ch-7. Kleene's Theorem

Th Any language that can be defined by
regular expression or,
finite Automata or,
transition graph, can be defined by all three methods.

Proof:- The three sections of our proof will be:

Part 1: Every language that can be defined by a finite Automata can also be defined by a transition graph.

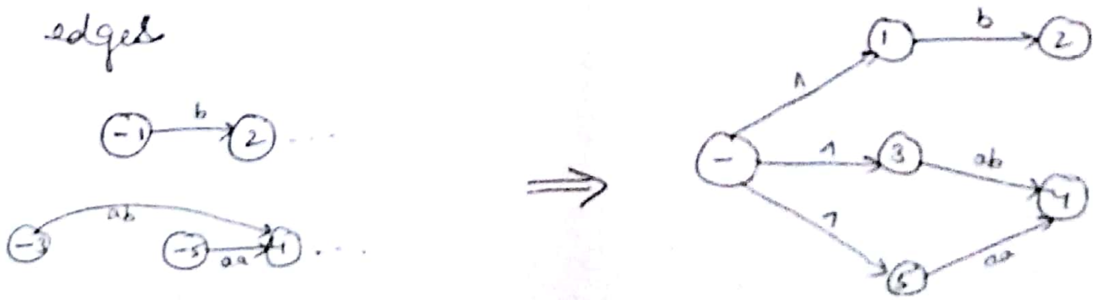
Part 2: Every language that can be defined by a transition graph can also be defined by a regular expression.

Part 3: Every language that can be defined by a regular expression can also be defined by a finite Automata.

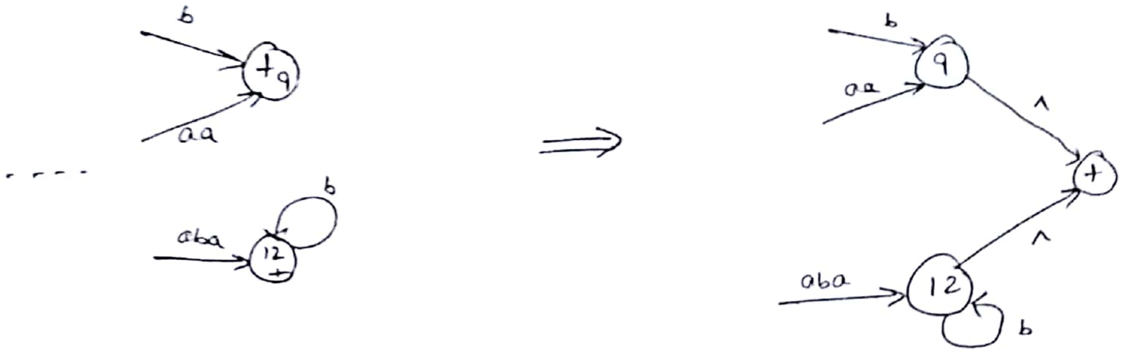
Proof of Part 1 : Every finite automata is itself already a transition graph. Therefore, any language that has been defined by a FA. has already been defined by a transition graph.

Proof of Part 2 : TG's to Regular Expressions.

Case-I only one start state should be there with no incoming edges.

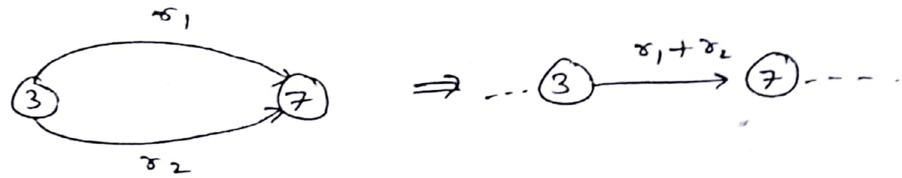


Case II. Single final state with no outgoing edges:-

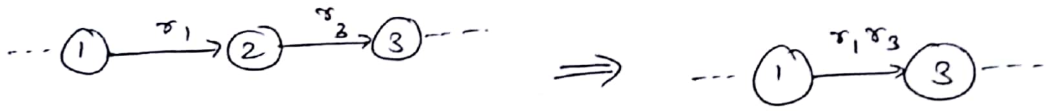


Case III.

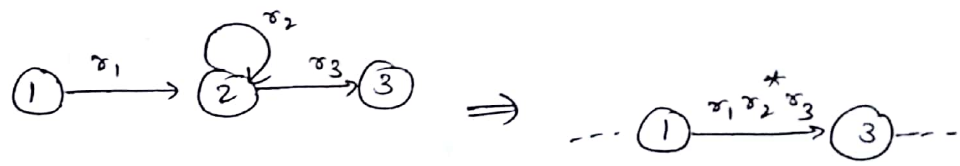
eg.



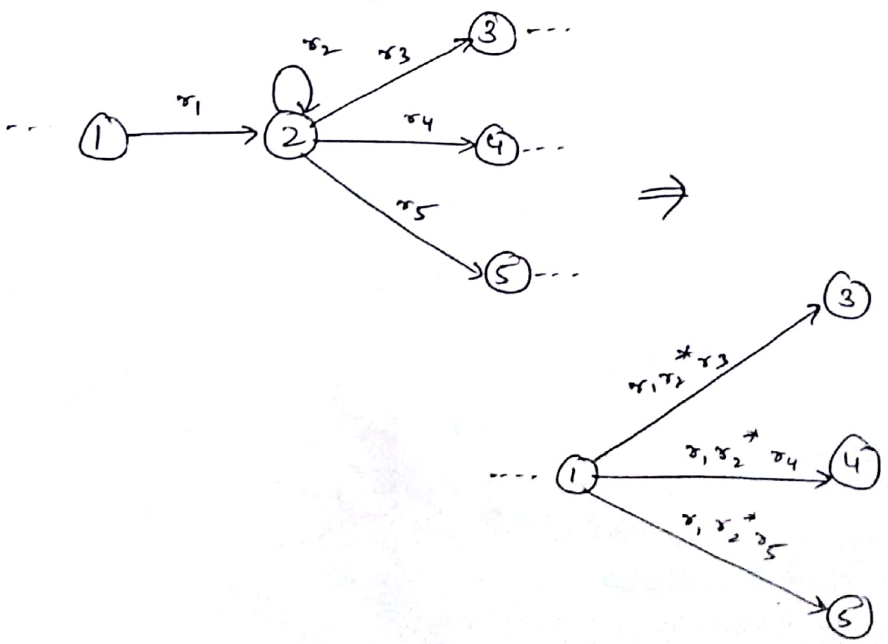
Case IV.



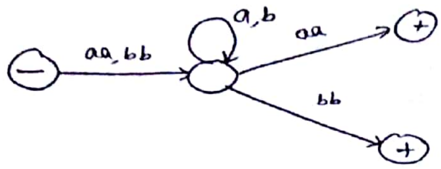
eg.



eg.

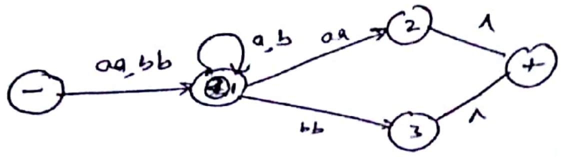


eg: Convert TG to Regular Expression :-

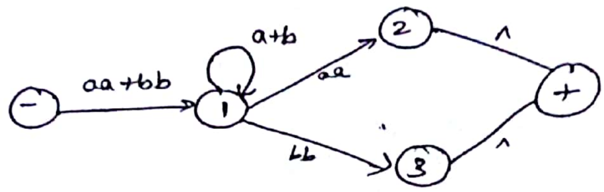


Sol^n :- Two final states

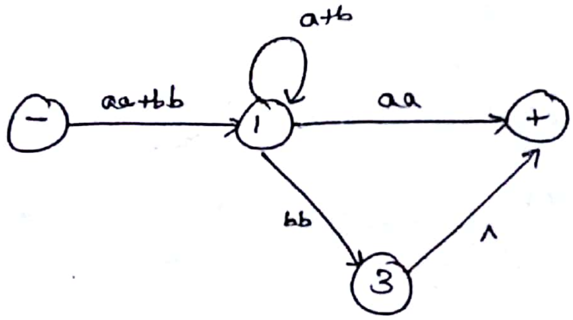
∴



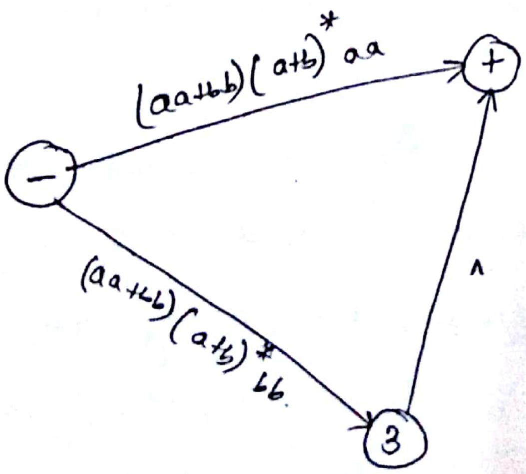
Then,



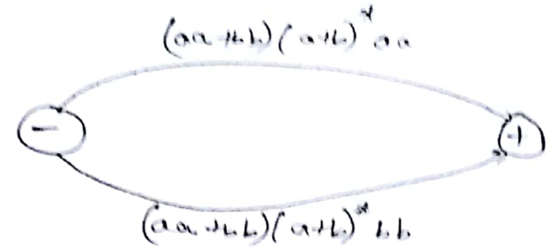
ByPass state 2,



ByPass state 1,



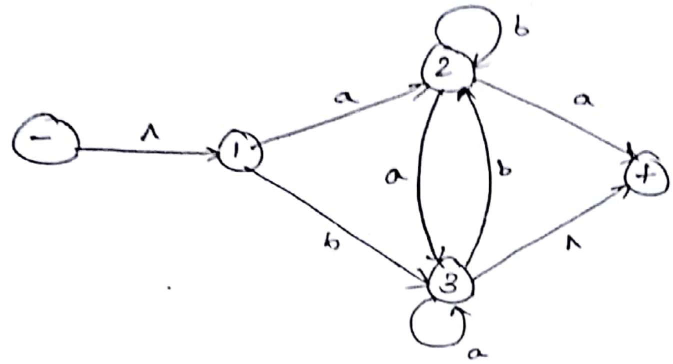
By Pass state (3),



$\therefore$

$$(aa+bb)(a+b)^*aa \;+\; (aa+bb)(a+b)^*bb.$$

Exercise:- TG to Reg. Expression.



eliminate the states in order 1, 2, 3.

Ans. → $ab^*a + [b + ab^*a][a + bb^*a]^*[\Lambda + bb^*a]$

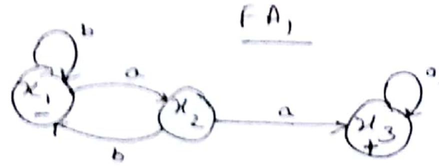
Eliminate in order 3, 2, 1

Ans:- $ba^* + [a + ba^*b][b + aa^*b]^*[a + aa^*]$.

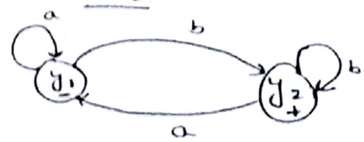
Proof 3 (a) If there is an FA called FA_1 that accepts the language defined by the regular Expression r_1 and there is an FA called FA_2 that accepts the language defined by the regular expression r_2, then there is an FA that we shall call FA_3 that accepts the language defined by the regular expression $(r_1 + r_2)$.

eg.

FA₁



FA_2



$- z_1 = x_1$ or y_1 (z_1 is start state of both the machines).

'a' at $z_1 = x_2$ or $y_1 \Rightarrow z_2$

'b' at $z_1 = x_1$ or $y_2 \Rightarrow z_3(+)$

'a' at $z_2 = x_3$ or $y_1 \Rightarrow z_4(+)$

'b' at $z_2 = x_1$ or $y_2 \Rightarrow z_3(+)$

'a' at $z_3 = x_2$ or $y_1 \Rightarrow z_2$

'b' at $z_3 = x_1$ or $y_2 \Rightarrow z_3(+)$

'a' at $z_4 = x_3$ or $y_1 \Rightarrow z_4(+)$

'b' at $z_4 = x_3$ or $y_2 \Rightarrow z_5(+)$

'a' at $z_5 = x_3$ or $y_1 = z_4(+)$

'b' at $z_5 = x_3$ or $y_2 = z_5(+)$

$(-) Z_1 = x_1$ or y_1

$Z_2 = x_2$ or y_1

$(+) Z_3 = x_1$ or y_2 [y_2 is final state]

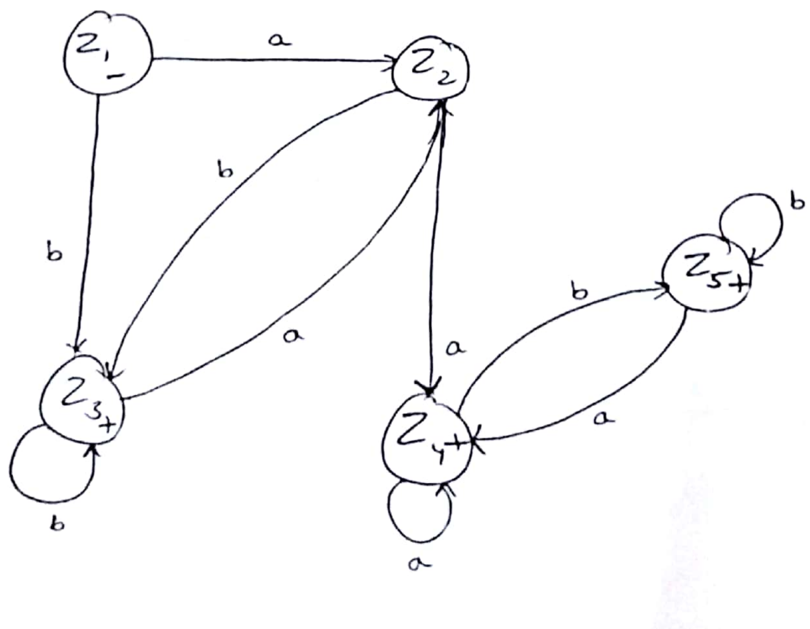
$(+) Z_4 = x_3$ or y_1

$(+) Z_5 = x_3$ or y_2

Transition Table

	a	b
$(-) Z_1$	Z_2	Z_3
Z_2	Z_4	Z_3
$(+) Z_3$	Z_2	Z_3
$(+) Z_4$	Z_4	Z_5
$(+) Z_5$	Z_4	Z_5

The whole machine looks like this

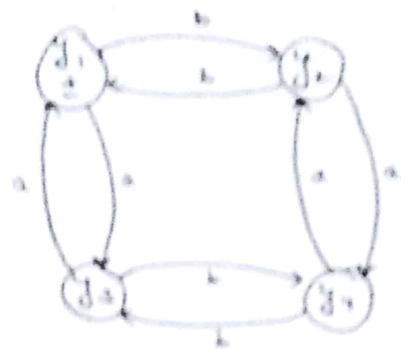


Exercises

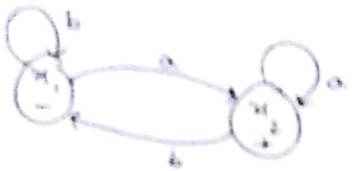
(1)

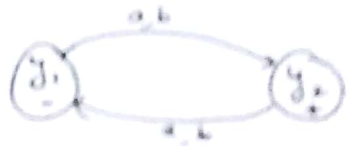
FA_1

FA_2

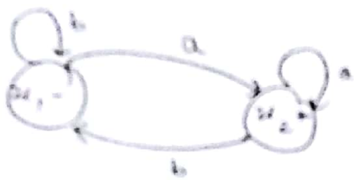


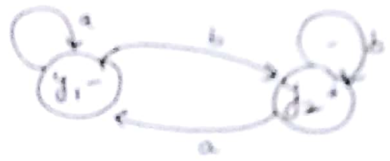
(2)





(3)



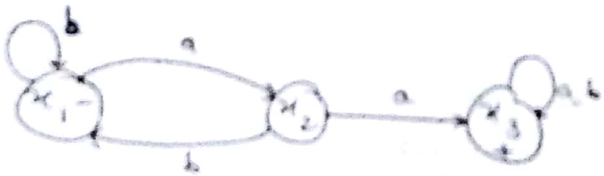


Assign

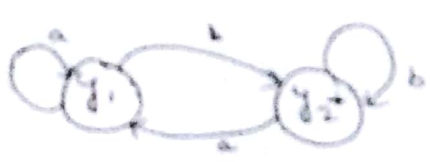
Proof 3 (b). If there is an FA_1 that accepts the language defined by the regular expression r_1 and an FA_2 that accepts the language defined by the regular expression r_2, then there is an FA_3 that accepts the language defined by the concatenation r_1r_2, the product language.

eg

FA_1



FA_2



Scanned by CamScanner

Solⁿ:

The start state is $\boxed{z_1 = x_1}$, $x_1 = z_1 (-)$ $(-)\ z_1 = x_1$

'a' at z_1 = x_2 (z_2)

'b' at z_1 = x_1 (z_1)

'a' at z_2 = x_3 or y_1 (z_3)

'b' at z_2 = x_1 (z_1)

'a' at z_3 = x_3 or y_1 (z_3)

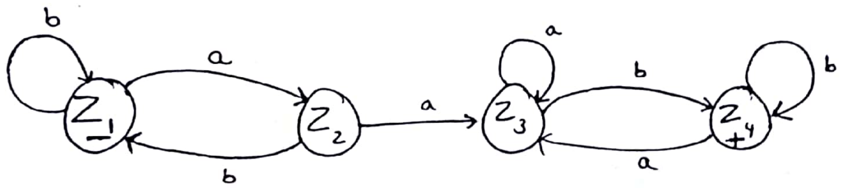
'b' at z_3 = x_3 or y_1 or y_2 (z_4) +

'a' at z_4 = x_3 or y_1 (z_3)

'b' at z_4 = x_3 or y_1 or y_2 (z_4) +

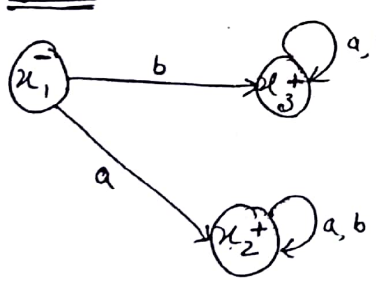
transition table

	a	b
$z_1\ (-)$	z_2	z_1
z_2	z_3	z_1
z_3	z_3	z_4
$z_4\ (+)$	z_3	z_4

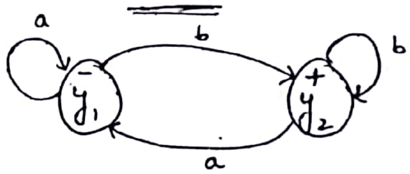


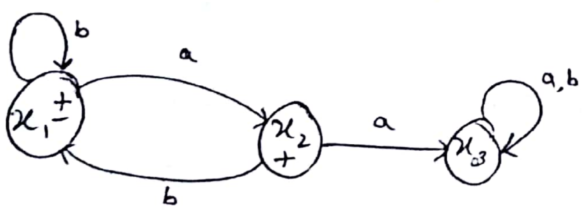
Exercise :-

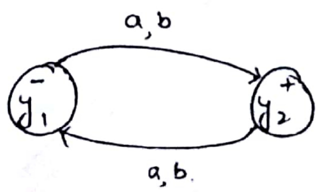
FA₁



FA₂



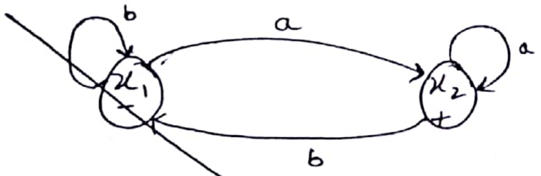




Proof 3(c).

If r is a regular expression and FA_1 is a finite Automaton that accepts exactly the language defined by r, then there is an FA called FA_2 that will accept exactly the language defined by r^*.

eg



(−) $z_1 = x_1$

'a' at z_1 = x_2 or x_1 (z_2)

'b' at z_1 = x_1 (z_1)

'a' at z_2 = x_2 ~~or~~ x_1 (+)

'b' at z_2 = x_1 ~~or~~ (z_1)

Transition Table

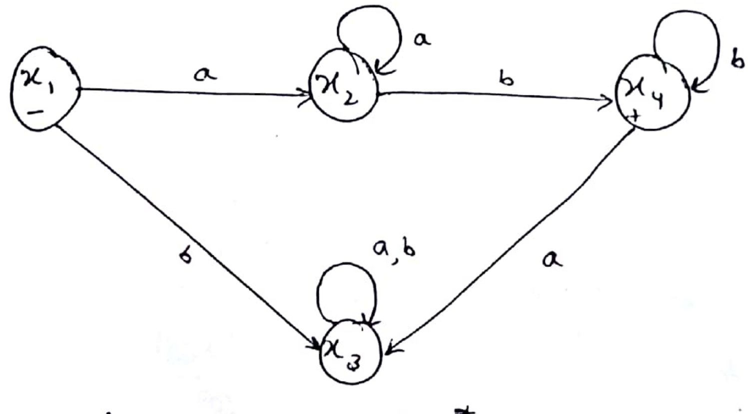
	a	b
z_1		
z_2		
z_3		

Two cases:-

1). start state has no loop.

2). start state has ~~no~~ a loop.

Case I. No loop.



Since, we have to find r^*, Add ± at the start state (to accept Λ).

Solⁿ

$(\pm)\ z_1 = x_1$ [start state].

'a' at $z_1 = x_2\ [z_2]$

'b' at $z_1 = x_3\ [z_3]$

'a' at $z_2 = x_2\ [z_2]$

'b' at $z_2 = x_4$ or $x_1\ [z_4](+)$

'a' at $z_3 = x_3\ [z_3]$

'b' at $z_3 = x_3\ [z_3]$

'a' at $z_4 = x_3$ or $x_2\ [z_5]$

'b' at $z_4 = x_4$ or x_1 or $x_3\ [z_6]\ (+)$

'a' at $z_5 = x_3$ or $x_2\ [z_5]$

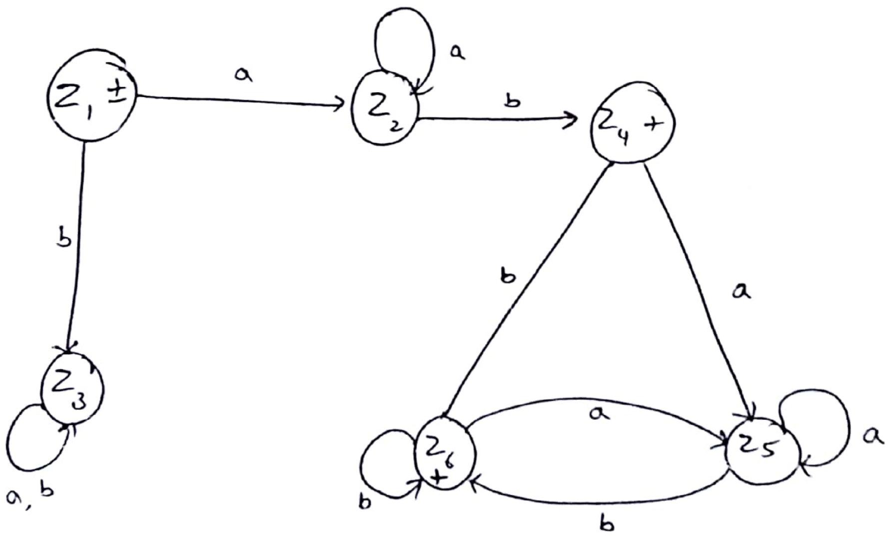
'b' at $z_5 = x_3$ or x_4 or $x_1\ [z_6]\ (+)$

'a' at $z_6 = x_3$ or $x_2\ (z_5)$

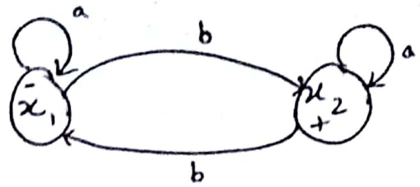
'b' at $z_6 = x_3$ or x_4 or $x_1\ [z_6]\ (+)$.

Transition Table.

	a	b
$(\pm)\ z_1$	z_2	z_3
z_2	z_2	$z_4(+)$
z_3	z_3	z_3
$(+)\ z_4$	z_5	z_6
z_5	z_5	z_6
$(+)\ z_6$	z_5	z_6



Case II Loop at start state



$Z_1 = x_1$ and final state ($\pm$)

$Z_2 = x_1$ and a non-final state.

'a' at $Z_1 = x_1 = Z_2$

b at $Z_1 = x_2$ or $x_1 = Z_3$ (+)

'a' at $Z_2 = x_1 = Z_2$

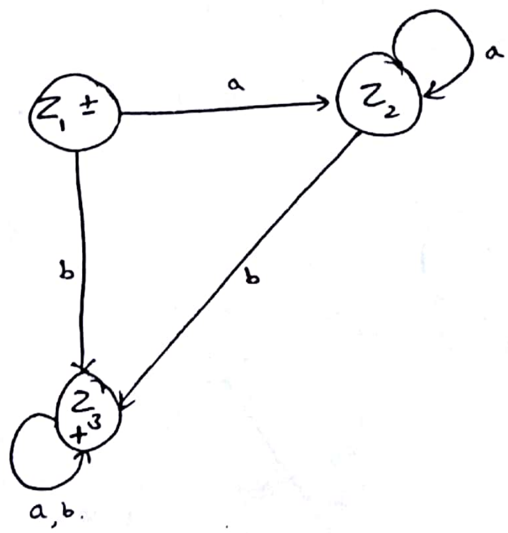
'b' at $Z_2 = x_2$ or $x_1 = Z_3$ (+)

'a' at $Z_3 = x_2$ or x_1 (Z_3)(+)

'b' at $Z_3 = x_1$ or $x_2 = Z_3$ (+)

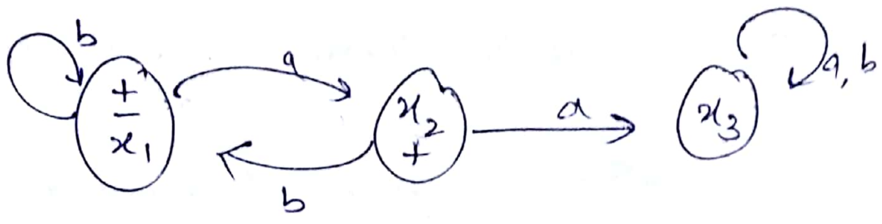
Transition Table.

	a	b
($\pm$) Z_1	Z_2	Z_3
Z_2	Z_2	Z_3
(+) Z_3	Z_3	Z_3

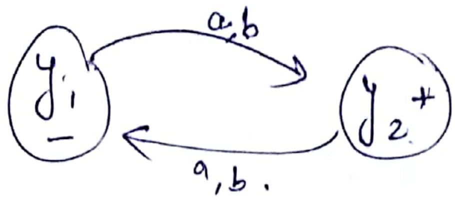


$FA_1 . FA_2$

FA_1



FA_2



$(-) z_1 = x_1 \text{ or } y_1$

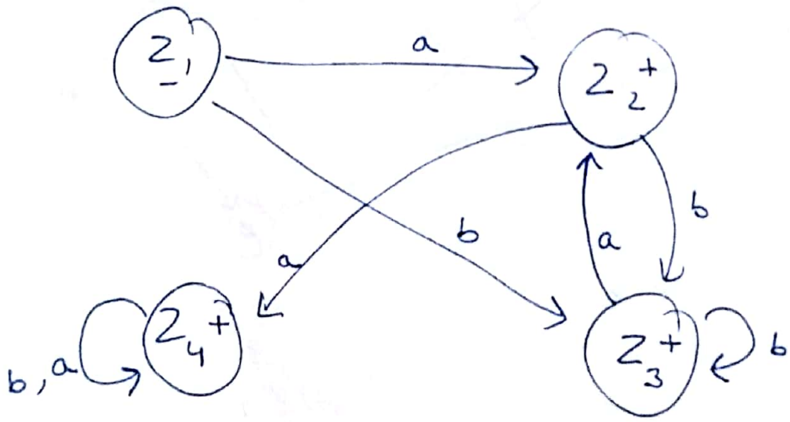
a at $z_1 = x_2$ or y_1 or $y_2(+)$ $(z_2)(+)$
b at $z_1 = x_1$ or y_1 or or $y_2(+)$ $(z_3)(+)$

a at $z_2 = x_3$ or $y_2(+)$ or y_1 $(z_4)(+)$
b at $z_2 = x_1$ or y_1 or $y_2(+)$ $(z_3)(+)$

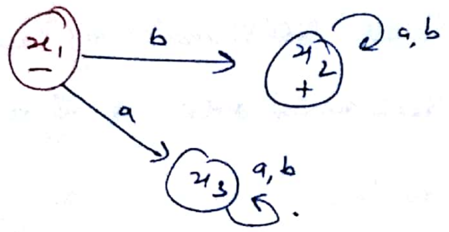
a at $z_3 = x_2$ or y_1 or $y_2(+)$ $(z_2)(+)$
b at $z_3 = x_1$ or y_1 or $y_2(+)$ $(z_3)(+)$

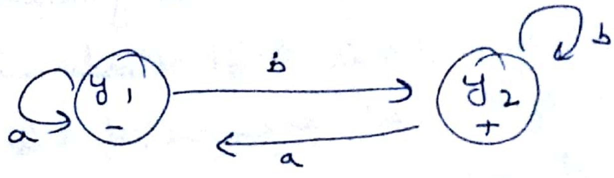
a at $z_4 = x_3$ or y_1 or $y_2(+)$ $(z_4)(+)$
b at $z_4 = x_3$ or y_1 or $y_2(+)$ $(z_4)(+)$

	a	b
(−) z_1	z_2	z_3
(+) z_2	z_4	z_3
(+) z_3	z_2	z_3
(+) z_4	z_4	z_4



Q. find $FA_1 + FA_2$





(−) z_1 = x_1 or y_1

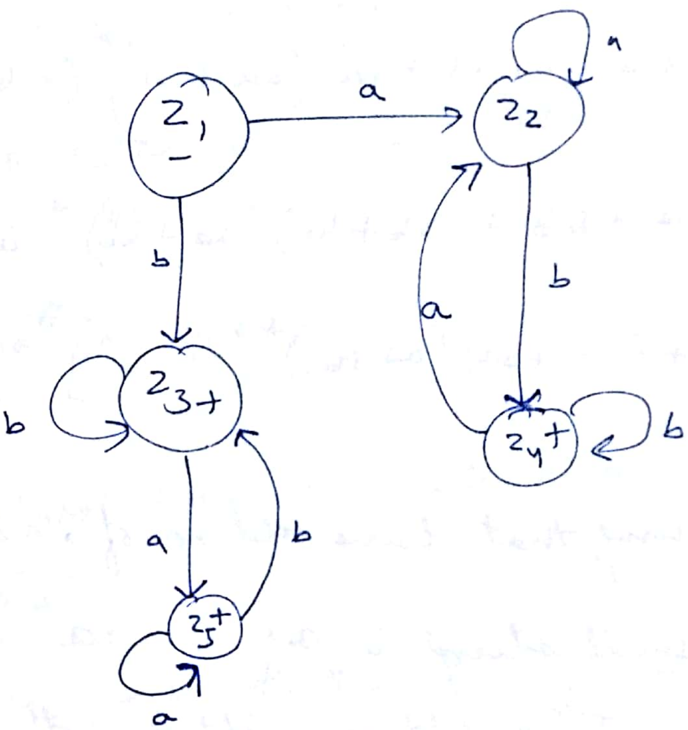
'a' at z_1 = x_3 or y_1 (z_2)
'b' at z_1 = x_2 or y_2 (z_3)(+)

'a' at z_2 = x_3 or y_1 (z_2)
'b' at z_2 = x_3 or y_2(+) (z_4)

'a' at z_3 = x_2 or y_1 (z_5)(+)
'b' at z_3 = x_2 or y_2 (z_3)

'a' at z_4 = x_3 or y_1 (z_2)
'b' at z_4 = x_3 or y_2 (z_4)(+)

'a' at z_5 = x_2 or y_1 (z_5)(+)
'b' at z_5 = x_2 or y_2(+) (z_3)



	a	b
(−) z_1	z_2	z_3
z_2	z_2	z_4
(+) z_3	z_5	z_3
(+) z_4	z_2	z_4
(+) z_5	z_5	z_3

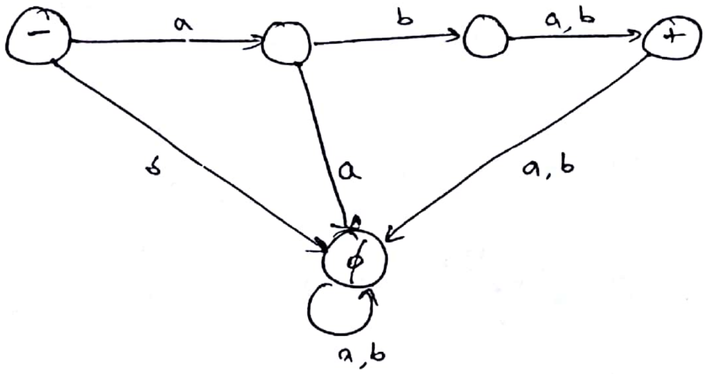
Complements And Intersection

Complements

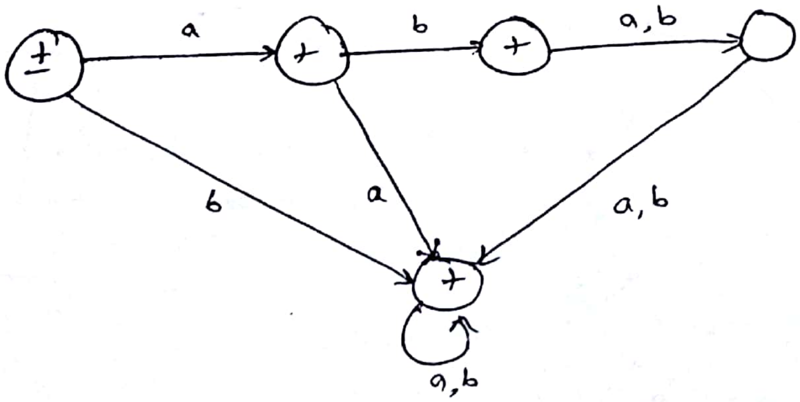
Def^n :- If L is a language over the alphabet Σ, we define its complement, L' to be the language of all strings of letters from Σ that are not words in L

- \+ $\Rightarrow$ Non final state — $(+) \Rightarrow ( )$
- Nonfinal state $\Rightarrow$ final state — $( ) \Rightarrow (+)$
- (–) start state becomes $\Rightarrow$ $\pm$ — $(-) \Rightarrow (\pm)$.

e.g. FA that accepts the string aba and abb.



FA that accepts all strings other than aba and abb is :-



Intersection

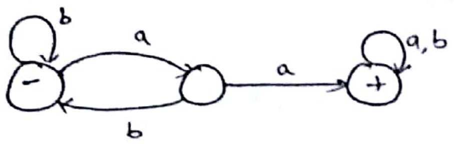
Th. If L_1 & L_2 are regular languages, then $L_1 \cap L_2$ is also a regular language. That is, set of regular languages is closed under intersection.

Proof

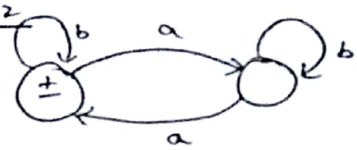
$$L_1 \cap L_2 = (L_1' + L_2')' \quad \text{[Demorgan's Law]}$$

eg.

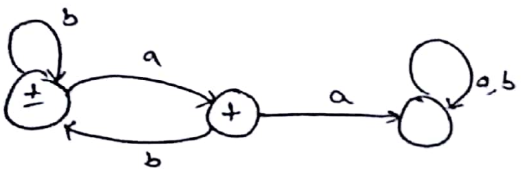
FA$_1$



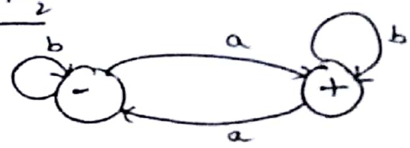
FA$_2$



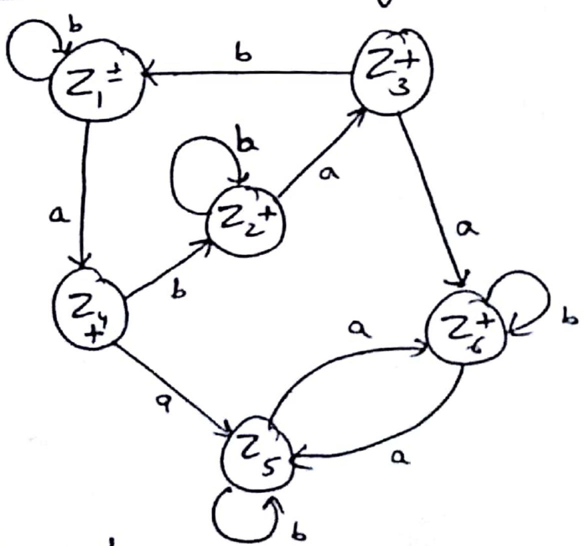
FA$_1'$



FA$_2'$



$(FA_1' + FA_2')$ [using Kleene's Theorem (Proof 3(a))]

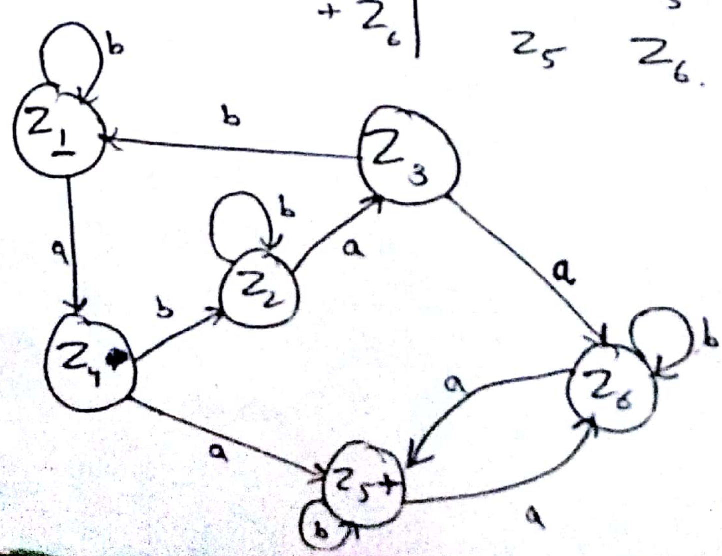


⟹

Transition Table

	a	b
$\pm z_1$	z_4	z_1
$+ z_2$	z_3	z_2
$+ z_3$	z_6	z_1
$+ z_4$	z_5	z_2
z_5	z_6	z_5
$+ z_6$	z_5	z_6.

ow, find $(FA_1' + FA_2')'$, i.e



(16)

Exercise

1) find intersection of foll Languages:-
   words with a double 'a' and EVEN-EVEN.

2) all strings with a double 'a', all strings with an even no of a's.

3) all words that begin with an a, all words that end with an 'a'.