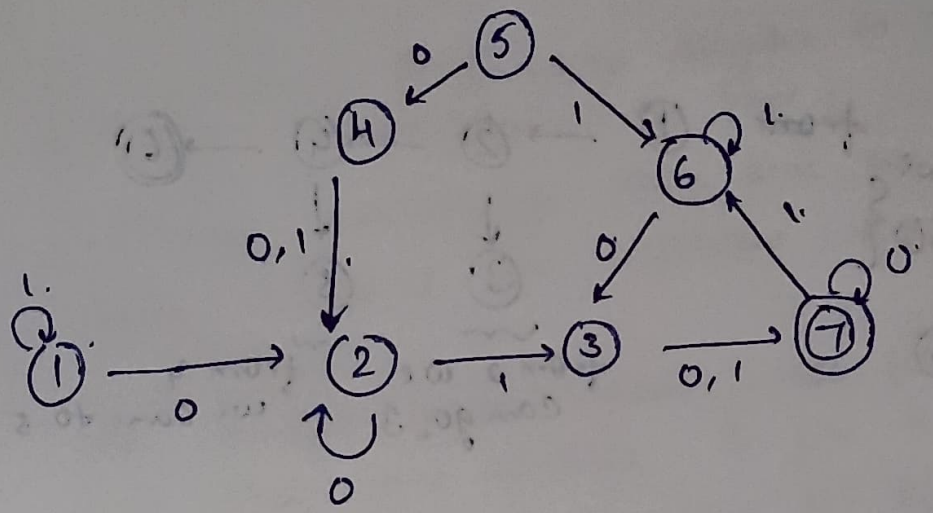
Theory Assignment

Automata Theory.

(1)

a).



✓ for state (5) there is no incoming transition & its is not initial state also. so (5) cant be reached from initial state. so (5) is unreachable state.

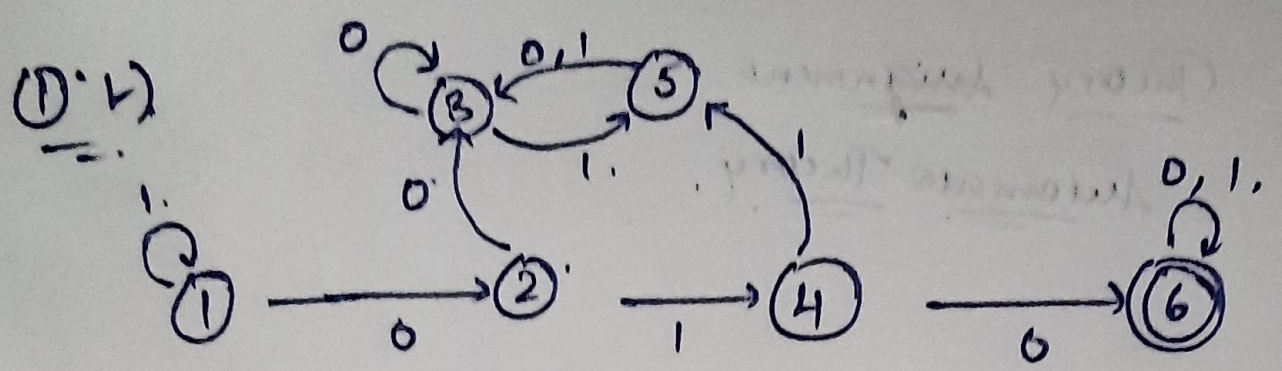
✓ similarly (4) has only incoming transition. that is from (5) which is unreachable so (4) is also unreachable state.

✓ from (1) we can go to (2) then to (3) to then to (7) then to (6) → (3) → (7) so there are no dead state.

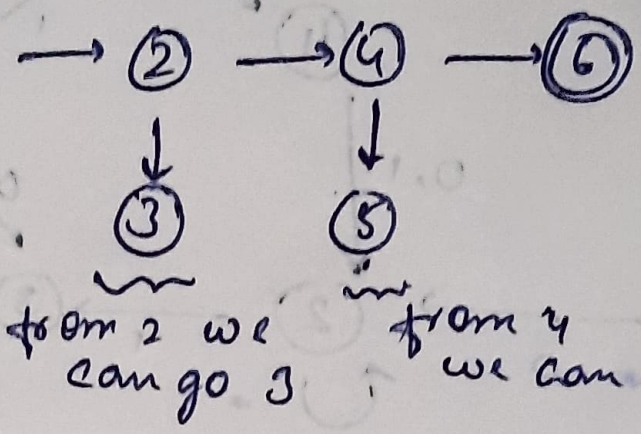
Set of spurious states are :- {(4), (5)} ✓

Ans

(1) (4)



→ from (1) we can go to (2) then to (4) then to (6)



from 2 we can go 3

from 4 we can to 5

so every state is reachable.

so if we get to state (3) then we can go to (5) & from there we can only go to (3). so once we get to state (3) we can only go to (5), or remain in (3). NO path to final state. so both (3), (5) are dead states

∴ set of spurious states are :- {(3), (5)}.

(2) Clearly as there are no outgoing transitions from (1) on 0 it goes to. State (8) is a dead state.

from (1) —0→ (2)

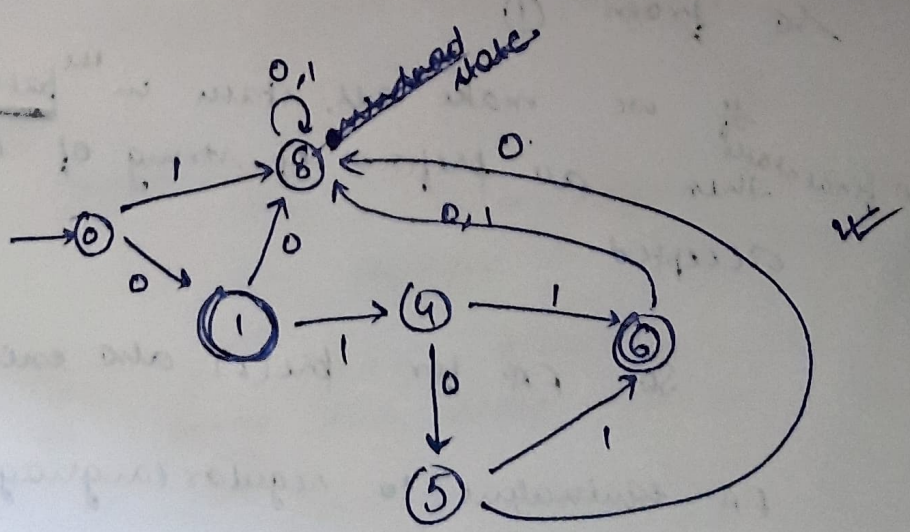
from (2) on 1 it goes to (3) & on 0 it goes to dead state.

from (3) it goes to (2) on 0 & goes to dead on 1

Clearly (2), (3) are also dead state.

No outgoing transitions from (7) also so (7) is also dead state.

for DFA one dead state is enough.



(8) → dead state.

③

Prove :- pre(L) is regular whenever L is regular

Proof :- As L is regular, we can draw finite automaton for L. (FA) — ①

eg :- w = aabba

prefixes (w) = {ε, a, aa, aab, aabb, aabba}

~~so prefix (L) is that first small~~

pre(L) = set of prefix of all strings of L

prefix :- substring that starts from the beginning of the string.

So from ①

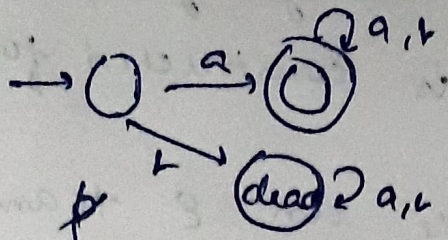
If we make all states in the path of final. also final states. then all prefixes of string of L are accepted.

So FA for pre(L) also exist

FA equivalent to regular language

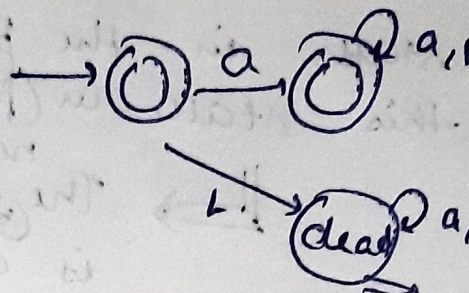
⇒ pre(L) is regular whenever L is regular proved.

eg : $L = a(a+L)^*$



↳ final state.

pre(L) =



Can't be final state because string through this path dont reach final so not accepted by L. so neither by pre(L).

(5)

Let, $A = a + b + c + d + e + \ldots x + y + z$.

Let, $B =$ research.iiit.ac.in + students.iiit.ac.in + iiit.ac.in

Let, $C = \varepsilon + .$

Regular expression : $AA^* \circ C \circ A^* \circ @ \circ B$

concatenation

eg1 : shchakra@iiit.ac.in.

$AA^* \circ C \circ A^* \circ @ \circ B$

shchakra ε ε @ iiit.ac.in.

no accepted.

eg 2 : zeeshan.ahmed@research.iiit.ac.in

AA^* C A^* @ B.

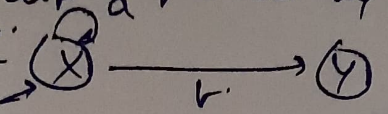
no accepted

(6) (a) Regular expression to RLG.

step: (1) Convert regular expression to finite automaton.

step (2) In the finite automata focus only on the outgoing transitions of the states. Write the production as output followed by the state on which transition is going.

eg:-



↳ $X \xrightarrow{a} X \Rightarrow X \to aX$ (present state, input symbol, next state)

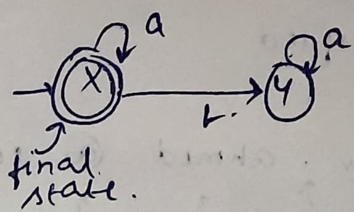
↳ $X \xrightarrow{b} Y \Rightarrow X \to bY$

$X \to aX \mid bY$.

step (3) In this do for all states. (focus on outgoing states as mentioned above)

step (4) If final state then add ε at the end of derivation.

eg:-



$X \xrightarrow{a} X \equiv X \to aX$

$X \xrightarrow{b} Y \equiv X \to bY$

$X \to \varepsilon$ ← as final state

$X \to aX \mid bY \mid \varepsilon$

$Y \xrightarrow{a} Y \equiv Y \to aY$

Ans RLG

$X \to aX \mid bY \mid \varepsilon$

$Y \to aY$

⑥ b). Right linear grammar to left linear grammar.
(RLG) (LLG)

Sol first convert right linear grammar to FA (finite automaton.) using following steps :-

step 1 :- start from the first production, make an ε transition to all those which are in RHs of production

step 2 :- for every left alphabet go to symbol followed by it.

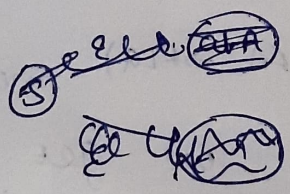
step 3 :- Repeat this for all productions. and make state with ε as final.

eg :-

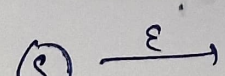
S → abA

A → as | bA | ε

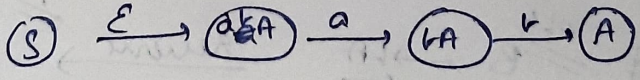
step 1 :-



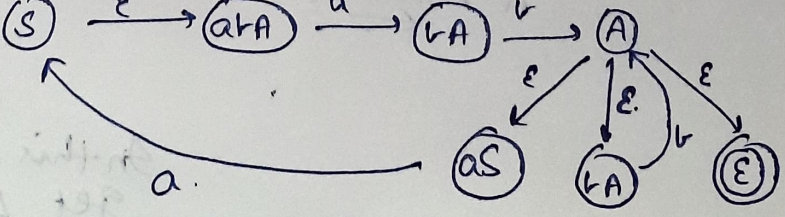
step 1 :-



step 2 :-



step 3 :-



Now we get
right linear grammar to finite automaton.
(let FA1)

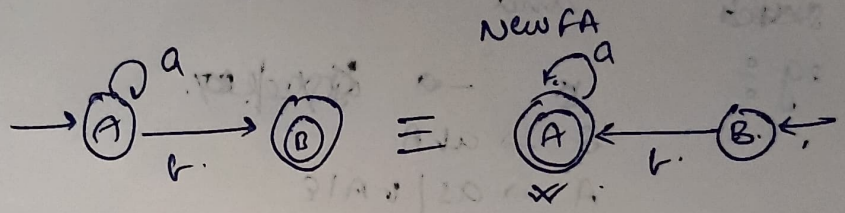
in
New FA mentioned previous page.
we now convert FA1 to LLG :=
↓
Left linear grammar

FA ⟹ reversal. New FA ⟹ RLG ⟹ reverse LLG

using method mentioned in previous page. we get RLG

Step 4: reverse all the transitions & make initial state as final & final as initial

eg :=



Step 5: from method (FA → RLG) mentioned in (steps 1-3) we get RLG.

Step 6. reverse the RHS of productions to get LLG.

eg := RLG
B → LA
A → AA | ε

reverse ⟹

A → Aa | ε
B → AL

↓

In this way we get LLG

7) a) Let us consider a string w of a language given.

Let, it be a regular language.

By pumping lemma for regular languages, we will divide the string into 3 parts x, y, z such that $|y| > 0$ & $|xy| \leq p$ (pumping length)

and all elements of xy^*z must belong to that language.

Here length of string = prime.

⇒ length can be 2, 3, 5, 7, 11, 13, 17, ——

clearly There is no any sequence.

Let $|y| = k$ such that $|xy| \leq p$.

$\therefore xyz = w$

Let length of $w = n$

len → length

$len(xy^0z) = n - k$ (k ≥ 0)

$len(xyz) = n$

$len(xy^2z) = n + k$

$len(xy^3z) = n + 2k$

(+k)

⇓ there is a sequence.

But seeing prime numbers there is no any sequence b/w prime numbers.

Thus by contradiction.

It is not regular.

7) (b)

Let w be a string of language L.

$L = \{a^{n!} \mid n \geq 0\}$

Let L be a regular language.

Let $P \rightarrow$ pumping length.

By pumping lemma, we divide the string into 3 parts x, y, z $|xy| \leq P$, $|y| \geq 0$ and all element in xy^*z. must be in L because we considered it as string

Let say $|y| = k (\geq 0)$

$w = xyz$ in L

xy^2z in L

xy^3z in L

xy^4z in L

⋮

Because we considered L is regular.

→ as the strings of L contain only a's. y also has a's. By x, z has a's only

Now, from xyz to xy^2z k more a's are added. By to xy^3z, k more a's are added. so on.

clearly there is a sequence — (1)

$a^{n!} = \{a^1, a, a^2, a^6, a^{24}, a^{120}, \ldots\}$

NO sequence found

so we cant find a sequence of numbers for $|y| > 0$

⇒ Not regular

by contradiction.

(11) for $i=j$ $\quad i, j, k \geq 0$

$a^i b^j c^k$.

we push a's & we pop a's when we read b. & skip c's when read

for $j = k$.

$a^i b^j c^k$

we push a's & push b's & pop b's corresponding to number of c's.

~~of course~~ ~~we use~~ of course we dont use.

we will any way push a. — (1)

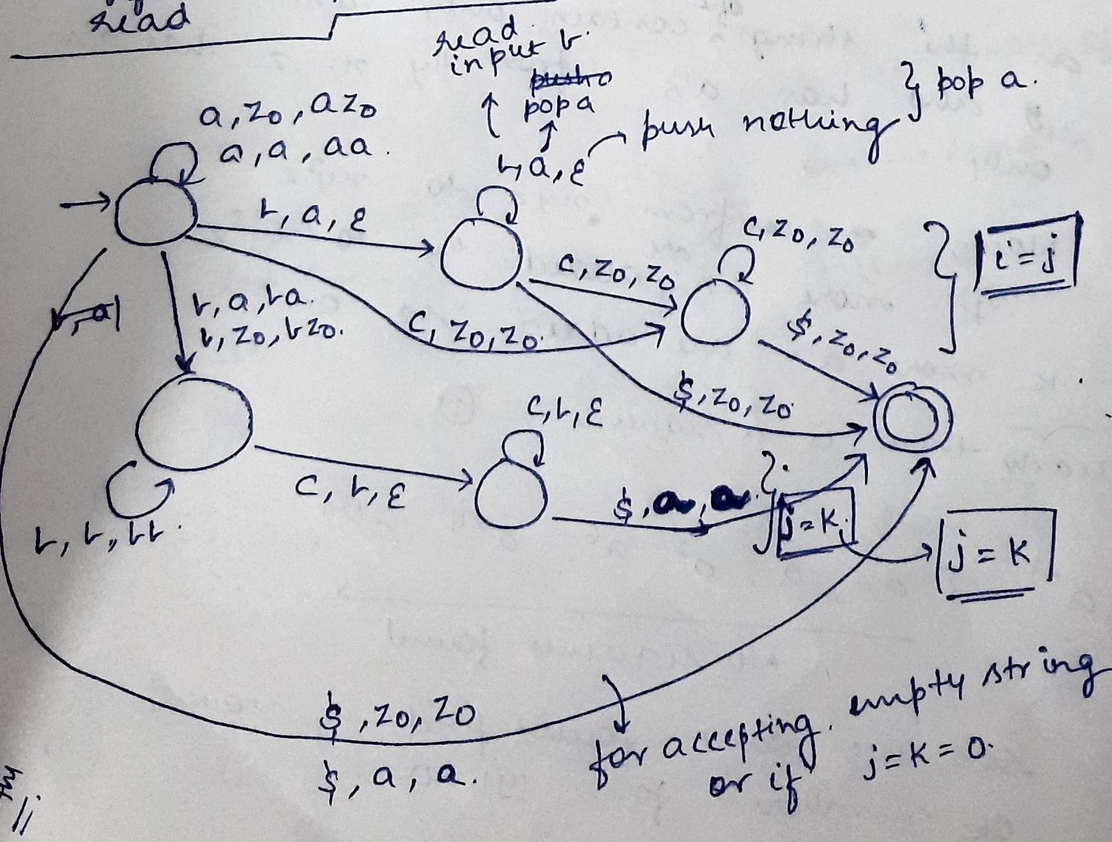
from then we can ~~goto~~ go in two paths.

let Z_0 → top of the stack

let \$ → end of string

format used: — — — (*)

input read, pop, push



Ans //

To prove :-

(1) If L is regular then $pre(L)$ is regular (prefix)

(2) If L is regular then $suffix(L)$ is regular

After proving we can make an ε transition for machine which accept $pre(L)$ to machine which accept $suffix(L)$

↓

The newly constructed machine is also a finite automaton (FA)

∴ FA equivalent to regular language — (1)

Thus | Dropout*(A) is regular if A is regular

Hence proved.

now we will prove how $pre(L)$ & $suffix(L)$ is regular when L is regular.

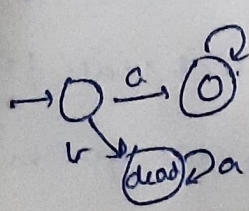
Draw a FA(finite automaton) for L.

Now for $pre(L)$. make all states final in path to final of FA.

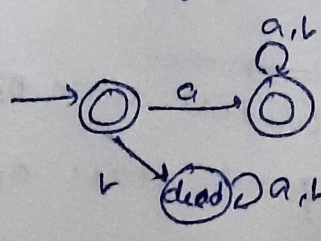
↳ This works because prefix of L means the first part of a string. so as it go from initial to final make all state final

from (1) Thus $pre(L)$ is regular

eg :- $L = a(a+b)^*$



$pre(L)$ ↳ also contain ε or start with a



make final of only those states which goto final.

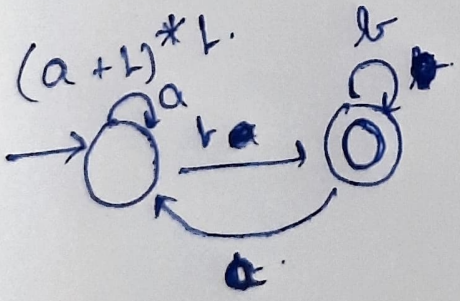
for suffix(L).

make an ε transition from initial to all state in the path of final state.

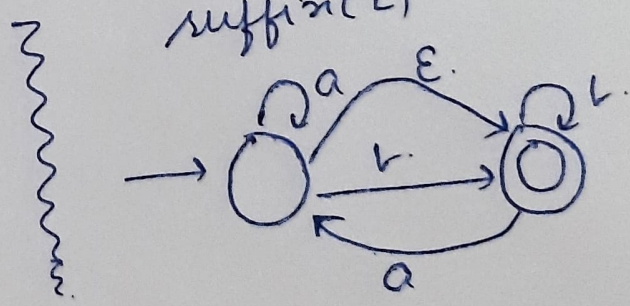
↳ This works because we only want the last part of string to be accepted

from ① suffix(L) is regular.

eg :- $L = (a+b)^*b$.



suffix(L)



(7)

$S \rightarrow a \mid b \mid aTa \mid bTb \mid aSb \mid bSa \mid aXb \mid aXa \mid bXb \mid bYa \mid bYb \mid aYa$

~~S → aTa | bTb | aSb | bSa | ab | ba | aXb | bYa~~

~~S → a | b | aTa | bTb~~

$T \rightarrow aT \mid bT \mid \varepsilon$

$X \rightarrow bXa \mid ba \longrightarrow b^n a^n \mid n \geq 1$

$Y \rightarrow aYb \mid ab \longrightarrow a^n b^n \mid n \geq 1.$

explanation :

$S \rightarrow a \mid b$ accept 1 length string ;

$T \rightarrow aT \mid bT \mid \varepsilon$
$S \rightarrow aTa \mid bTb$ accept all those strings which end with same symbol as starting one.

aXb is for a ba b
These type of string are not accepted by S. so we introduced a new variable X where it stores $b^n a^n$. kept in aXb.
& in Y stored $a^n b^n$.

kept in bYa.

lly aXa, bXb, bYb, aYa.

aSb used because

aaa ---- bbb
(can be anything except ε) → so not used.

like

↳ aaababbb (it is used to accept these kind of strings).

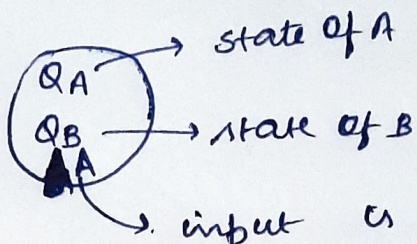
lly bSa

(10)

states

consider states such that it contain state of machine A, state of machine B and from which machine input is taken

eg :-



Σ (alphabets) :- it will be same ~~q_1, q_2~~.

Now new DFA will contain both the DFA's :- A, B given.

final state

by the end if both the states of A, B are final & input is read from a. then that is the final state because the last read from it.

from question

start state.

starting state may start states of A, B. & input read from should be a.

δ (transition function)

Q_A' → a state in DFA A

Q_B' → a state in DFA B

$$\delta((Q_A', Q_B', A), \Sigma) \rightarrow (\delta_A(Q_A', \Sigma), Q_B', B)$$

$$\delta((Q_A', Q_B', B), \Sigma) \rightarrow (\delta_B(Q_B', \Sigma), Q_A', A).$$

If input corresponding to A is read then next we will read from B.

& vice versa.

(8) Yes! context free.
We dont want palindromes.

so at some point it must ...a X b...

Or ...b X a...

same

at some point from starting & from ending they should be different

CFG :- $S \to asa \mid bSb \mid a\mathbf{X}b \mid b\mathbf{X}a$.

$\mathbf{X} \to a\mathbf{X} \mid b\mathbf{X} \mid \varepsilon$.

at some palindromic nature must be gone so used.

converting to chomsky Normal form :-

(1) Add a new start variable S'

$S' \to S$
$S \to asa \mid bSb \mid aXb \mid bXa$.
$X \to aX \mid bX \mid \varepsilon$.

(2) Remove ε transition. $(X \to \varepsilon)$

$S' \to S$
$S \to asa \mid bSb \mid ab \mid ba$.
$X \to a \mid b \mid aX \mid bX$.

(3) Remove short rules $(S' \to S)$

$S' \to asa \mid bSb \mid ab \mid ba$
$S \to asa \mid bSb \mid ab \mid ba$
$X \to a \mid b \mid aX \mid bX$

(4) Remove long rules.

4a) $S' \rightarrow VSV \mid USU \mid VU \mid UV$

$S \rightarrow VSV \mid USU \mid VU \mid UV$

$X \rightarrow V \mid U \mid VX \mid UX$

$V \rightarrow a.$

$U \rightarrow b.$

4b) $S' \rightarrow VW \mid UX \mid VU \mid UV$

$S \rightarrow VW \mid UX \mid VU \mid UV$

$X \rightarrow V \mid U \mid VX \mid UX$

$W \rightarrow SV$

$X \rightarrow SU$

$V \rightarrow a.$

$U \rightarrow b.$

Chomsky normal form of the grammar.