Turing machine can be seen as language acceptor -

Let $M = (K, \Sigma, \delta, s, H)$ be TM such that

$H = \{h_a, h_s\}$ two halt states such that

(1) any halting config whose state config is h_a is called accepting configuration.

(2) Any halting config whose state config is h_s is called rejecting configuration.

we say M accepts $w \in (\Sigma - \{\#, \Delta\})$ if $(s, \Delta \# w)$ yields an accepting configuration

② if ~~w~~ $w \in L$ M accepts it

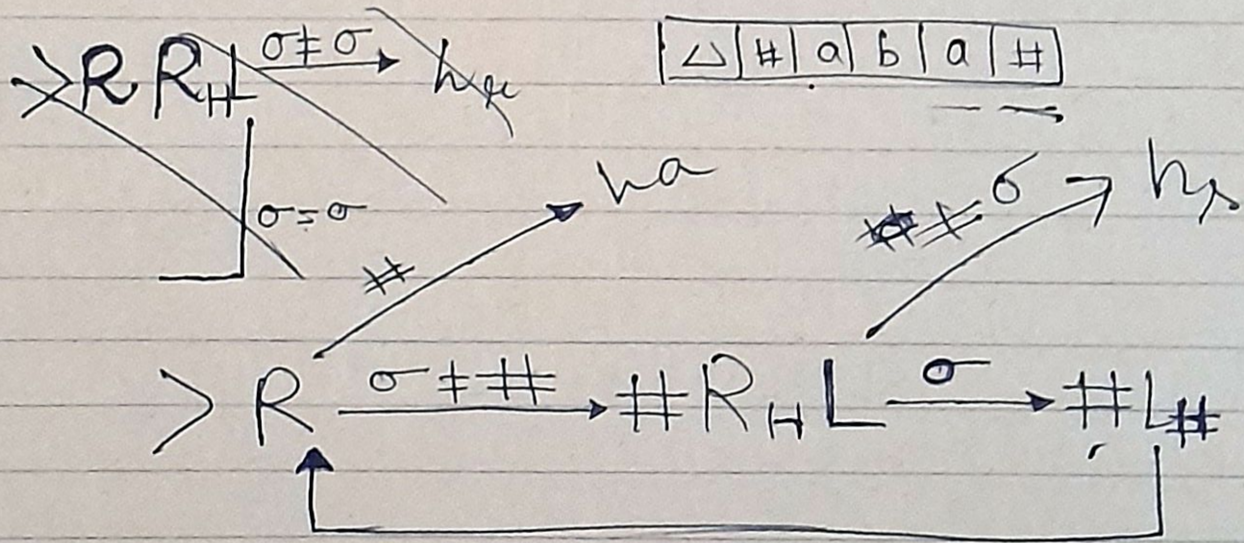
if $w \notin L$ M rejects it.

A language is called recursive if $\exists$ TM if it halts at (h_a)

Q- $L = \{ww^R \mid w \in \Sigma^*\}$

| △ | # | w | # | # | # | → by def.

Sol-



~~Note here we~~

Here notation as follows when we read σ we move ~~to~~ now that is to remembered and can be u…

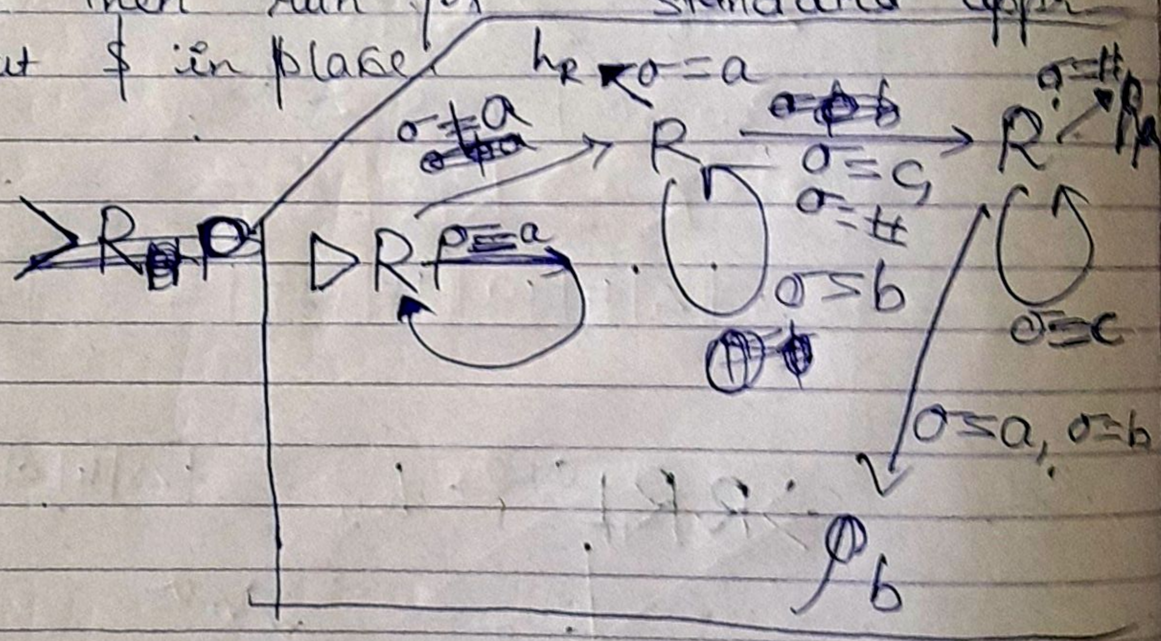
HW ques-

① $L = \{a^n b^n c^n \mid n \geq 0\}$

② $L = \{ww \mid w \in (a+b)^*\}$

Idea: check if string is in form $a^i b^j c^k$ or not and then run for standard appr to put \$ in place

ⓐ →



Two stacks with

Date
Page

Context free languages

Language generator – Device begins when given start signal to construct st[ring]. It has limited set of rules, eventually process halts & string is given. Language defined by device is set of all st[rings] it produces.

Regular ~~languages~~ exp. can be seen as language g[enerator].

(why context free)? when applying rule on symbol (non term[inal]) it not matters what outside ~~bo~~ sy[mbol] (context) is.

Context free gramm[ar] G is quad (V, Σ, R, S)

V = alphabet, Σ set of terminals
R = the set of rules in subset $(V-\Sigma) \times V^*$ and
S = Start symbol $(V-\Sigma)$

(R =) For any $A \in (V-\Sigma)$ & $u \in V^*$ we write $A \rightarrow_G u$ when $(A, u) \in R$.

A language L is called context free language if $L = L(G)$ for some context free grammar.

Note: For any $u, v \in V^*$ we write $u \Rightarrow_G v$ if and only if $x, y \in V^*$ & $A \in (V - \Sigma)$ such that

$$u = xAy$$
$$v = xv'y$$
$$A \longrightarrow_G v'$$

($\Rightarrow_G$ is reflexive)

$$L(G) = \{ \omega \in \Sigma^* : S \Rightarrow_G^* \omega \}$$

Derivation of G → we call the seq in form

$\omega_0 \Rightarrow \omega_1 \Rightarrow \ldots \Rightarrow \omega_n$ a deriv of G in ω_n from ω_0.

→ All reg languages are context free → opposite is not true.

Date
Page

eg-

$S \rightarrow$ ~~EE~~ $\quad S \rightarrow SS$

$S \rightarrow (S)$

(6) Class of context free languages is closed under union, concatenation & Kleene star.

Proof- Regular languages are context free. (by direct construct)

· A finite deterministic automata
$M = (K, \Sigma, \delta, s, F)$ same language is generated by grammar
$G(M) = (V, \Sigma, R, S)$
here
$V = K \cup \Sigma \quad S = s$
$\& \; R = \{q \rightarrow ap : \delta(q, a) = p\}$
$\cup \{q \rightarrow e : q \in F\}$

Parse trees -

Points are called nodes each node carries a label that is symbol in V.

Top most node = Root
Nodes along bottom = Leaves.

All leaves are terminals

To get derived string we concat label from left to right which is yield of tree.

Formally -

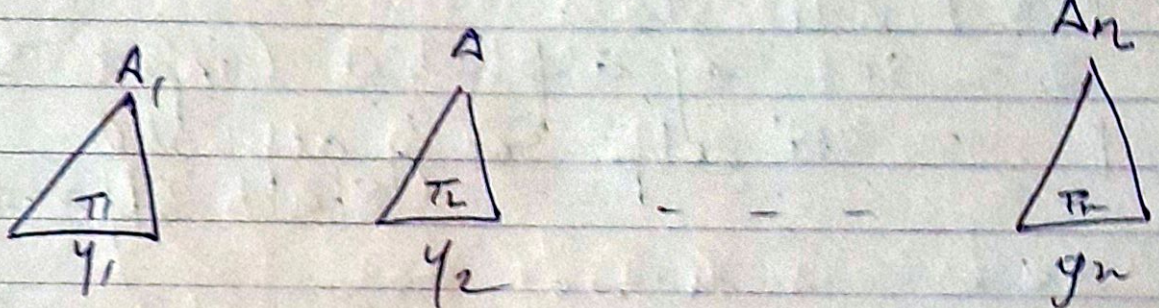
$G = (V, \Sigma, R, S)$ we define parse trees root leaves & yield a

① o a

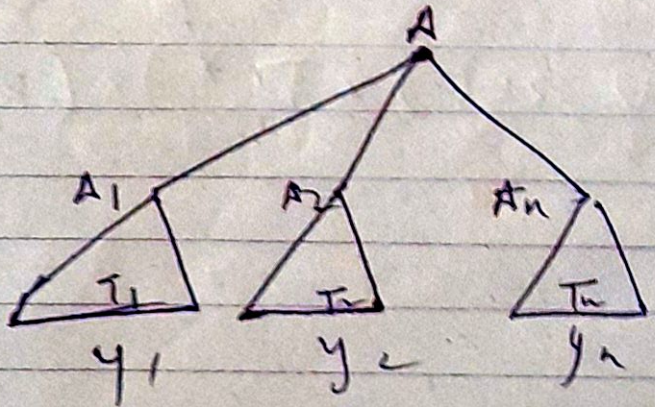
Root, leave = a
yield = a

② If $A \to e$ is rule then

is part of

③ If 

are parse trees with root label $A_1 \cdots A_n$ & yield $y_1 \cdots y_n$ & $A \to A_1 \cdots A_n$ is rule in R then



Root → A. Yield = $(y_1 \cdots y_n)$