Chomsky hierarchy

DFA, NFA } Finite automata

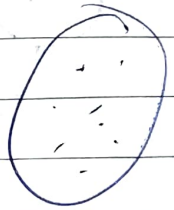
reg expressions represent same set of languages

eventually becomes

CFG, NPDA }

They represent the same set of languages

regular languages.



context free languages.

eventually becomes

* (less than equal to sign)

(less than)

DTM, NTM, unrestricted grammar, Partial recursive function, RAM model of computation

Its Equivalence is church turing thesis

Turing recognisable languages

↓

$x \in L$ machine halts in accept state

$x \notin L$ machine may or may not halt.

decidable languages is a set where machine halts. so it is a subset of turing recognisable languages

We will prove equivalence of CFG (context free grammar) and PDA (we refer NPDA as PDA)

- If a CFG G recognises a language L, then some PDA P accepts L

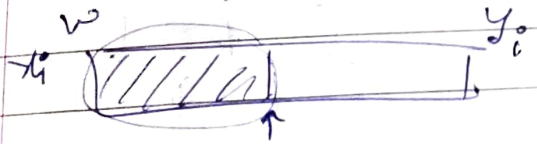
G starts from S. $G(V, T, P, S)$

If $x \in L$ then $\exists$ a derivation $S \overset{*}{\underset{G}{\Rightarrow}} x$

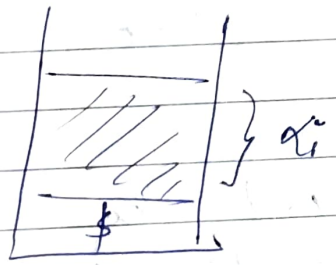
↳ consists only of terminals.

$(x = w)$ our older notation

We will have a w(input)



a stack



With help of some productions, popping elements off stack, we have formed x_i part of input. With α_i part we will form y_i part if possible.

If $S \Rightarrow Y_1 \Rightarrow Y_2 \Rightarrow Y_3 \Rightarrow \dots \Rightarrow Y_i \Rightarrow \dots x$

so

$Y_i = x_i . \alpha_i$

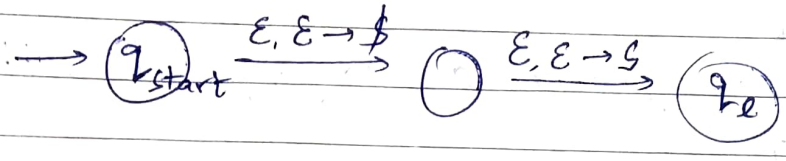
$(w/x) = x_i y_i$

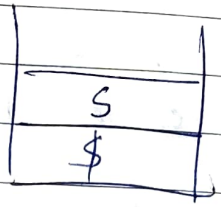
eg. $A \to abCD$

$S \Rightarrow abAd$

$\Rightarrow ababCD$

Suppose S → abcd | abAe

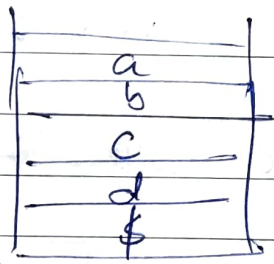




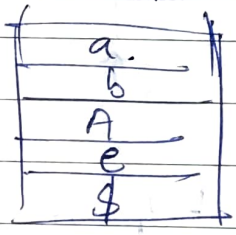
Now we will start transformations

As it is NPDA, different threads form up taking each possible production

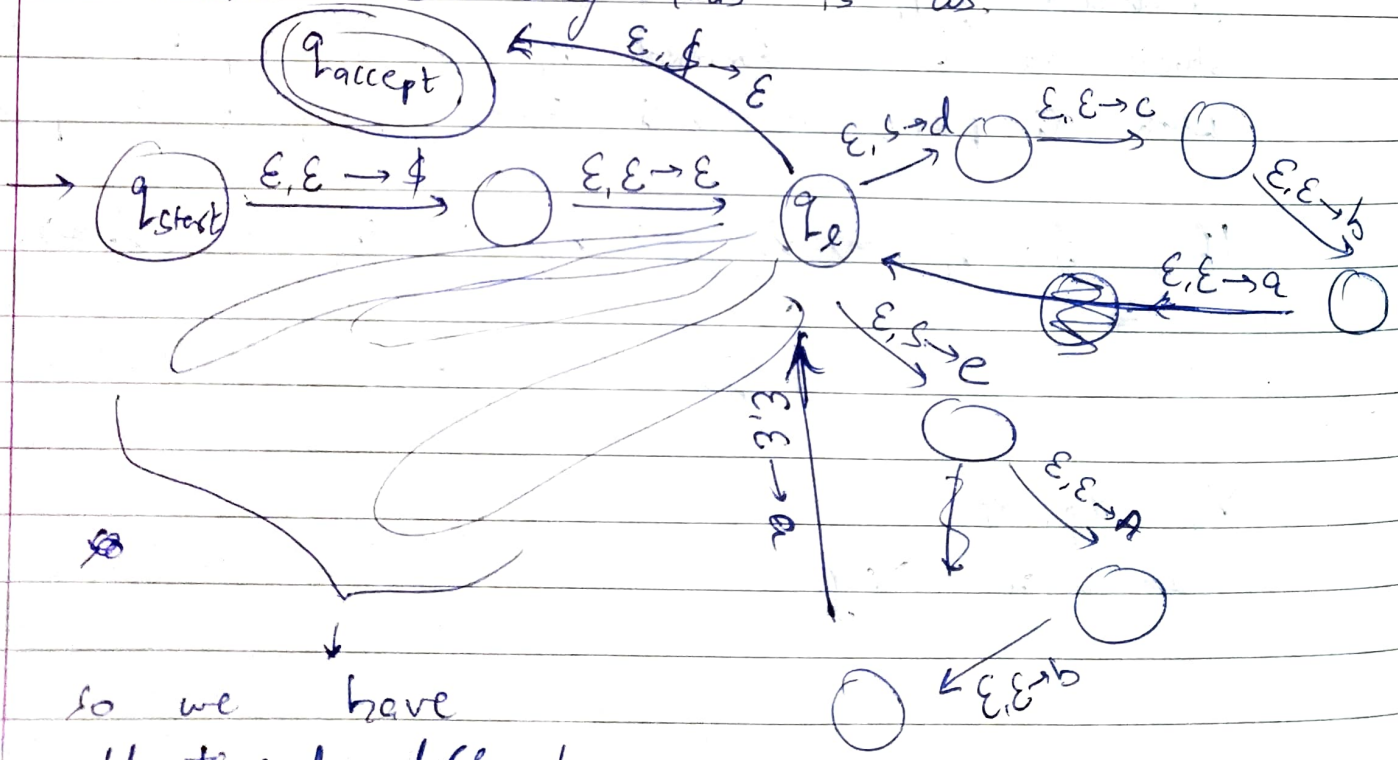
S → abcd then stack looks as



S → abAe the stack looks as

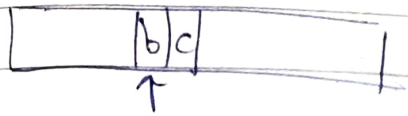


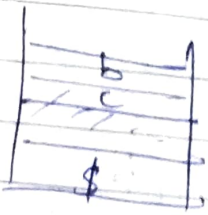
And the machine transforming this is as.



So we have all 7 such different transitions attached to q_e.

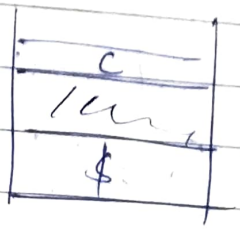
w



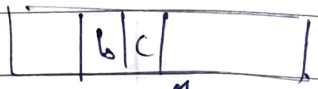


If top element is same, pop b and move head further.

w



↓ remove C.



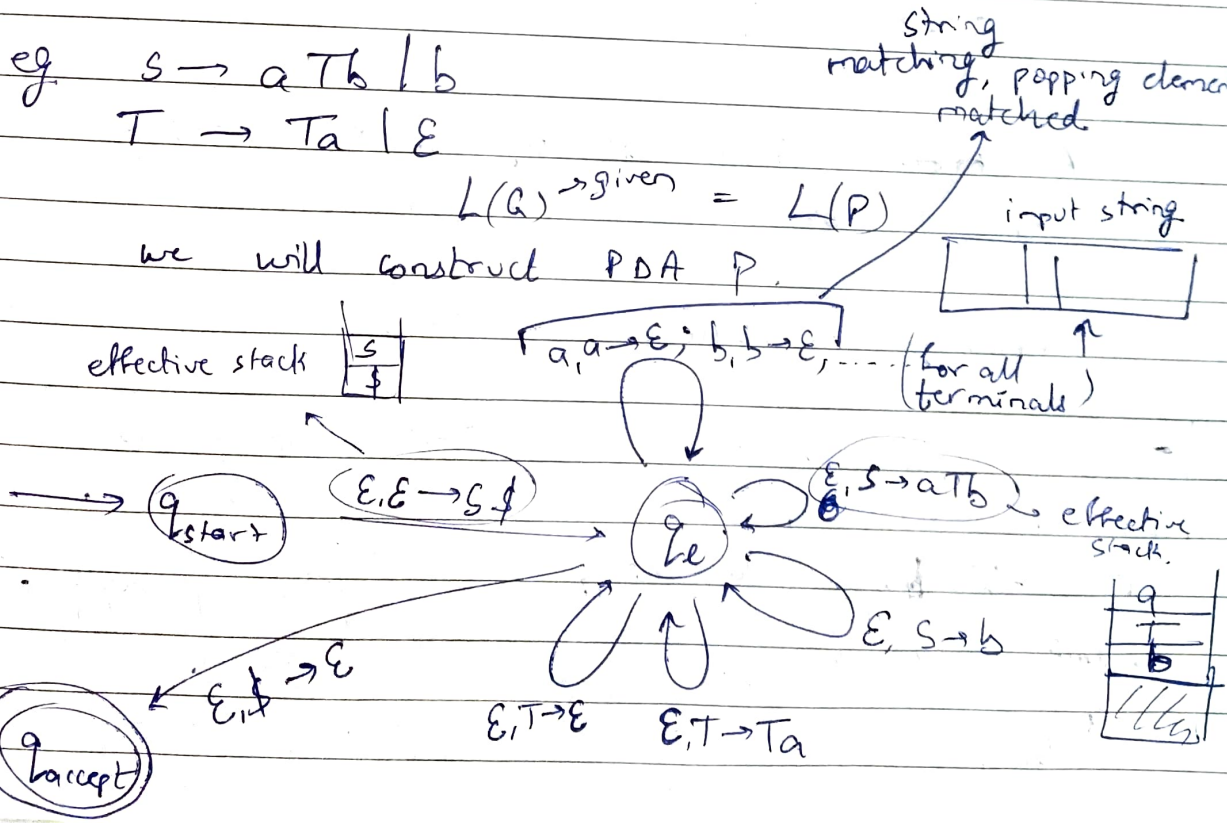
let's suppose, top of stack is non terminal, then again transistion takes place.

In this manner we compare element, transform them if string fully matches and stack is empty then PDA accepts w/x.

- eg. $S \rightarrow aTb \mid b$
  $T \rightarrow Ta \mid \varepsilon$

$L(G) \rightarrow$ given $= L(P)$

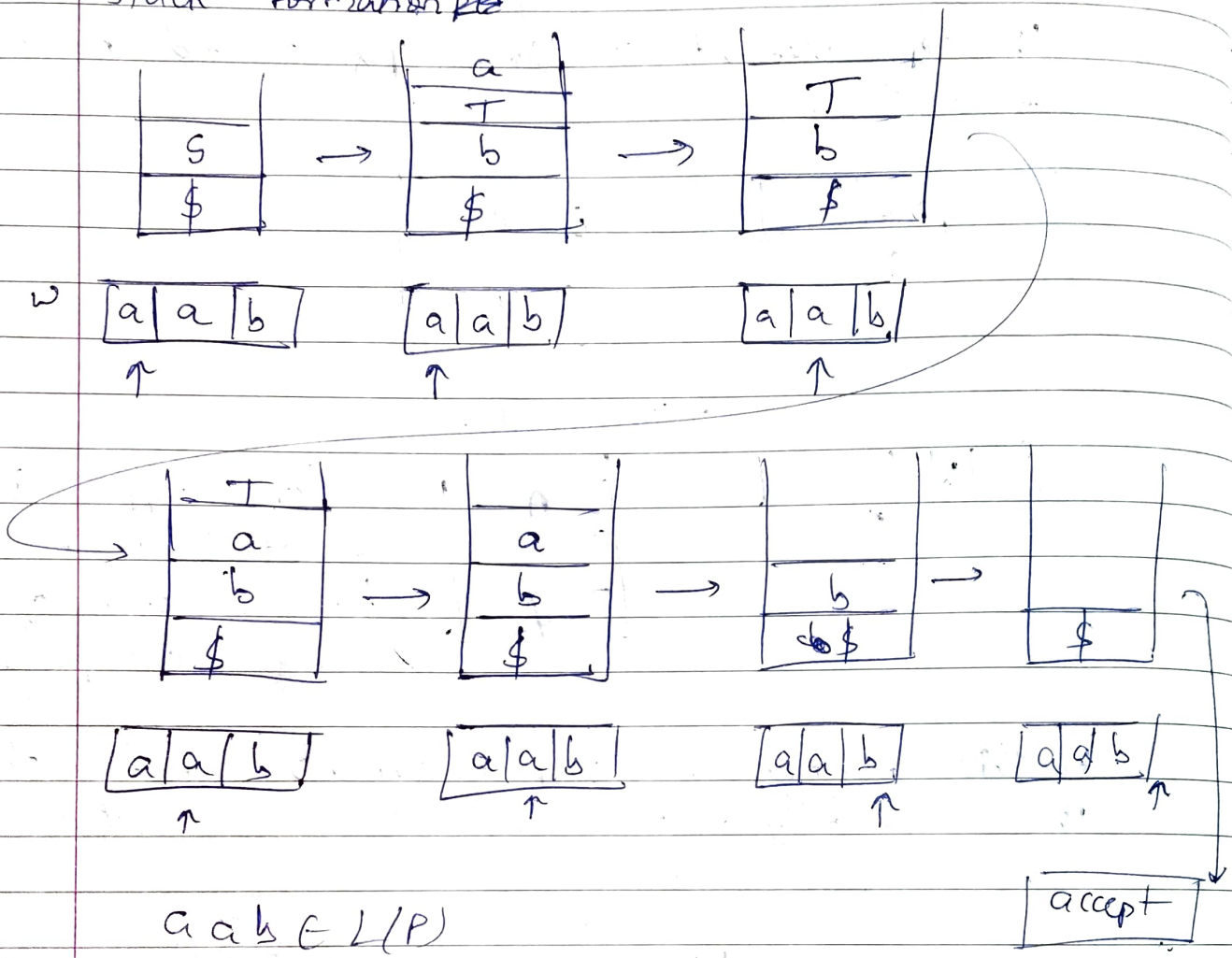
we will construct PDA P.



Date: YOUVA

$S \Rightarrow aTb \Rightarrow aTab \Rightarrow aaab.$

Stack formation



$aaab \in L(P)$

A terminal at top of the stack is removed only by comparision.

- Proving correctness of this

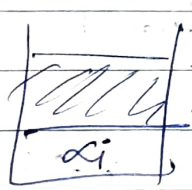
"$\Rightarrow$" this way proof

$S \Rightarrow \gamma_1 \Rightarrow \gamma_2 \ldots \rightarrow \gamma_i \Rightarrow \ldots \Rightarrow w$

Prove this by induction on no. of steps in derivation.

$w = x_i y_i$

$\gamma_i = x_i \alpha_i$

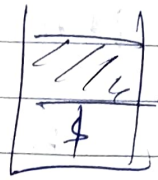


→ invariant

"⟸" this way proof

Given $w = x_i y_i$

$y_i = x_i \alpha_i$



By induction on the no. of moves of P we will prove

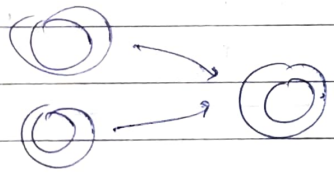
$$S \overset{*}{\Rightarrow}_G x_i \alpha_i$$

- If a PDA P recognises L, then some CFG (G) recognises L.

  s.t. $L(G) = L(P)$

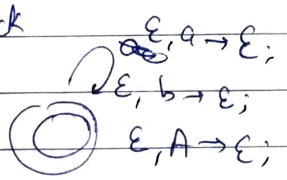
Assumptions → to make production of CFG easy.

i) Let P have one accept state ~~are~~ we make it in this way



ii) Empties stack at end of input

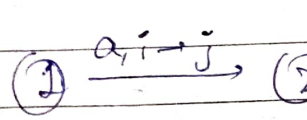
If we ~~reach~~ in final $ state and have ~~non~~ non empty stack, then empty stack



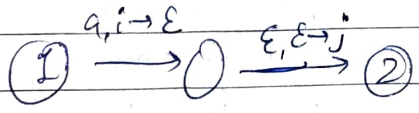
iii) In every transition it does one of the following

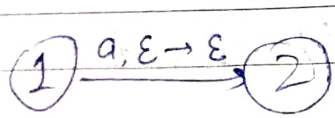
a) push

b) pop

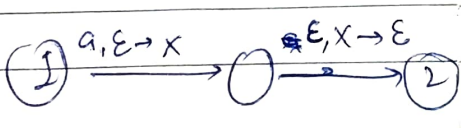


can be simulated in 2 steps





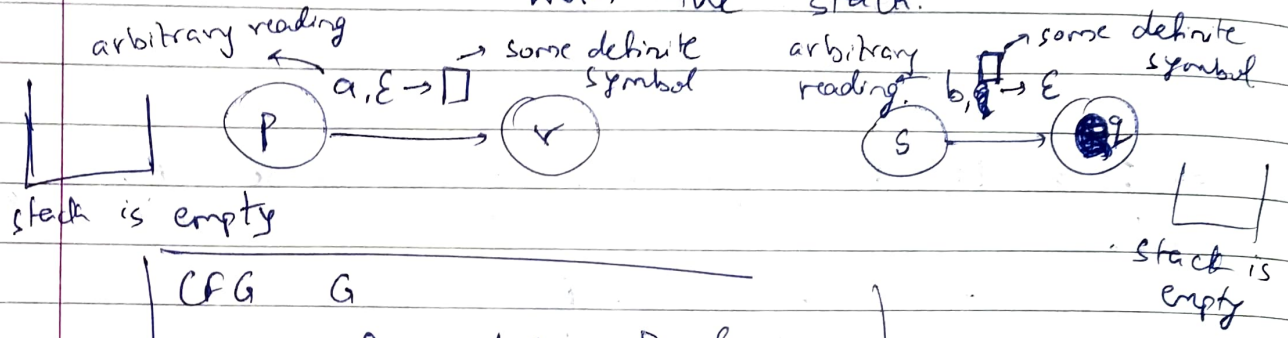
can be simulated as



PDA P

first step → You must always push a symbol on stack.

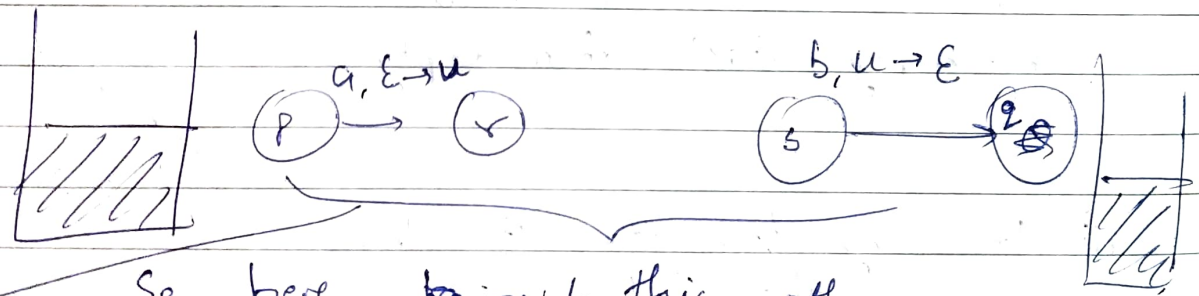
last step → You must have popped a symbol from the stack.



CFG G

A_{pq} takes P from state p to state q with stack empty with stack empty

Let p,q be some other states in between which push and pop same symbol, at same position

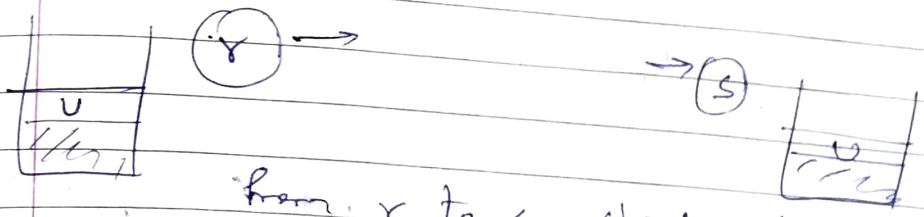


So here inside this path bottom of stack is never touched.

assume here stack does not get empty. we look at the this case at last.

$A_{pq} \rightarrow a \; \_\_ \; b.$

A_{pq} is of this format.



From r to s, stack above U is only accessed.

From r to s, first time it is empty above "U" at state s. (This is assumed, and for further states between r, s this follows).

A_{rs} ⇒ is taking all strings x that take PDA from state r with empty stack to state s with empty stack (refers to the part above U).

And in between it never gets empty (or it doesn't reach "U" this state).

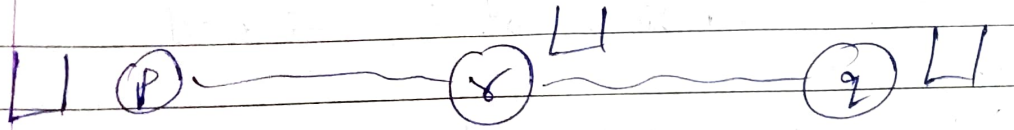
$A_{rs} = 0 \_\_ 0$

So there is definite production here.

$A_{pq} = a\, A_{rs}\, b$

↳ so on inductively.

If stack had been empty midway.



stack getting empty at a state r.

$A_{pq} \Rightarrow A_{pr} \cdot A_{rs}$

for $p, q, r, s \in Q$ $\quad (r,t) \in \delta(p,a,\varepsilon)$ and $(q,\varepsilon) \in \delta(s,b,t)$

tape symbol $\leftarrow t \in T$

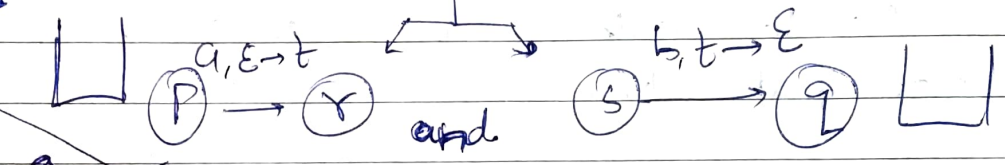
$\forall\ a, b \in \Sigma \cup \{\varepsilon\}$ ← input symbols (in string)

: NPDA transistion representation

$A_{pq} \rightarrow a A_{rs} b$ (if not empty in between)

~~$A_{pq} \rightarrow A_{pr} A_{rq}$ (if empty in between)~~

for this PDA



① st rule in ~~the~~ setup context

Final ~~for~~ Rules

$\forall\ a, b \in \Sigma \cup \{\varepsilon\}$

$A_{pq} \rightarrow a A_{rs} b$

② nd rule

$\forall\ p, q, r \in Q \quad A_{pq} \rightarrow A_{pr} \cdot A_{rq}$

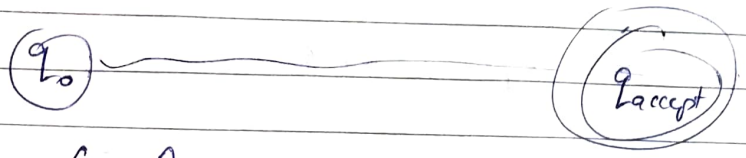
③ rd rule $\forall\ p \in Q \quad A_{pp} \rightarrow \varepsilon$ (0 number of steps)

started at p, remained at p, and stack is also empty

So 0 move in PDA and thus

0 ~~move~~ steps in production

4th rule



$S = A_{q_0 q_{accept}}$ $\quad A_{q_0 q_{accept}} \overset{*}{\Rightarrow}_G x$

We constructed the defination of A_{pq} for CFG G in starting. Now we will prove that A_{pq} is invariant in each step.

Proof

If A_{pq} generates x, then x can take PDA P from state p with empty stack to state q with empty stack.

$A_{pq} \overset{*}{\Rightarrow}_G x$ implies

Induction on the number of steps in the derivation of x

basis : no. of steps are 1.

$A_{pq} \Rightarrow x$

rule 3 rule 3 takes place here as in one step we go to string of terminals

so $x = \varepsilon$ and A_{pq} is A_{pp}

we stay at p itself.

IH: This follows for no. of steps $\leq k$

IS: no. of steps $= k+1$

$A_{pq} \Rightarrow \square \Rightarrow \cdots\cdots \Rightarrow x$

first step — k steps

first step has 2 possibilities

~~A_{pq}~~

i) $A_{pq} \rightarrow a A_{rs} b$

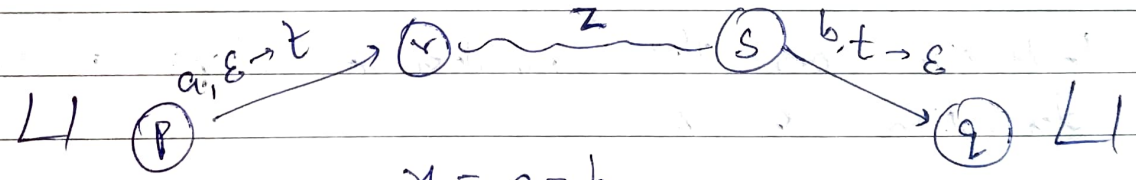
$A_{pq} \Rightarrow a A_{rs} b \Rightarrow \cdots\cdots \Rightarrow x = a \square b$

A_{rs} has $\leq k$ steps of derivation

by IH

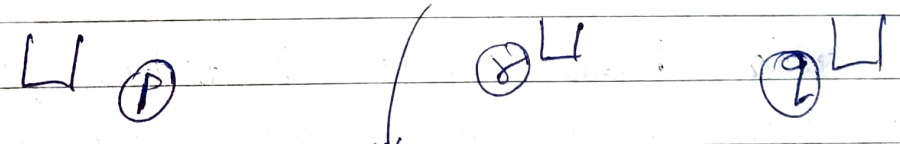
$A_{rs} \overset{\leq k}{\Rightarrow} z$

So our PDA could be like



$x = azb$

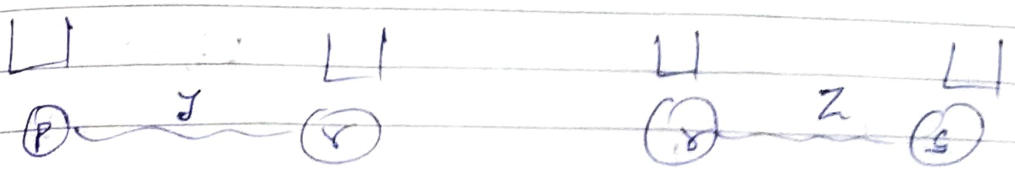
ii) $A_{pq} \Rightarrow A_{pr} A_{rs} \Rightarrow \cdots\cdots \Rightarrow x$

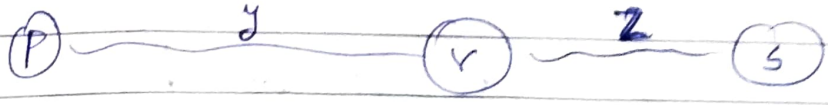


$\leq k$ no. of steps

By IH

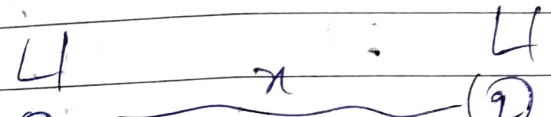
$A_{pr} \overset{\leq k}{\Rightarrow} y$ $\quad$ $A_{rs} \overset{\leq k}{\Rightarrow} z$





$x = yz$

- If x can bring PDA P from P with empty stake to q with empty state then A_{pq} generates x.



implies $A_{pq} \overset{*}{\underset{G}{\Rightarrow}} x$

induction on number of moves in the computation of P.

basis: no. of moves = 0.

so remain in same state.

$A_{pp} \to \varepsilon$

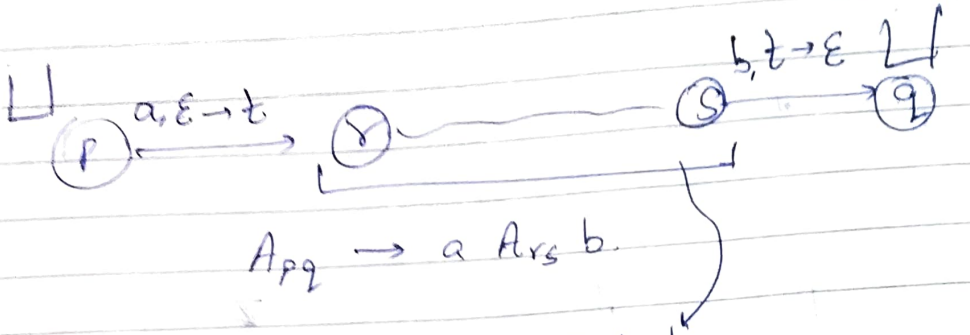
IH: this is true for $\leq k$ moves.

IS: no. of moves is $k+1$

2 cases

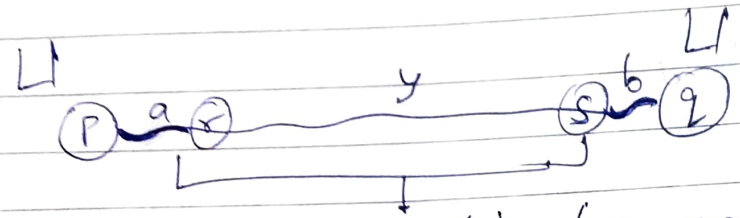
stack is empty in between → HW

stack is never empty in between → will do here



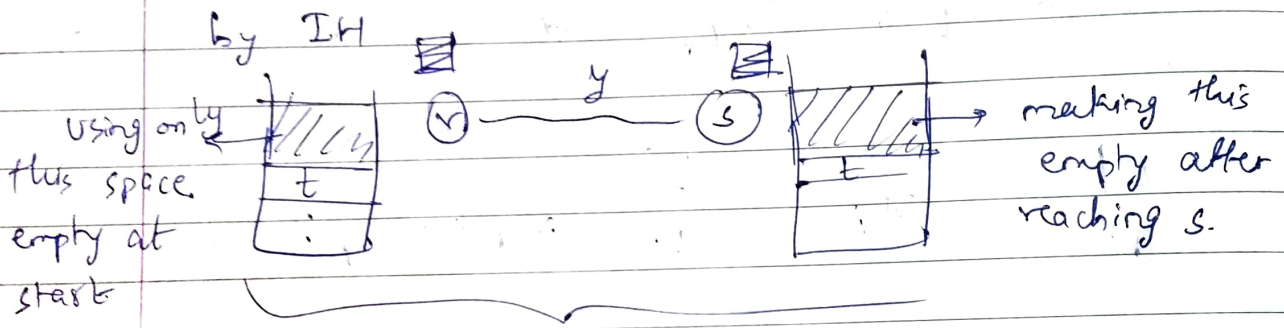
$A_{pq} \rightarrow a\ A_{rs}\ b$.

we use IH here.



$x = ayb$

moves $< k$ (we removed 2 moves ie q, b)



~~A_{pq}~~ A_{rs} takes care of r to s.

So finally we have $A_{pq} \rightarrow a\ A_{rs}\ b$ which ~~take~~ helps us go from p with empty stack to q with empty stack

$A_{pq} \rightarrow A_{pr}\ A_{rq}$ can be done by yourself.

Thus we are completed with the proof.
CFG's and ~~PDA~~ PDA's are equivalent and they form the set of context free languages.