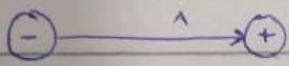
⇒ NFA (Non deterministic finite automata)

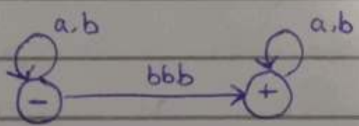
- allow missing edges
- allow duplicate edges

NFA can't be DFA but DFA can be NFA
NFA has different behavior on same input.

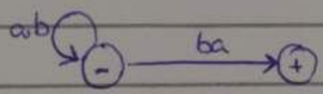
⇒ TG (Transition graph)

- Null 

- all conditions of NFA are allowed.
- Can take substring / substrings are allowed.



substring 'bbb'

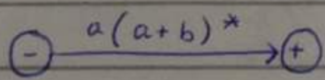


end with 'ba'

- Every DFA can be TG.

⇒ GTG (Generalize Transition Graph)

- Start from 'a'



- all conditions of NFA & TG
- GTG can't be DFA, but every DFA is GTG

Kleen's Theorem

Part 1

If a language can be accepted by DFA then it can be accepted by TG.

Prove

Every DFA itself called TG.

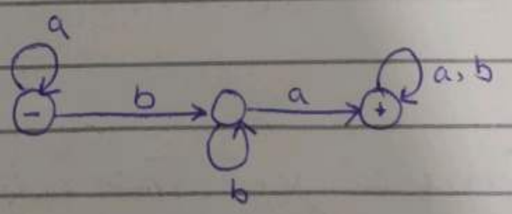
Part 2

If a language can be accepted by TG then it can be accepted by/expressed by RE as well.

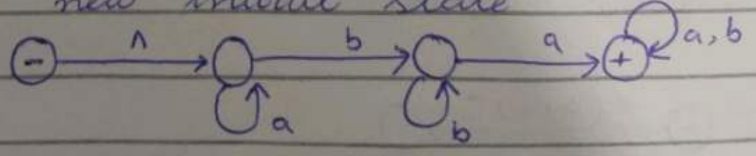
Prove

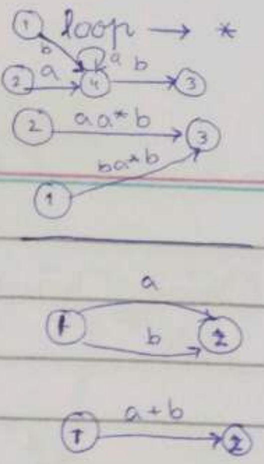
DFA → must have substring ba

Step 1

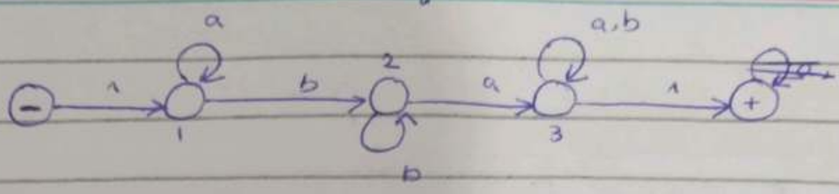


Step 2 : add new initial state



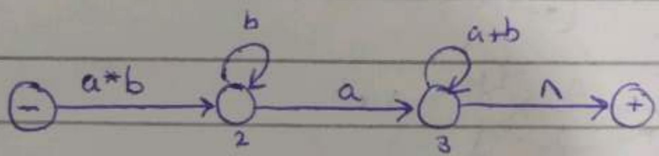


Step 3 : add new final state

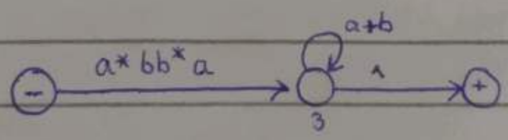


Choose the state which has less no. of edges.

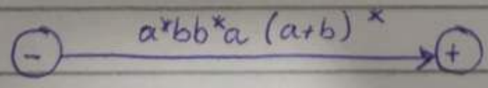
By removing '1' state



Now, removing '2'



Now, By removing '3'



$a(a+b)^*$ → union → Concatination → +/* operation

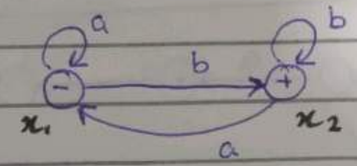
DFA → TG
TG → RE
RE → DFA

Part 3

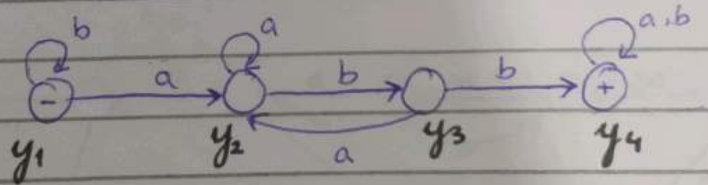
If a language is accepted by RE then there must be a DFA for that language.

DFA_1 → "end with b"
+
DFA_2 → must have "abb"

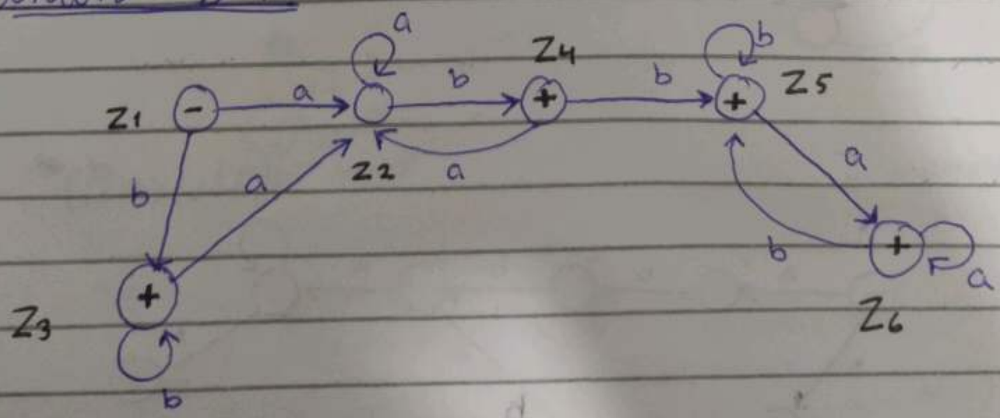
<u>DFA_1</u>



<u>DFA_1</u>



<u>Union DFA</u>



Union = dono mn sy kisi bhi RE ya DFA sy start ya end kr sakty.

Concatination = start from first. end at last.

	a	b
$Z_1(x_1, y_1)$	$x_1, y_2 = Z_2$	$x_2, y_1 = Z_3$
$Z_2(x_1, y_2)$	$x_1, y_2 = Z_2$	$x_2, y_3 = Z_4$
$Z_3(x_2, y_1)$	$x_1, y_2 = Z_2$	$x_2, y_1 = Z_3$
$Z_4(x_2, y_3)$	$x_1, y_2 = Z_2$	$x_2, y_4 = Z_5$
$Z_5(x_2, y_4)$	$x_1, y_4 = Z_6$	$x_2, y_4 = Z_5$
$Z_6(x_1, y_4)$	$x_1, y_4 = Z_6$	$x_2, y_4 = Z_5$

$Z_1 = x_1, y_1$
$Z_2 = x_1, y_2$
$Z_3 = x_2, y_1$
$Z_4 = x_2, y_3$
$Z_5 = x_2, y_4$
$Z_6 = x_1, y_4$

Final states for Union

x_2, y_4

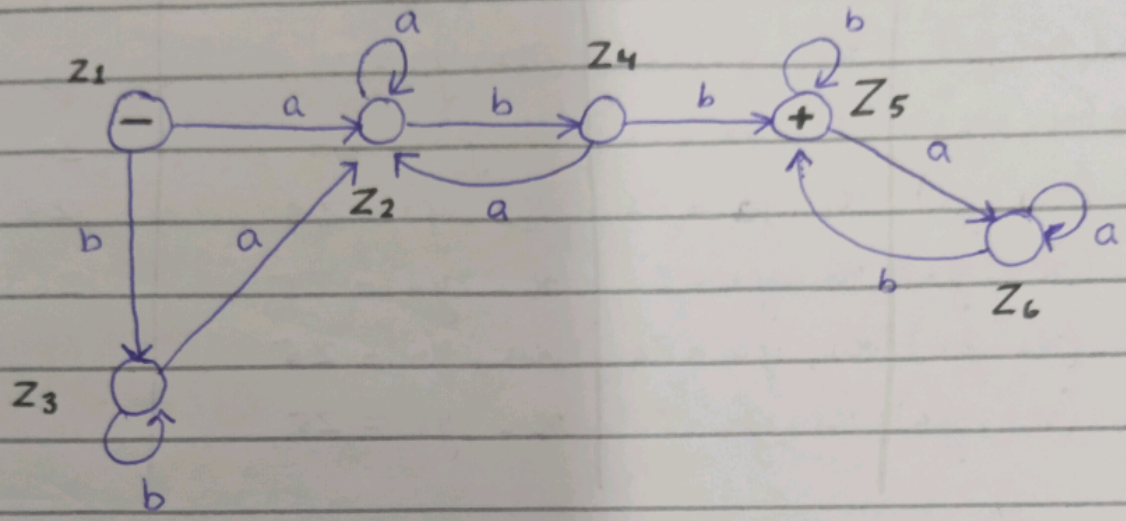
+ $Z_3 = x_2, y_1$
+ $Z_4 = x_2, y_3$
+ $Z_5 = x_2, y_4$
+ $Z_6 = x_1, y_4$

Final states for intersection.

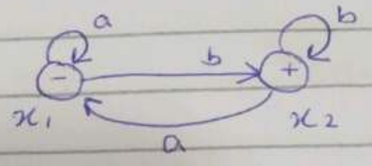
x_2, y_4

+ $Z_5 = x_2, y_4$

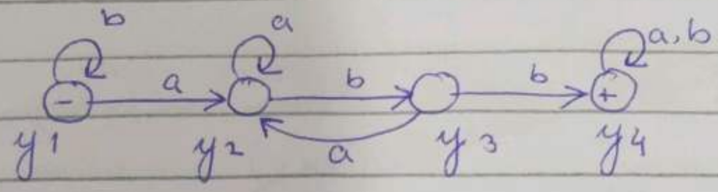
Intersection DFA



DFA1



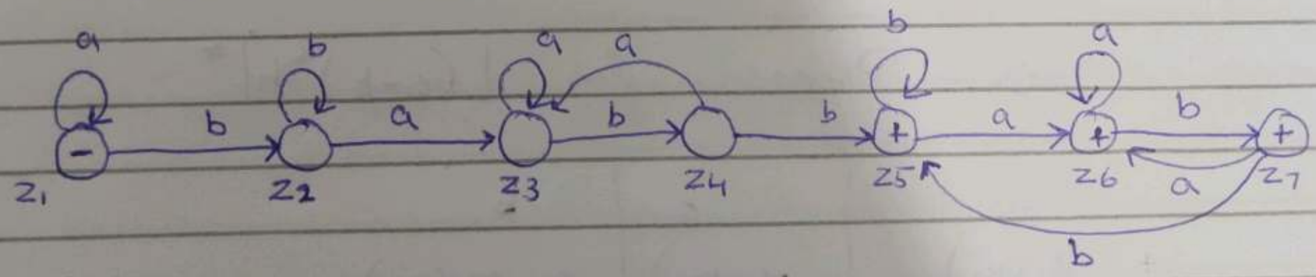
DFA2



Concatination DFA

	a	b
$z_1(x_1)$	$x_1 = z_1$	$x_2, y_1 = z_2$
$z_2(x_2, y_1)$	$x_1, y_2 = z_3$	$x_2, y_1 = z_2$
$z_3(x_1, y_2)$	$x_1, y_2 = z_3$	$x_2, y_1, y_3 = z_4$
$z_4(x_2, y_1, y_3)$	$x_1, y_2 = z_3$	$x_2, y_1, y_4 = z_5$
$z_5(x_2, y_1, y_4)$	$x_1, y_2, y_4 = z_6$	$x_2, y_1, y_4 = z_5$
$z_6(x_1, y_2, y_4)$	$x_1, y_2, y_4 = z_6$	$x_2, y_1, y_3, y_4 = z_7$
$z_7(x_2, y_1, y_3, y_4)$	$x_1, y_2, y_4 = z_6$	$x_2, y_1, y_4 = z_5$

$z_1 = x_1$
$z_2 = x_2, y_1$
$z_3 = x_1, y_2$
$z_4 = x_2, y_1, y_3$
$z_5 = x_2, y_1, y_4$
$z_6 = x_1, y_2, y_4$
$z_7 = x_2, y_1, y_3, y_4$



⇒ All finite languages are regular (True)

⇒ RE, TG, DFA → regular language

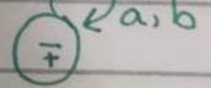
⇒ Every finite language has DFA
→ Every DFA has TG
→ Every TG has RE.

⇒ non regular language → No DFA
→ No TG
→ No RE
→ Has PDA

Example
$a^n b^n$, $n \geq 0$

⇒ Subset of regular language is always regular? (False)

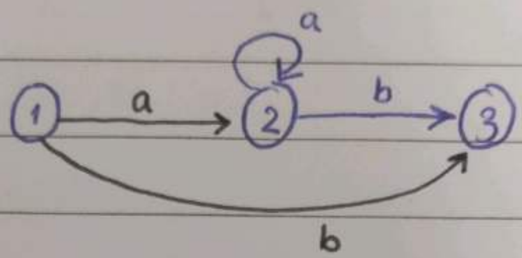
Example
$(a+b)^*$



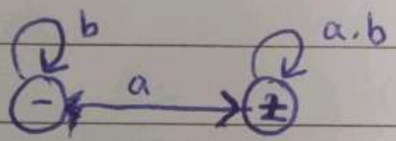
→ Regular language

Union of NFA

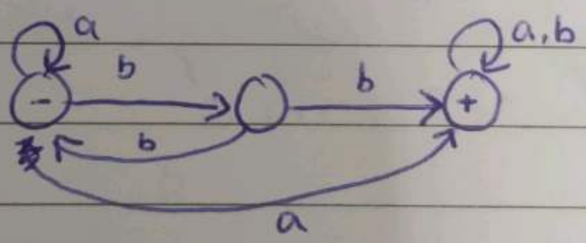
جس state سے جوڑنا ہے اس کی B or A جہاں جہاں
جا رہے ہیں ، جس کو جوڑنا ہے اس کو بھی وہاں وہاں جوڑیں گے۔



NFA₁



NFA₂

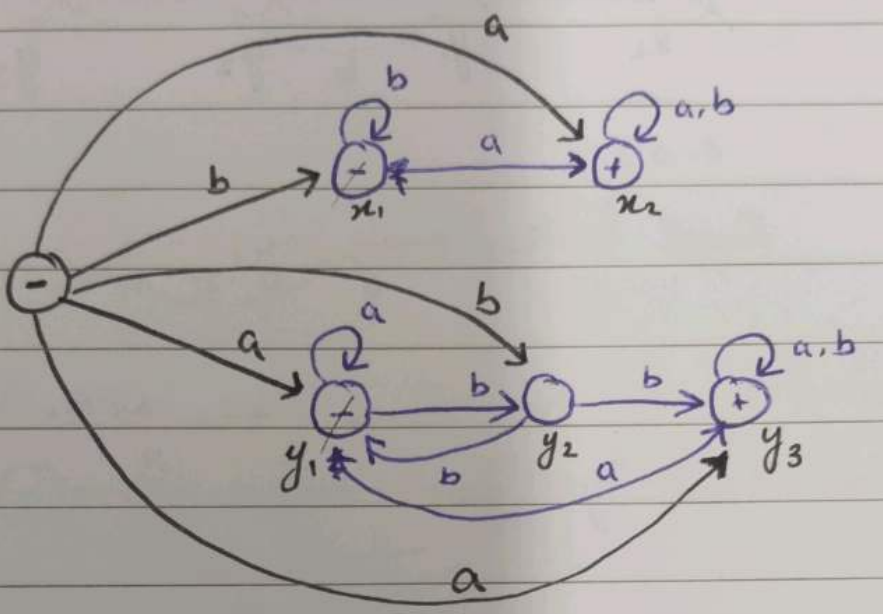


Steps:

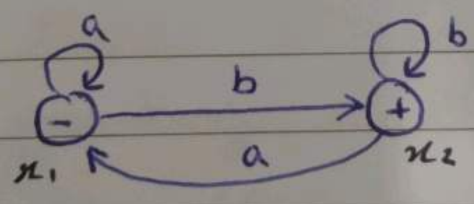
Make new initial

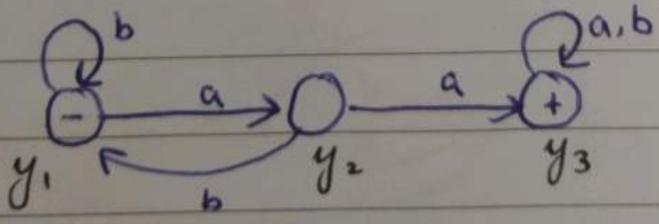
Connect with pervious initials.

Union NFA



Concatination





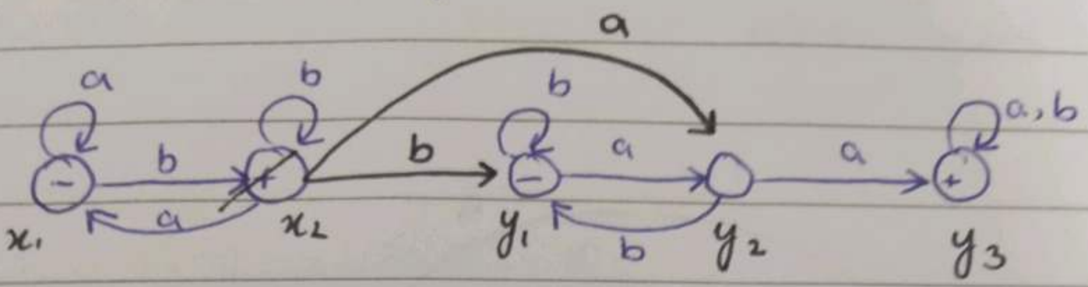
Steps

→ 1st ki starting last ki ending

→ 1st ki ending join ho jayegi 2nd ki start sy.

→ x_2 k a & b bhi y_1 ki A or B wali states per jayen gy.

Concatination of NFA



Final state only 'y_3'

→ NFA_1 ki har Final ko NFA_2 in initial sy join karen gy.

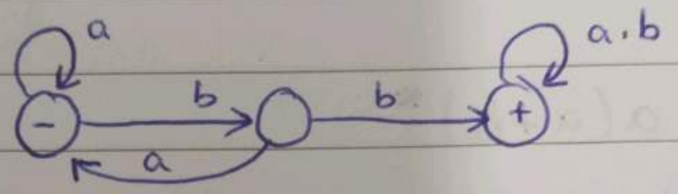
Question

kis condition mn NFA_1 ki final remove nhi hogi?

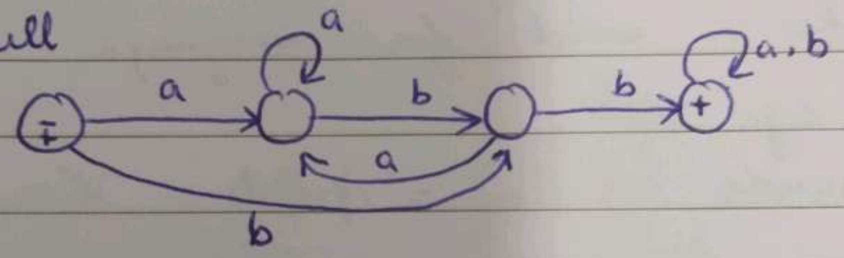
⇒ when NFA_2 is producing null.

⇒ * operation.

new initial & also make it final for null

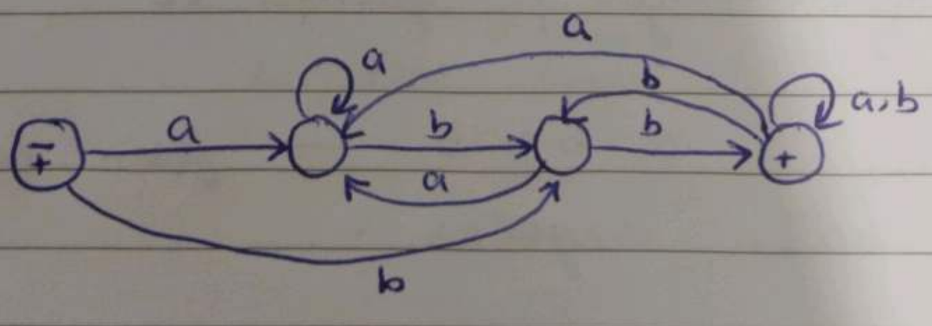


For null



For repetition.

→ Join initial with Final.
→ ~~Final~~ ki edges us state pr jayen gi jaha initial ki edges ja rhi hn.



Pumping Leema

Idenify ~~the~~ whether the language is regular ~~and~~ ~~wh~~ or not.

$a(a+b)^*$

Take a word for language
eg abb
Divide it in 3 parts

w Before null
y in loop
x After null

$\underbrace{a}_{w}\ \underbrace{b}_{y}\ \underbrace{b}_{x}$

Take the y & pump it

a b b b → ~~from~~ language

so, regular.

$\Rightarrow$ $a^n b^n$ language

e.g. $\underbrace{aa}_{y}bb$

pump the 'y' $\rightarrow$ aaaabb $\rightarrow$ not from language

So, not a regular language.

$\Rightarrow$ $a^* b^*$

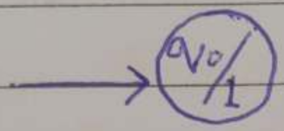
$\underbrace{ab}_{y}$ $\longrightarrow$ abab $\rightarrow$ not from language.

$\underbrace{a}_{y}b$ $\rightarrow$ aab $\rightarrow$ from langiqe.

$\rightarrow$ Agr 1 bhi word language ka aa jye to language regular hai.

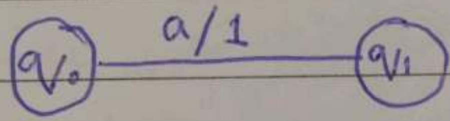
⇒ Moore & mealy Machine.

→ no accepting & rejecting.
→ input & sath output kia a rhi hai.
→ no final state, jahan rukty hn wohi final ban jati hai



— moore

output state k andr.



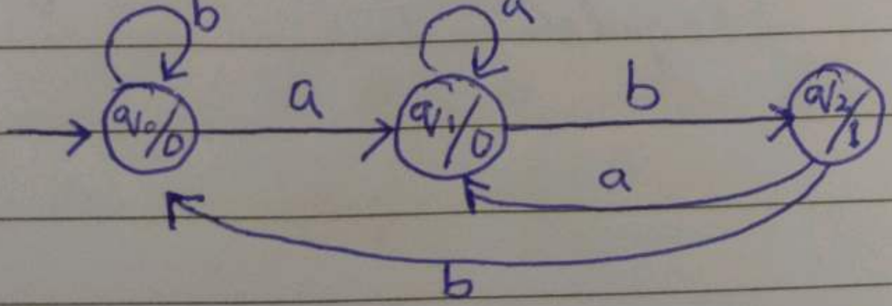
— mealy

output edge pr.

Output – J = {0, 1}

__Question__

machine produces ab (Moore)



⇒ 3 ab's in input = 3 1's in output.

<u>Question</u>

Machine that produces ab (mealy)