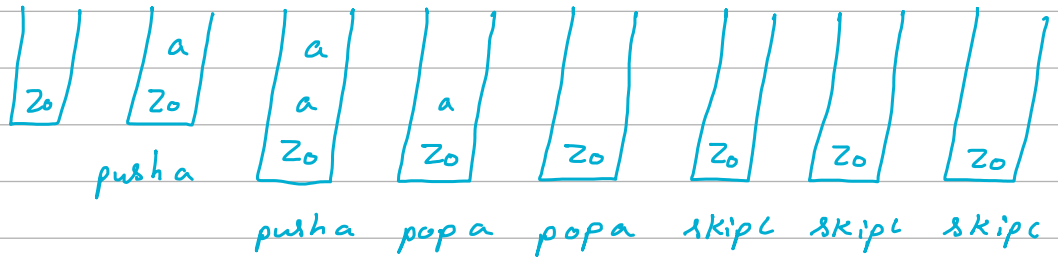
$L = \{ a^n b^n c^m \mid n, m \geq 1 \}$

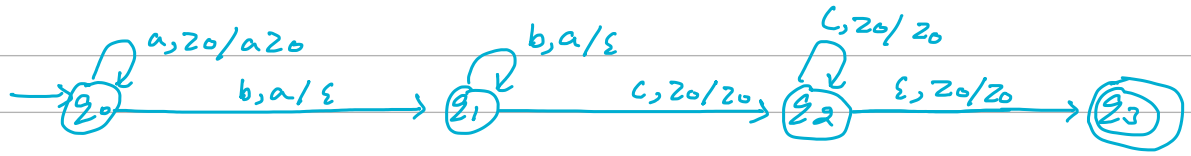
$n=1\ m=1$ abc

$n=1\ m=2$ abcc

$n=2\ m=1$ aabbc







$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ } push

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \varepsilon)$ } pop

$\delta(q_1, b, a) = (q_1, \varepsilon)$

$\delta(q_1, c, Z_0) = (q_2, Z_0)$ } skip c

$\delta(q_2, c, Z_0) = (q_2, Z_0)$

$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$ } skip

Q Design PDA for CFL

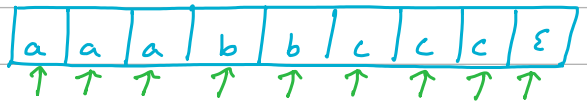
$L = \{ a^n b^m c^n \mid n, m \geq 1 \}$

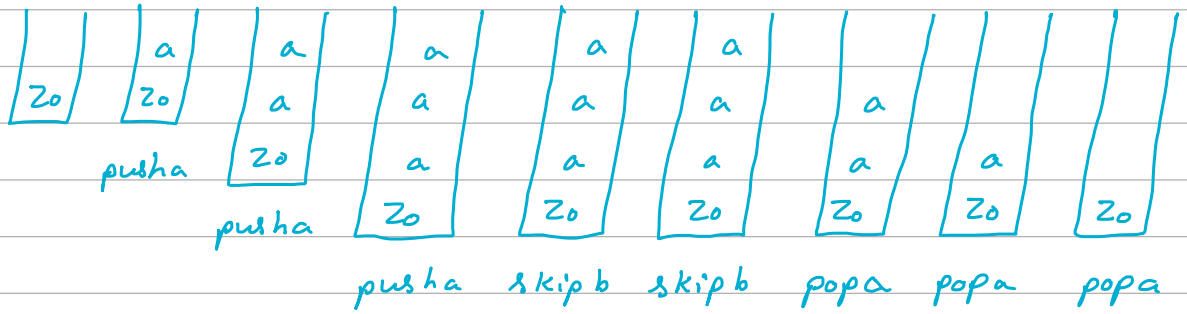
$n=1\ m=1$ abc

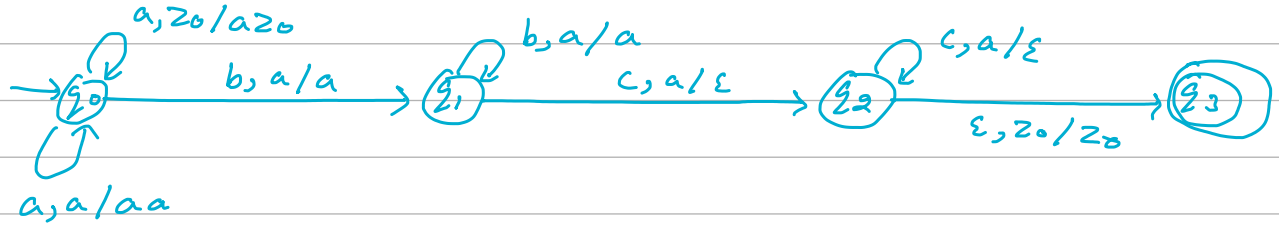
$n=2\ m=1$ aabcc

n = 3 m = 1 aaabccc

n = 3 m = 2 aaabbccc







$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ } push

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, a)$ } skip

$\delta(q_1, b, a) = (q_1, a)$

$\delta(q_1, c, a) = (q_2, \varepsilon)$ } pop

$\delta(q_2, c, a) = (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$ skip

Q Design PDA for CFL

$L = \{ a^{m+n} b^m c^n \mid m, n \geq 1 \}$

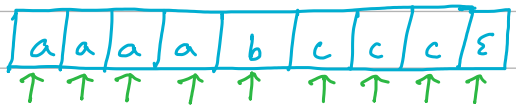
n = 1 m = 1 aabc

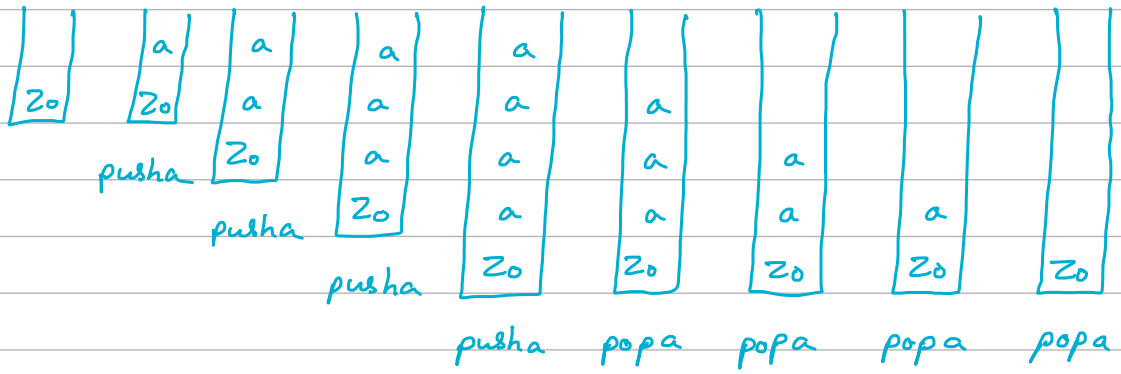
n = 1 m = 2 aaabbc

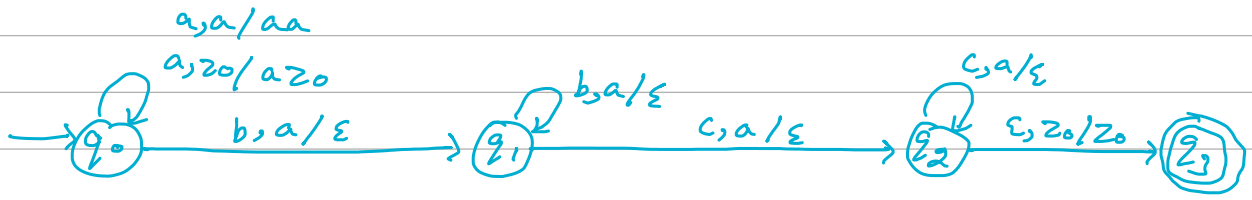
n = 1 m = 3 aaaabbbc

n = 2 m = 1 aaabcc

n = 3 m = 1 aaaabccc





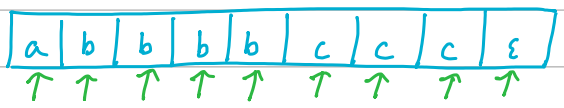


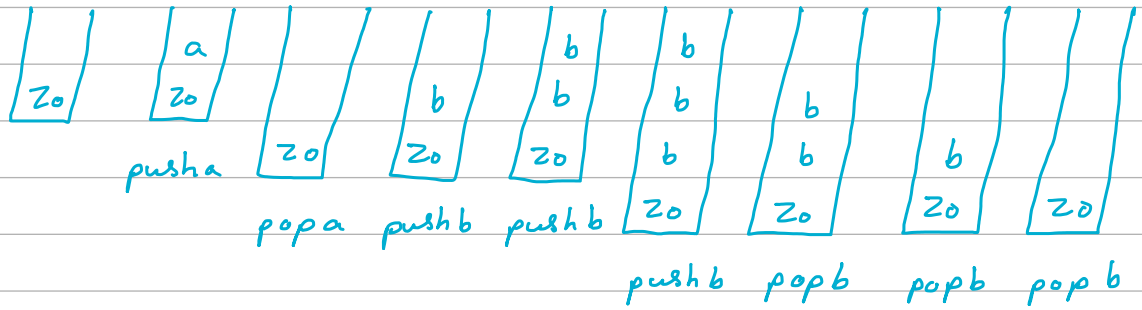
$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ } push
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \varepsilon)$ } pop
$\delta(q_1, b, a) = (q_1, \varepsilon)$
$\delta(q_1, c, a) = (q_2, \varepsilon)$ } pop
$\delta(q_2, c, a) = (q_2, \varepsilon)$
$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$ skip

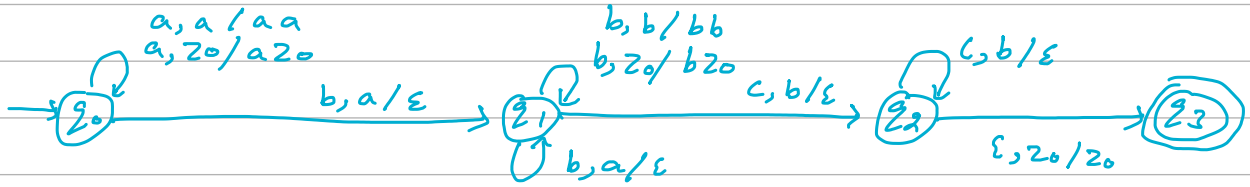
Q Design PDA for CFL

$L = \{ a^m b^{m+n} c^n \mid m, n \geq 1 \}$

n=1 m=1 abbc
n=1 m=2 aabbbc
n=1 m=3 aaabbbbc
n=2 m=1 abbbcc
n=3 m=1 abbbbccc





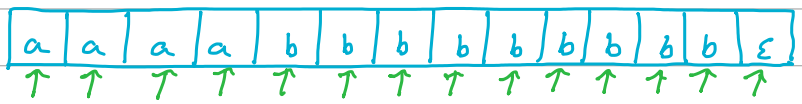


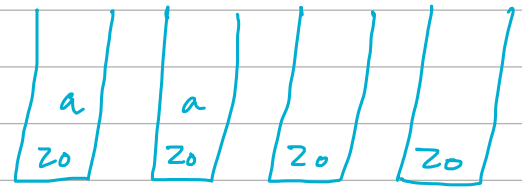
$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ } push
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \varepsilon)$ } pop
$\delta(q_1, b, a) = (q_1, \varepsilon)$
$\delta(q_1, b, Z_0) = (q_1, bZ_0)$ } push
$\delta(q_1, b, b) = (q_1, bb)$
$\delta(q_1, c, b) = (q_2, \varepsilon)$ } pop
$\delta(q_2, c, b) = (q_2, \varepsilon)$
$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$ skip

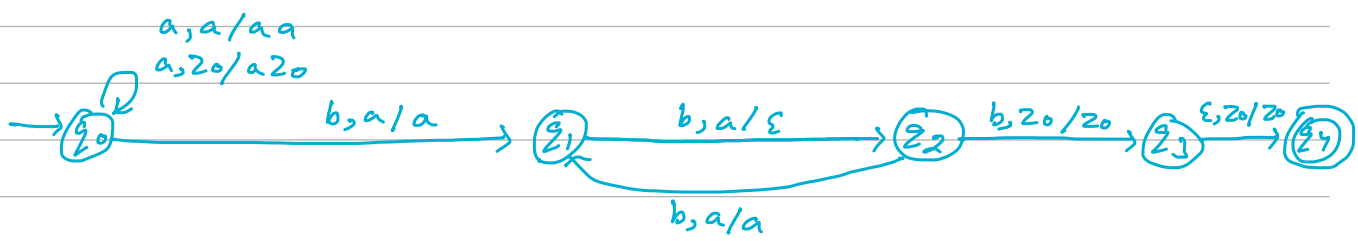
Q Design PDA for CFL

$L = \{ a^n b^{2n+1} \mid n \geq 1 \}$

n=1 abbb
n=2 aabbbbb
n=3 aaabbbbbbb
n=4 aaaabbbbbbbbb

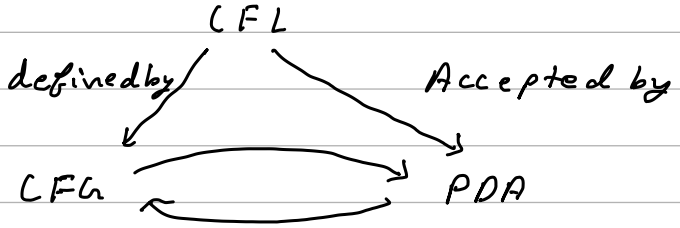






$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ } push
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, a)$ skip
$\delta(q_1, b, a) = (q_2, \varepsilon)$ pop
$\delta(q_2, b, a) = (q_1, a)$ skip
$\delta(q_2, b, Z_0) = (q_3, Z_0)$ skip
$\delta(q_3, \varepsilon, Z_0) = (q_4, Z_0)$ skip

Convert CFG to PDA



Rule 1 : for each variable A

$$\delta(q, \varepsilon, A) = (q, \beta) \quad \text{where } A \to \beta$$

i's a production of grammar

Rule 2 : For each terminal a

$$\delta(q, a, a) = (q, \varepsilon)$$

Q $S \to 0S1 \mid 00 \mid 11$

find the equivalent PDA for the given grammar

$\delta(q, \varepsilon, S) = (q, 0S1), (q, 00), (q, 11)$ Rule 1

$\delta(q, 0, 0) = (q, \varepsilon)$ Rule 2

$\delta(q, 1, 1) = (q, \varepsilon)$ Rule 3

Check these rule with some language so generate language and test

$S \to 0S1$

$\to 0111$

0111

$\delta(q, 0111, S)$ using rule 1

$\delta(q, 0111, 0S1)$

$\delta(q, \not{0}111, \not{0}S1)$ using rule 2

$\delta(q, 111, S1)$

$\delta(q, 111, 111)$ using rule 1

$\delta(q, \not{1}11, \not{1}11)$ using rule 3

$\delta(q, \not{1}1, \not{1}1)$ using rule 3

$\delta(q, \not{1}, \not{1})$ using rule 3

$\delta(q, \varepsilon, \varepsilon)$ Accept

Q Find the equivalent PDA for given grammer

$S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

$\delta(q, \varepsilon, S) = (q, 0BB)$ Rule 1

$\delta(q, \varepsilon, B) = (q, 0S), (q, 1S), (q, 0)$ Rule 2

$\delta(q, 0, 0) = (q, \varepsilon)$ Rule 3

$\delta(q, 1, 1) = (q, \varepsilon)$ Rule 4

Check these rule with some language so generate language and test

$S \rightarrow 0BB$

$B \rightarrow 0S / 1S / 0$

$S \rightarrow 0BB$

$0 0 S B$

$0 0 0 B B B$

$0 0 0 1 S 1 S 0$

$0 0 0 1 0 B B 1 0 B B 0$

$0 0 0 1 0 0 0 1 0 0 0 0$

$\delta(q, 0001000010000, S)$ using rule 1

$\delta(q, 0001000010000, 0BB)$

$\delta(q, \not{0}001000010000, \not{0}BB)$ using rule 3

$\delta(q, 001000010000, BB)$

$\delta(q, 001000010000, 0SB)$ using rule 2

$\delta(q, \not{0}01000010000, \not{0}SB)$ using rule 3

$\delta(q, 01000010000, SB)$

$\delta(q, 01000010000, 0BBB)$ using rule 1

$\delta(q, \not{1}1000010000, \not{0}BBB)$ using rule 3

$\delta(q, 1000010000, BBB)$

$\delta(q, 1000010000, 1SBB)$ using rule 2

$\delta(q, 1000010000, 1SBB)$ using rule 4

$\delta(q, 000010000, SBB)$

$\delta(q, 0000010000, 0BBBB)$ using rule 3

$\delta(q, 000010000, BBBB)$

$\delta(q, 00010000, OBBB)$ using rule 2

$\delta(q, 0010000, 0BBB)$ using rule 3

$\delta(q, 010000, BBB)$

$\delta(q, 01000, OBB)$ using rule 2

$\delta(q, 0100000, 0BB)$ using rule 3

$\delta(q, 10000, BB)$

$\delta(q, 10000, 1SB)$ using rule 2

$\delta(q, 1000, 1SB)$ using rule 4

$\delta(q, 0000, SB)$

$\delta(q, 0000, 0BBB)$ using rule 3

$\delta(q, 000, BBB)$

$\delta(q, 000, OBB)$ using rule 2

$\delta(q, 00, 0B)$ using rule 3

$\delta(q, 00, OB)$ using rule 2

$\delta(q, \phi 0, \phi B)$ using rule 3

$\delta(q, 0, 0)$ using rule 2

$\delta(q, \phi, \phi)$ using rule 3

$\delta(q, \varepsilon, \varepsilon)$ Accepted

Pumping Lemma for CFL

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

e.g. $L = \{a^n b^n c^n \quad n \geq 1\}$

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 x in L

[w null bhi ho sakta h]

$\underbrace{aa}_{u}\underbrace{a}_{v}\underbrace{bb}_{w}\underbrace{b}_{x}\underbrace{ccc}_{z}$

Case 1 :- $v = a^k$

$x = b^k$

$\underbrace{aaa}_{u}\underbrace{b}_{v}\underbrace{b}_{w}\underbrace{b}_{x}\underbrace{ccc}_{z}$

Case 2 :- $v = b^k$

$x = c^k$

$k = 2$

$u = aa$

$v^k = (a^1)^2$

$w = b$

$x^k = (b^1)^2$

$y = bccc$

aaaabbbbcccc

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

Assume L is content-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

① $|vwx| \leq n$

② $|vx| \geq 1$

③ for $i \geq 0$, uv^iwx^iy must in L

uv^iwx^iy must be in L

$|uwy| = q$ [q is prime no (assume)] [u, w and y composed of all a's)

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

pump $i = q$ times uv^qwx^qy

$= |uwy| + q|vx|$

$= 2 + qp$

$= q(1+p)$ → It is not a prime no

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