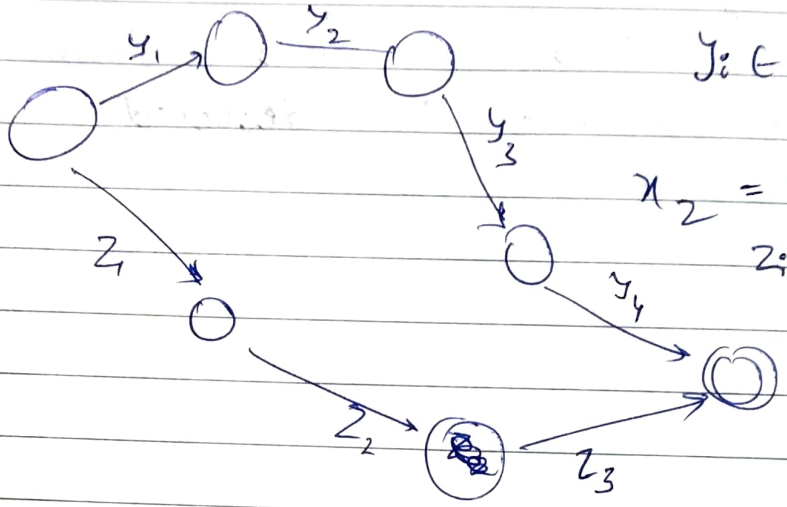
- If a language L is finite, then $\exists$ a Finite Automata (FA) that precisely accepts L.

$$L = \{x_1, x_2, x_3\}$$

$x_1 = y_1 y_2 y_3 y_4$

$y_i \in \Sigma$



$x_2 = z_1 z_2 z_3$

$z_i \in \Sigma$

We made a FA, NFA here, y_i may be equal to z_i, corresponding to a language L.

- Closed operations for regular languages

DFA, NFA, reg exp.

R_1, R_2 are regular languages.

i) union.

$R_1 \cup R_2$ is a reg language

ii) concatenation

$R_1 R_2$ is a reg. language

iii) Kleene closure

$\forall_L$ L is a reg language $\Rightarrow L^*$ is a reg language

$x = x_1 x_2 \ldots x_k \in L^+$

$x_1 \in L, \; x_2 \in L \ldots, x_k \in L$

iv) Positive closure

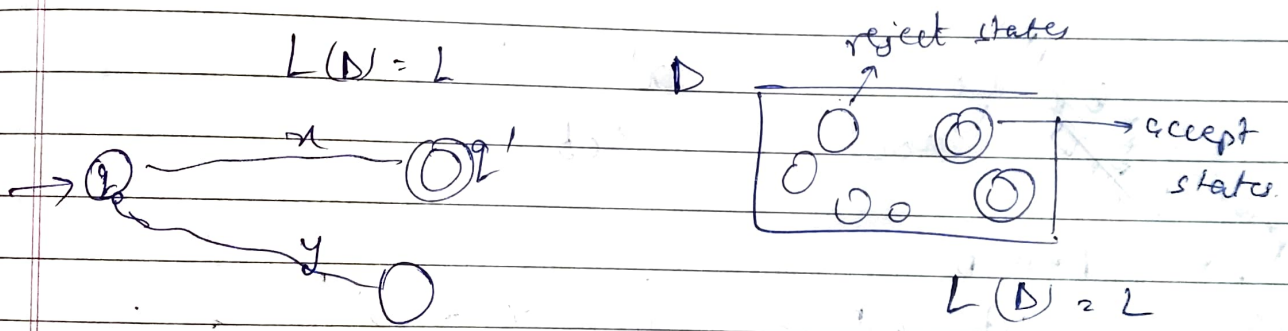
$\forall_L$ L is a reg. language $\Rightarrow L^+$ is a reg language

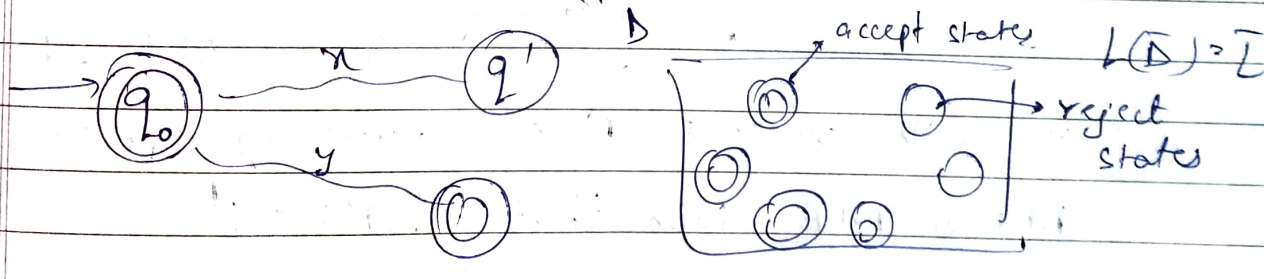
v) Compliment

L is reg language $\Rightarrow \bar{L}$ is also regular.

$\bar{L} = \Sigma^* - L$

Let L be represented by DFA D





$\bar{D}$ is also a DFA, as we only changed ~~start~~ accept and reject states.

Thus $L(\bar{D}) = \bar{L}$

for $\bar{L}$, we were able to construct a DFA, so $\bar{L}$ must be a regular language.

vi) Intersection

$\forall L_1, L_2$, L_1 is a reg language, L_2 is a reg language } $L_1 \cap L_2$ is a regular language

$$L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}}$$

$$= \overline{\bar{L_1} \cup \bar{L_2}}$$

As $\bar{L_1}, \bar{L_2}$ is reg. lang., union, compliment is closed operation

then $L_1 \cap L_2$ is also reg language

vii) reverse.

$\forall_L$ L is regular $\Rightarrow$ L^R is regular.

$x \in L$ ↳ abc

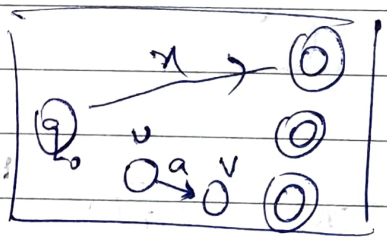
$x^R \in L^R$ ↳ cba

proving $x \in L(D^R)$

$x^R \in L(D) \Rightarrow$

DFA D

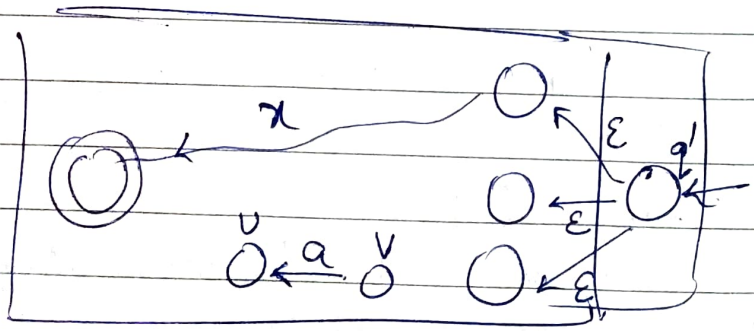
s.t $L(D) = L$



$\delta(q_0, x) = q_f$

$x \in L(D)$

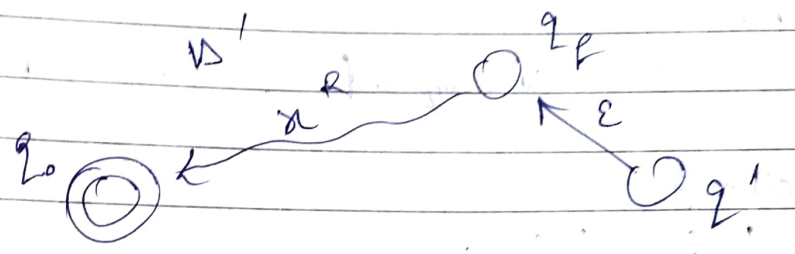
we will construct D^R ~~such~~ being reg expression, DFA or NFA. such that $L(D^R) = L^R$.



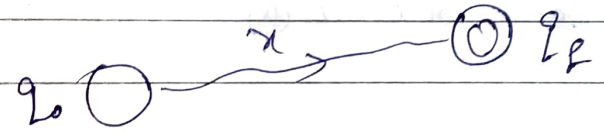
In D^R, we have one extra state q', having ε transitions to final states of D. And we will reverse all directions of all transitions from D, making q_0 as a final state then.

$$\delta(q', x^R) = q_0 \qquad x^R \in L(D^R)$$

Proving $x^R \in L(D^R) \Rightarrow x \in L(D)$



we construct D such that we reverse all transitions, remove ε transitions from q', remove q' state, and start from q_0 make q_f final state



Hence proved

We could come up with a D such that $x \in L(D)$

Closed operations for Context free Languages.

CFGs and PDAs

:) Union

CFG $G_1(V_1, T_1, P_1, S_1)$ ... $L(G_1)$

$G_2(V_2, T_2, P_2, S_2)$ ... $L(G_2)$

CFG

$G(V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1 | S_2\}, S)$

$L(G)$

claiming

$L(G) = L(G_1) \cup L(G_2)$

Part 1 $x \in L(G) \Rightarrow x \in L(G_1)$ or $x \in L(G_2)$.

Part 2 $x \in L(G_1)$ or $x \in L(G_2) \Rightarrow x \in L(G)$

Proving part 1

$S \Rightarrow S_1$

$\Rightarrow \ldots$

$\Rightarrow \ldots$

$\Rightarrow x$

Independent of T_2, V_2 so it produces x that belongs to $L(G_1)$

$S \Rightarrow S_2$

$\Rightarrow \ldots$

$\Rightarrow \ldots$

$\Rightarrow x$

Similarly this produces $x \in L(G_2)$

Part 1 proved

Proving Part 2.

Without loss of generality, we take

$x \in L(G_1)$, $S_1 \overset{*}{\Rightarrow} x$

By adding one more step to it

$S \Rightarrow S_1 \overset{*}{\Rightarrow} x$, we can say

$S \overset{*}{\Rightarrow} x$.

Hence Proved //

ii) Concatenation

$G_1 (V_1, T_1, P_1, S_1)$ .... L_1

$G_2 (V_2, T_2, P_2, S_2)$ .... L_2

Let $L = L_1 L_2$

For $G\{ V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$

~~We claim~~ ~~$x \in L$~~

Part 1 proof

$x \in L \Rightarrow x_1 \in L_1, x_2 \in L_2, x = x_1 x_2$

we have $S \overset{*}{\Rightarrow} x$

$S \Rightarrow S_1 S_2 \overset{*}{\Rightarrow} x$

$\exists$ $S_1 \overset{*}{\Rightarrow} x_1$ $S_2 \overset{*}{\Rightarrow} x_2$, so then

$x_1 \in L_1$ $x_2 \in L_2$, $x = x_1 x_2$

Part 2 proof

$x_1 \in L_1, x_2 \in L_2, x = x_1 x_2 \Rightarrow x \in L$

we have

so $S_1 \overset{*}{\Rightarrow} x_1 \quad S_2 \overset{*}{\Rightarrow} x_2, \quad x = x_1 x_2$

$\exists$ a production $S \Rightarrow S_1 S_2$

using it $S \Rightarrow S_1 S_2 \overset{*}{\Rightarrow} x_1 x_2 = x$

$S \overset{*}{\Rightarrow} x$

so $x \in L$

Hence Proved

iii) Kleene closure

$L \dots G_1(V_1, T_1, P_1, S_1)$

$L^* \dots G'(V_1 \cup \{S\}, T_1, P_1 \cup \{S \to S_1 S \mid \varepsilon\}, S)$

Part 1 proof

$x \in L^* \Rightarrow x_1 \in L, x_2 \in L, \dots x_r \in L, x = x_1 x_2 \dots x_r$

$S \Rightarrow S_1 S$

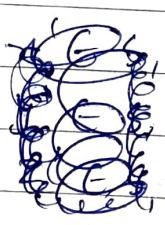
$\Rightarrow S_1 S_1 S$

$\Rightarrow S_1 S_1 S_1 S$

$\Rightarrow S_1 S_1 S_1$

$\Rightarrow x'y'z'$

this is for 3 extend it for r.



$S_1 \overset{*}{\Rightarrow} x'$

$S_1 \overset{*}{\Rightarrow} y'$

$S_1 \overset{*}{\Rightarrow} z'$

Part 2 proof HW.

iv) Positive closure

It also can be said closed. It is like a subset of Kleene closure.

- Pumping lemma for regular languages

It shows a language is not regular

$(\forall$ reg lang $L)(\exists p)(\forall z)(z \in L$ and $z \geq p)$

(implies

$\Rightarrow (\exists u, v, w)(z = uvw,\ |v| \geq 1,\ |uv| \leq p$ and $\forall i \geq 0\ uv^i w \in L)$

p : no. of states of a DFA that accepts L

(pumping length)

$z = uvw \rightarrow$ pumpable decomposition

$uv^0 w \in L$

$uv^1 w \in L$

$uv^2 w \in L$

$uv^2w \rightarrow$ pumping up

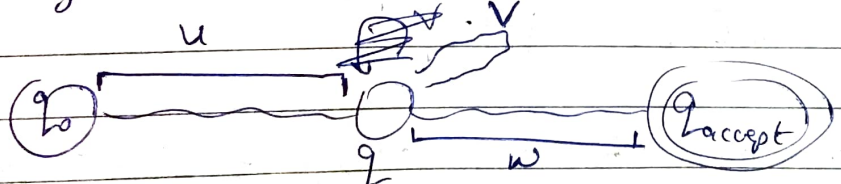
$uv^0w \rightarrow$ pumping down

~~If $z \geq p$ and~~

$z = p$, then distinct states possible are $p+1$

for $z \geq p$, and a DFA with p states, atleast one state must be repeated.

DFA may look like



As it is a DFA, $|v| \geq 1$

~~If we just~~
Even though $z \geq p$, but in first p input letters, we get our repeating state, u, v input strings. $uv \leq p$
so we are interested only in that.

$uvw \in L$
now the looping v if increased wouldn't cause any harm.
so $uv^0w \in L$, $uv^2w \in L$, ...

↑ understanding what statement told

How to show a lang is not regular

We will start, supposing language is regular, so there must accept any DFA to accept it, having p states.
Now for some string being sufficiently large, there doesn't exist pumpable decomposition.

eg. $L = \{a^i b^i \mid i \geq 1\}$ $\Sigma = \{a, b\}$

Suppose L is regular ⇒ ∃ a DFA that has p states.
$z \in L$ $|z| \geq p$

let our choosen string be $z = a^p b^p$

$z = uvw$

$|v| \geq 1$

$|uv| \leq p$

~~uvt~~

if $p = 5$

aaaaaa bbbbbb.

let this be u, v

This is v

$uv^2w = aaaaaaaa\,bbbbbb$

$= a^7 b^5$.

Here we have a contradicting string.

So L is not regular.

eg. $L = \{ w \mid w$ has an equal number of a's and b's $\}$

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

~~$a^k b^k$~~ $a^k b^k a^k b^k \in L$

$z = a^p b^p \in L$

Similar proof can be done as previous example.

eg. $L = \{ ww \mid w \in \{a, b\}^* \}$ $\Sigma = \{a, b\}$

Assume L is regular

$z = a^p b a^p b \in L$.

$|z| > p$

$z' = uv$

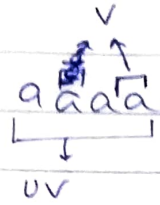
$z' = a^p$

$z = uvw$

$|v| \geq 1$

$|uv| \leq p$

let us choose v, uv



$uv^2w = a^{p+k} b a^p b.$

$k \geq 1$ so $uv^2w \notin L$

Thus L isn't regular.

- eg. $L = \{ a^{i^2} \mid i \geq 0 \}$

suppose L is regular

$z = a^{p^2}$ $\quad z \in L, \ |z| \geq p$

$z' = a^p$

$z = uvw$

$|uv| \leq p$ $\quad |z'| = p$

$p \geq |v| \geq 1$

$|uv^2w| = |uvw| + |v|$

$\leq p^2 + p$

$< (p+1)^2$

$|uv^2w| > |uvw|$

$> p^2$

there is no string of length p^2+p as it isn't perfect square

Thus we dont have a perfect pumping decomposi..

- eg $L = \{ a^i b^j \mid i > j \}$ $\Sigma = \{a, b\}$

taking $z = a^{p+1} . b^p$

v here, consisting only a's.

aaaaaa..bbb..b

uv

$|v| \geq 1$

$uv^2w = a^{p+1+\delta} b^p$

so for all pumping up cases, they are present in language L.

while for pumping down case :

$uv^0w = a^{p+1-\delta} b.$

$|\delta| \geq 1$

keeping $\delta = 1$

$uv^0w = a^p b^p$

but this is not present in language L.

Hence our assumption is wrong. L isn't regular.

- eg. $\{ w \mid w$ is a palindrome $\}$ $\Sigma = \{a, b\}$

$z = a^p b b a^p$

v comes from here, $|v| \geq 1$

So pumped up string is

$z = a^p b b a^p$

$z = uvw$

$|uv| \leq p$ $uv \leq p$

$uv^2w = a^{p+\delta} bb a^p$

$\notin L$

L isn't regular

- eg. $L' = \{ w \mid w$ is not a palindrome$\}$ $\Sigma = \{a, b\}$

Suppose L' is regular.

By closure properties, compliment, $\overline{L'}$ is also regular

So $\overline{L'} = L$

L must be regular.

But we saw in previous example, that it isn't regular.