CS-303

End sem

Name : P.V. Sriram

Roll No. : 1801CS37

1) Given,

$\Sigma = \{ \text{double}, +, *, /, (,) \}$ and

$Comp = \{ w \in \Sigma^* \mid w$ is a legal arithematic operation

RTP: If we can find a context free grammar which could generate comp, then we can say that there exists a pushdown Automata which can accept comp language

This is because of the theorem,

Context-Free languages = Languages Accepted by PDA.

Context free grammar :

$S \longrightarrow T$

$T \longrightarrow VAV \mid B_1 T B_2 \mid TAT \mid V$

$V \rightarrow$ 'double'

$A \rightarrow + \mid * \mid /$

$B_1 \rightarrow ($

$B_2 \longrightarrow)$

(i) double + double * double

(ii) ((double / double) * (double + double)) + (double)

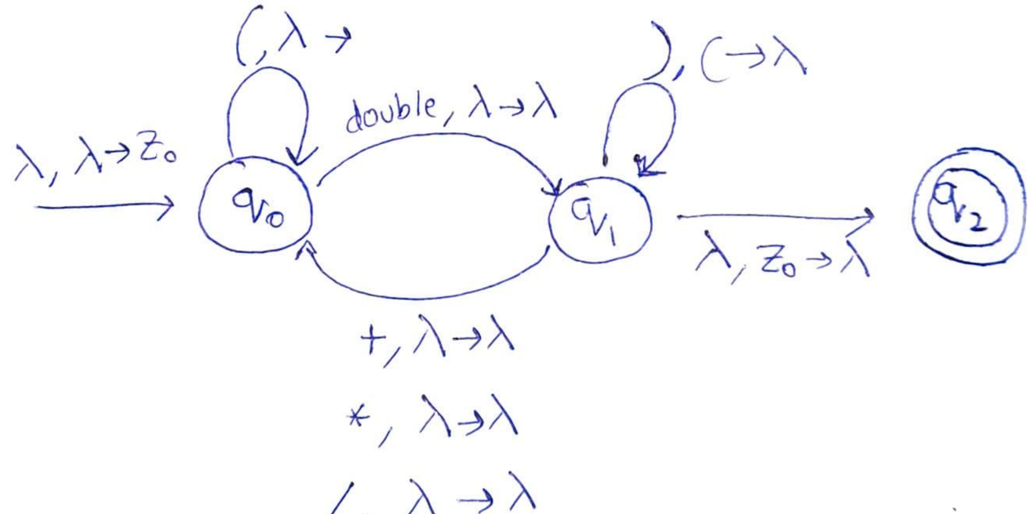
For (ii)

$S \to T \to B_1 T B_2 \to B_1 T A T B_2 \longrightarrow B_1 B_1 T B_2 A B_1 T B_2 B_2$

~~T~~ $\Rightarrow$ ((double/double) * (double + double)) +(double)

∴ Legal arithematic operations are generated from the language.

<u>Push down Automata</u>:



∴ We can see that there are both CFG, PDA for Comp language.

2) Required to find a context free grammar which generates the language $C_g = \{x \# y \mid |x| \neq |y|\}$ from the alphabet $\Sigma = \{0, 1, \#\}$.

Essentially, we have to generate two unequally sized strings on either side of the segregator '#'.

This can be done by initially taking strings x, y such that $|x| = |y|$. And then we add extra characters either on the left or right.

CFG:

$S \to TA$ } Responsible for the extra part

$S \to BT$

$A \to TA \mid X$

$B \to BT \mid X$

$X \to TXT \mid \#$ } Responsible for equal part

$T \to 0 \mid 1$ } Terminals

If we want the left part ('A') to be extra, we start with $S \to TA$ and get enough number of T's as necessary, and then use $A \to X$. X then gives rise to TXT which from then on creates equal number of characters on both sides.

Similarly to generate extra characters on right side we use $S \to BT$ in the starting step and repeat the same.

3) Required to convert Context-free grammar

$S \to 1SVO$

$S \to 1$

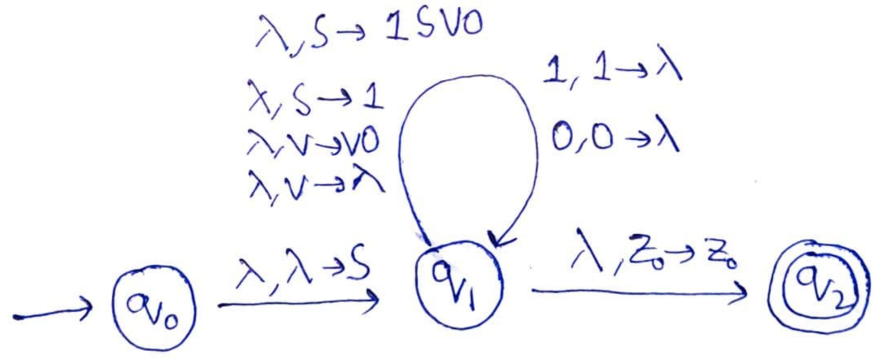
$V \to VO$

$V \to \lambda$

Into Push down Automata.

We can convert a CFG into PDA by adding productions and ~~transitions~~ terminals of the form $A \to w$ and a respectively into transitions $\lambda, A \to w$ and $a, a \to \lambda$

Using these rules, we get following PDA:



we add each production, for each λ - transition, to our Pda as it simulates the left most derivations.

4)

a) $L = \{0^{2L} w \mid w \in \{0,1\}^*, |w| = L\}$

Pumping Lemma :

We first assume L is regular. And therefore has to satisfy the lemma.

Let the pumping length be 'p'. Consider s ∈ L such that $|s| \geq p$.

ie) Let $s = 0^{2P} w$ where w starts with 1 and $|w| = p$

According to the lemma if $s = xyz$ then $s' = xy^i z$ ∀ $i \geq 0$ should belong to L aswell.

$|S| = 2P + P = 3P \geq P$ (critical length)

∵ L is regular, ∃ $x, y, z \in \{0,1\}^*$ such that

$xyz = 0^{2P} w$ and (i) for each $i \geq 0$ $xy^i z \in A$

(ii) $|y| > 0$

(iii) $|xy| \leq P$

Consider $y = 0^k$ ∴ $k > 0$, $|x| + k \leq P$

$$\underbrace{0000}_{x}\underbrace{---}_{y}\underbrace{000\ w}_{z}$$

For $i = 0$, $xy^i z = 0^{2P-k} w \notin L$ as $2P - k < 2|w| = 2P$

∴ Contradiction ⟹ L is not regular

4)

b) $L = \{0^n 1^m 2^k \mid k \neq n+m\}$

Pumping Lemma

Assume L is regular. Then pumping lemma must hold on L. Let m be the critical length

Then, $\forall s \in L$ such that $|s| \geq m$ can be divided into 3 pieces $x, y, z \in \{0,1,2\}^*$

such that, (i) $\forall i \geq 0$, $xy^iz \in L$

(ii) $|y| > 0$

(iii) $|xy| \leq m$.

Let us choose $a^{m!} b^{m!} c^{(m+1)!}$

Since xy cannot exceed m, y must be in form a^p for some $p > 0$.

By pumping lemma, any string of form xy^iz must belong to L.

Now, xy^iz and $y = a^q$, $q > 0$ and $|xy| \leq p$

$$xy^iz = 0^{p! + (i-1)q} 1^{p!} 2^{(p+1)!} \text{ for } i \geq 0$$

Clearly $p! + (i-1)q + p! = (p+1)!$

when $i = 1 + \frac{(p+1)! - 2p!}{q} \Rightarrow 1 + \frac{(p-1)p!}{q}$

Also $q \leq p$, hence there is a chance that q divides $p!$. Hence, there exist i such that $k = n+m$. Hence

$\boxed{\text{Not regular}}$

5) RTP: Set of deterministic context free languages is a proper subset of the class of context free languages.

For this proof, if we are able to find atleast one CFL which is not deterministic, then the proof is done.

Now,

Let $L = \{a^n b^n\} \cup \{a^n b^{2n}\}, n \geq 0$

Now, L has its own context free grammar

$$S \rightarrow S_1 \mid S_2$$
$$S_1 \rightarrow aS_1b \mid \lambda$$
$$S_2 \rightarrow aS_2bb \mid \lambda$$

$\therefore$ L is context free language.

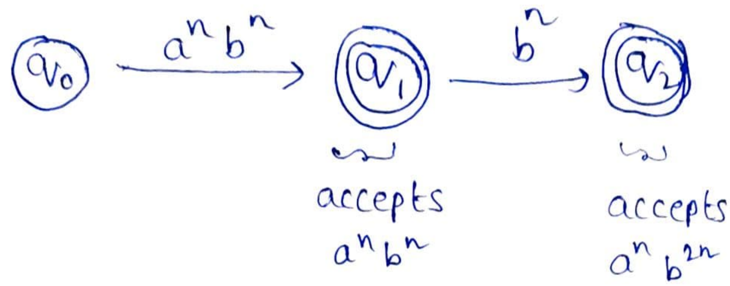
Now,

we need to prove L is non-deterministic. This can be done through contradiction.

Assume L is deterministic context free.

$\Rightarrow$ There exists a DPDA 'M' that accepts L

M:

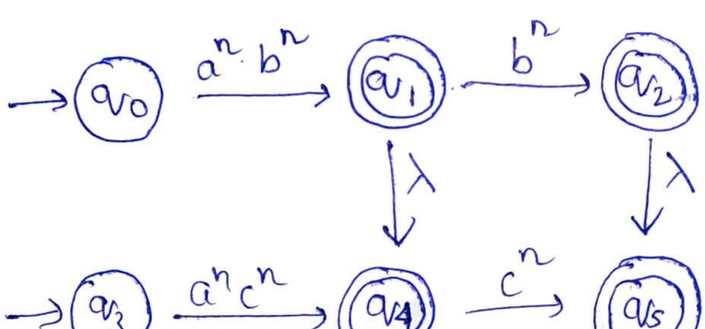


Now, consider $\{a^n b^n c^n\}$ which is non context free.

Ideally, $L \cup \{a^n b^n c^n\}$ should also be non context free if assumption is true.

$$L \cup \{a^n b^n c^n\} = \{a^n b^n\} \cup \{a^n b^{2n}\} \cup \{a^n b^n c^n\}$$

PDA:



} by replacing c in b

We can see this PDA accepts $L \cup \{a^n b^n c^n\}$ but this shouldn't have been the case. i.e) M shouldn't have been possible

∴ L is not deterministic but it is context free language

Hence proved

6) Required to design a Turing Machine for the language

$$L = \{0^n 1^{cn} : n, c \in \mathbb{N}\}$$

We can extend the idea of turing machine from $L' = \{a^n b^n, n \geq 1\}$

to design turing machine for L.

for $L' = \{a^n b^n, n \geq 1\}$, The algorithm is as follows.

(i) Match all a's with b's

(ii) Repeat:

Replace leftmost a with x

Find leftmost b and replace it with y

Until:

There are no more a's or b's

If there is a remaining a or b reject

At the end of the algorithm all a's are replaced by x and b's are replaced by y.

Similarly, for our language,

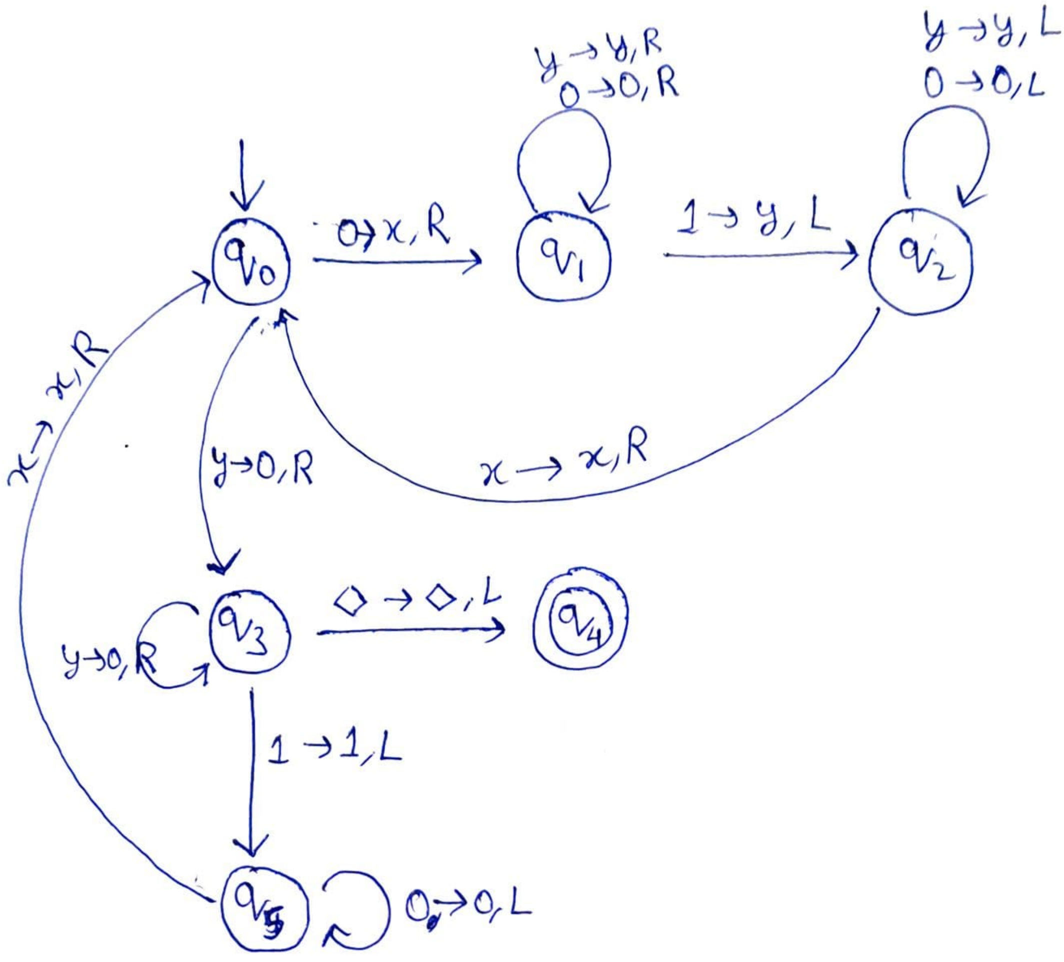
$0^n 1^n 1^n \ldots$ c times

If we perform the above algorithm once, result is $x^n y^n 1^n \ldots$ (c−1) times. If we again replace all y's by 0s and for $x^n 0^n 1^n$ and repeat c times same algorithm, we can identify 'L' language strings.

(arrow head) t_i

Turing machine :

$L = \{0^n 1^{cn} : n, c \in \mathbb{N}\}$



7) Consider a turing machine M.

Corresponding language:

Regular TM = { <M> : M is a turing machine that accepts a regular language.

We need to prove that $regular_{TM}$ is undecidable

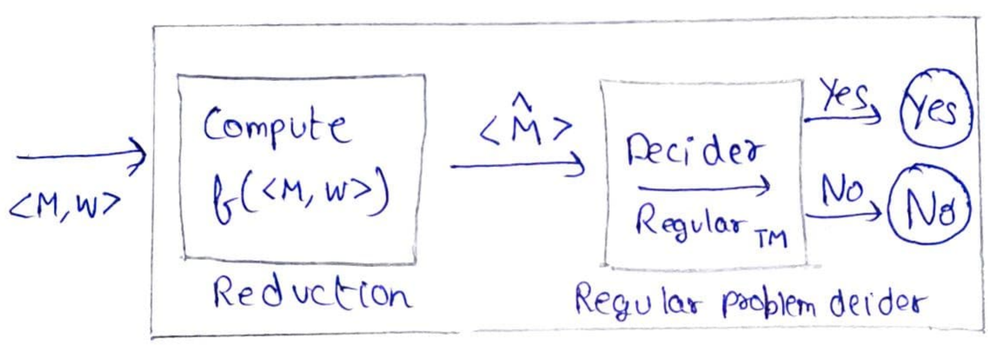
Reduce

A_{TM} (memlership problem)

to

$Regular_{TM}$

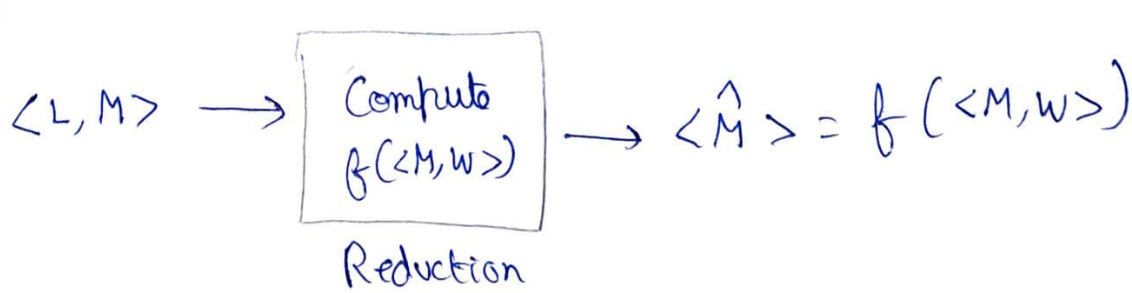
Membership Problem



Given the reduction,

If $Regular_{TM}$ is decidable, then A_{TM} is decidable

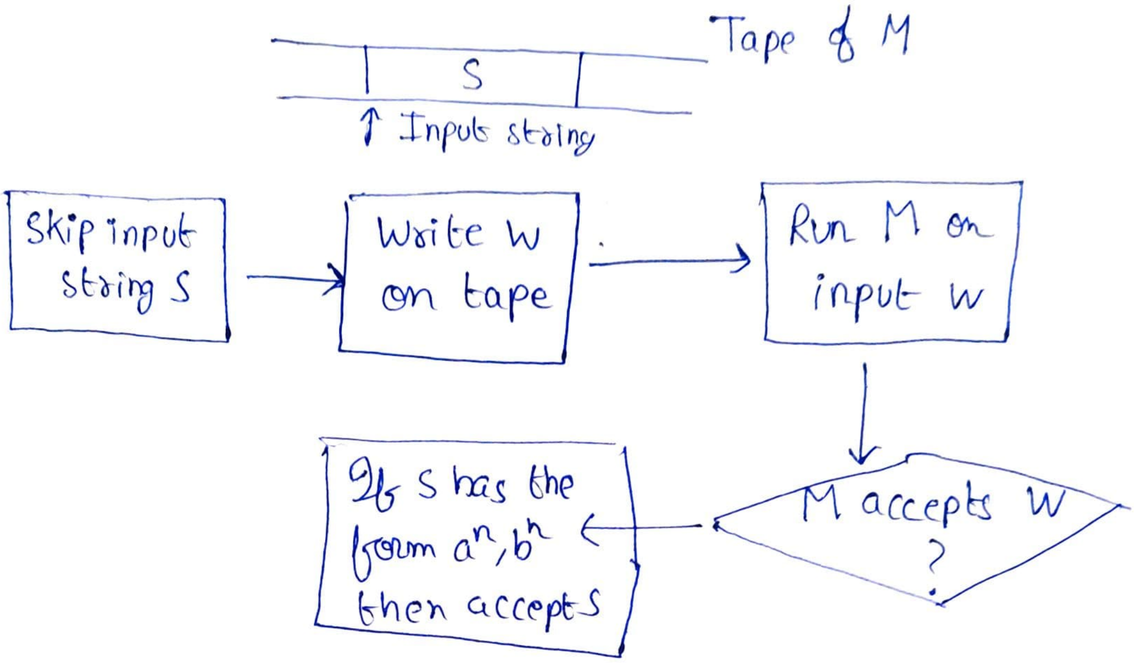
This is contradiction, since A_{TM} is undecidable. We only need to build the reduction



So,

$$<M, W> \in A_{TM} \iff <\hat{M}> \in \overline{Regular}_{TM}$$

Control $<\hat{M}>$ from $<M, W>$



$\Rightarrow$ M accepts $W \Rightarrow L(\hat{M}) \Rightarrow \{a^n b^n : n \geq 0\}$ (not regular)

M does not accept $W \Rightarrow L(\hat{M}) = \emptyset =$ (regular)

$\therefore$ M accepts $W \iff L(\hat{M})$ is not regular

Equivalently,

$$<M, W> \in A_{TM} \iff <\hat{M}> \in \overline{Regular}_{TM}$$

So, Turing Machine accepts a regular language is undecidable

Hence proved

8) Required to convert the following CFG into Chomsky Normal Form.

$S \rightarrow Aba,$

$A \rightarrow aab,$

$B \rightarrow Ac$

In CNF, only productions allowed are of form,

$A \rightarrow BC$ ← variables

or

$A \rightarrow a$ ← Terminals

Step: Add variables

$S \rightarrow Aba$
$A \rightarrow aab$
$B \rightarrow Ac$

$\xrightarrow[\text{for Terminals}]{\text{Using variables}}$

$S \rightarrow AT_bT_a$
$A \rightarrow T_aT_aT_b$
$B \rightarrow AT_c$
$T_a \rightarrow a$
$T_b \rightarrow b$
$T_c \rightarrow c$

Step: Reduction if RHS ≥ 3

$S \rightarrow AT_bT_a$
$A \rightarrow T_a T_a T_b$
$B \rightarrow AT_c$
$T_a \rightarrow a$
$T_b \rightarrow b$
$T_c \rightarrow c$

$\longrightarrow$

$\boxed{\begin{array}{l} S \rightarrow AV_1 \\ A \rightarrow T_aV_2 \\ B \rightarrow AT_c \\ V_1 \rightarrow T_bT_a \\ V_2 \rightarrow T_aT_b \\ T_a \rightarrow a \\ T_b \rightarrow b \\ T_c \rightarrow c \end{array}}$

7) RTP: $L = \{0^n 1^j : n = j^2\}$ is not Context-Free

Pumping Lemma

Let us assume L is a context free language

Let m be the pumping length, and let $s \in L$, $|s| > m$

i.e) Let $s = 0^{m^2} 1^m$

Now if $s = uvxyz$ then $uv^ixy^iz \in L \ \forall\ i \geq 0$,

$|vxy| \leq m$, $|vy| \geq 1$

We examine various cases for possible locations of vxy

Case 1: vxy in m^2

$$\underbrace{\overbrace{0}^{u}\overbrace{0}^{v}\overbrace{\cdots}^{x}\overbrace{0}^{y}}_{m^2}\ \overbrace{11\cdots1}^{m}$$

$v = 0^{k_1}$ $\quad y = 0^{k_2}$ $\quad \Rightarrow uv^2xy^2z = 0^{m^2+k_1+k_2} 1^m$

$m^2 + k_1 + k_2 \neq m^2 \Rightarrow$ Contradiction

Case 2: vxy in m

$$\overbrace{00\cdots0}^{m^2}\ \overbrace{\underbrace{1}_{v}\underbrace{1\cdots}_{x}\underbrace{\cdots1}_{y}}^{m}$$

$1 \leq k_1 + k_2 \leq m$

$v = 1^{k_1}$ $\quad y = 1^{k_2}$

$uv^2xy^2z = 0^{m^2} 1^{m+k_1+k_2}$

$m^2 \neq (m + k_1 + k_2)^2$

$\Rightarrow$ Contradiction

Case 3: vxy in both m^2, m

$$\underbrace{00\cdots0}_{v}\underbrace{11}_{x}\cdots\underbrace{1}_{y}$$

$1 \le k_1 + k_2 \le m$

$v = 0^{k_1} \quad y = 1^{k_2}$

$i = 0 \Rightarrow uv^0xy^0z = 0^{m^2-k_1} 1^{m-k_2}$

$m^2 - k_1$	$(m-k_2)^2$
$m^2 - k_1$	$\Rightarrow m^2 + k_2^2 - 2mk_2$

$(m-k_2)^2 \le (m-1)^2$

$= m^2 - 2m + 1$

$< m^2 - k_1 \quad \Rightarrow m^2 - k_1 \ne (m-k_2)^2$

$\therefore$ Contradiction

Case 4: $v = a^r b^s$ or $y = a^t b^r$ then

$uv^2xy^2z \notin L \Rightarrow$ Contradiction

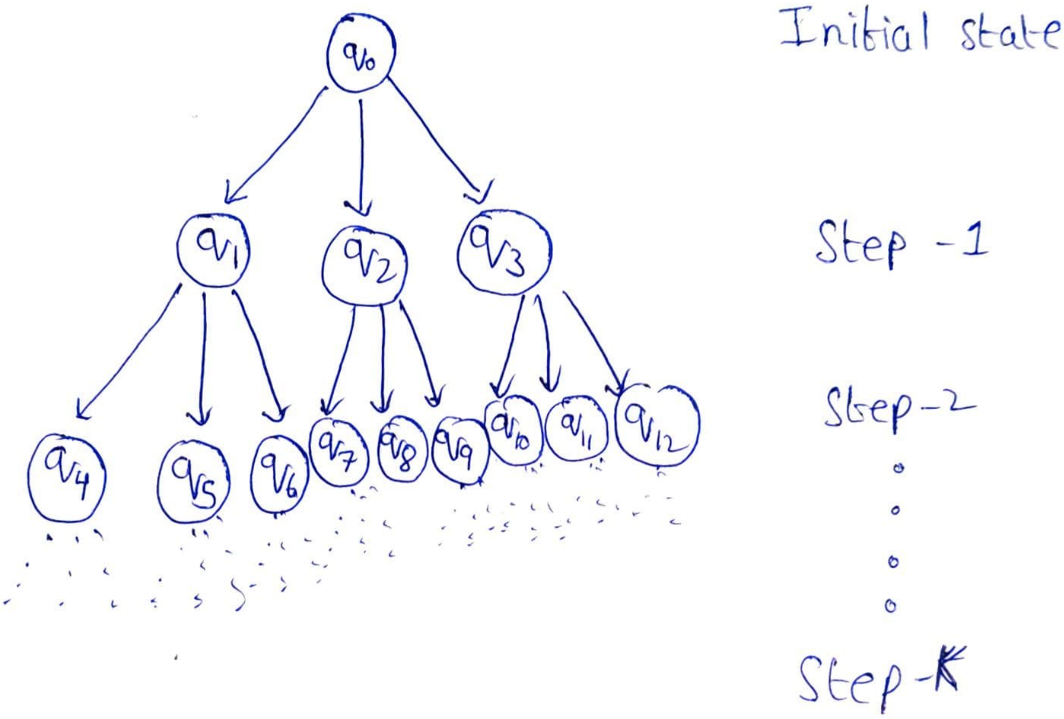
So, All cases indicate contradiction,

Hence our assumption that L is context free is wrong

$\therefore$ L is not context free.

10) RTP: If a Non-deterministic turing machine M takes K steps to solve a problem, then a standard turing machine takes $O(\alpha^{kn})$ steps.

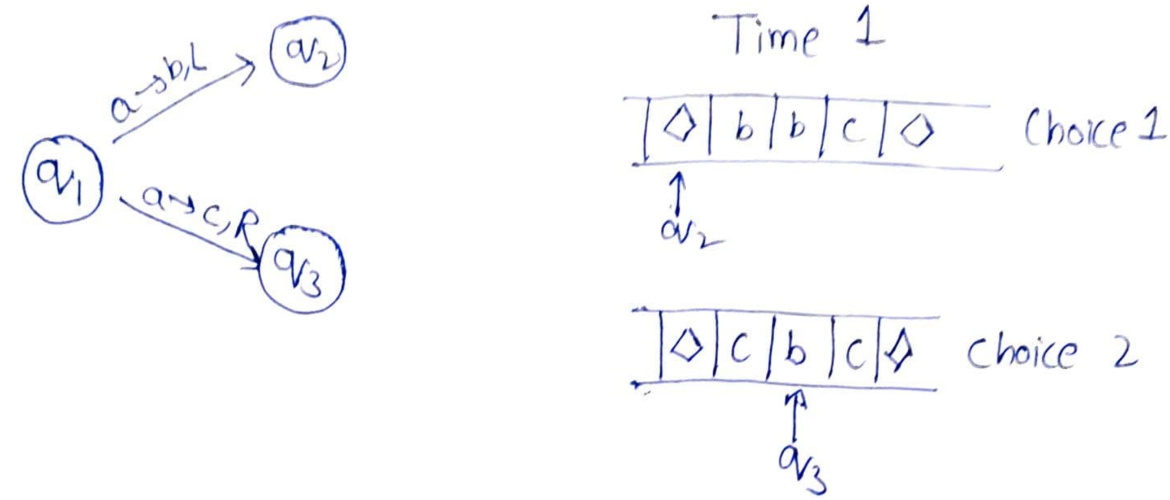
In a Non-deterministic machine,



given, step-K is the one where accepting state is found (HALT state)

Each step of a non-deterministic machine happens in a multi dimension tape. i.e) All the states at same level occur parallely

For example, in the following Non Deterministic machine



This implies, In a non deterministic machine, number of steps is number of input size.

Now, In a standard turing machine,

Total number of steps is equal to the total number of nodes in the initial tree.

~~And~~ $\Rightarrow$ Total number of computations $= 1 + b + b^2 + \dots + b^k$

i.e) $\sum_{i=0}^{k} b^i$ where b is the maximum branch size

$$\Rightarrow \quad 1 + b + b^2 + \dots + b^k = \frac{b^{k+1} - 1}{b - 1} = O(b^k)$$

Time for processing one node (starting from root)

$= O(k)$ ~~where k is input~~

$\therefore$ Total time for standard turing machine $= O(k)\,O(b^k)$

$$\Rightarrow O(k)\cdot O(b^k) \Rightarrow O(\alpha^{\log_\alpha k})\cdot O(b^k)$$

$$\Rightarrow O(\alpha^{\log k})\cdot O((\alpha^{\log b})^k)$$

$$\Rightarrow O(\alpha^{\log k}\cdot \alpha^{k\log b}) \Rightarrow O(\alpha^{\log k + k\log b})$$

$$\Rightarrow O(\alpha^{O(n)}) = O(\alpha^{kn})$$

$\boxed{\text{Hence proved}}$

-END--