25/11/2020

CS303 End Semester Examination

Name of Student: M. Maheeth Reddy

Roll No.: 1801CS31

Answers

1. Given, $\Sigma = \{double, +, *, /, (,)\}$ is alphabet for language Comp = $\{w \in \Sigma^* \mid w$ is a legal arithmetic expression$\}$

Grammar for Comp is

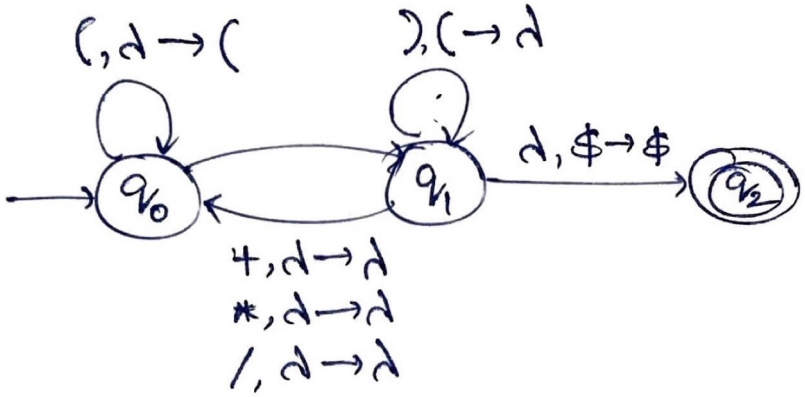
$$E \longrightarrow E + T \mid T$$
$$T \longrightarrow T * F \mid F$$
$$F \longrightarrow T / K \mid K$$
$$K \longrightarrow (E) \mid double$$

∵ Each term on LHS of all productions are single non-terminals, this is Context Free Grammar.

Hence, Comp is context-free.

∴ We can make a PDA for Comp.

The PDA for Comp is



2. Given, $C_g = \{x \# y \mid |x| \neq |y|\}$ is a language over alphabet $\Sigma = \{0, 1, \#\}$

$[x, y \in \Sigma^*]$

Context free grammar for C_g is:

$S \longrightarrow BH \mid KB$

$H \longrightarrow BH \mid A$

$K \longrightarrow KB \mid A$

$A \longrightarrow BAB \mid \#$

$B \longrightarrow 0 \mid 1$

In the above grammar, if the production $S \rightarrow BH$ is used, then strings with $|x| > |y|$ are generated.

If $S \rightarrow KB$ is used, then strings with $|x| < |y|$ are generated.

3. Given, grammar G is

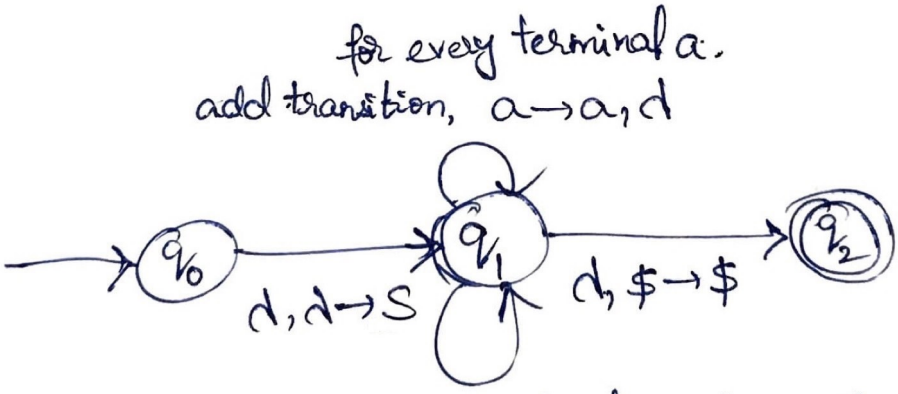
$$S \longrightarrow 1SV0$$
$$S \longrightarrow 1$$
$$V \longrightarrow V0$$
$$V \longrightarrow 1$$

To convert a grammar to PDA:

(1) we need to first push the start symbol to stack

(2) Rewrite the non-terminal on top of stack, according to production

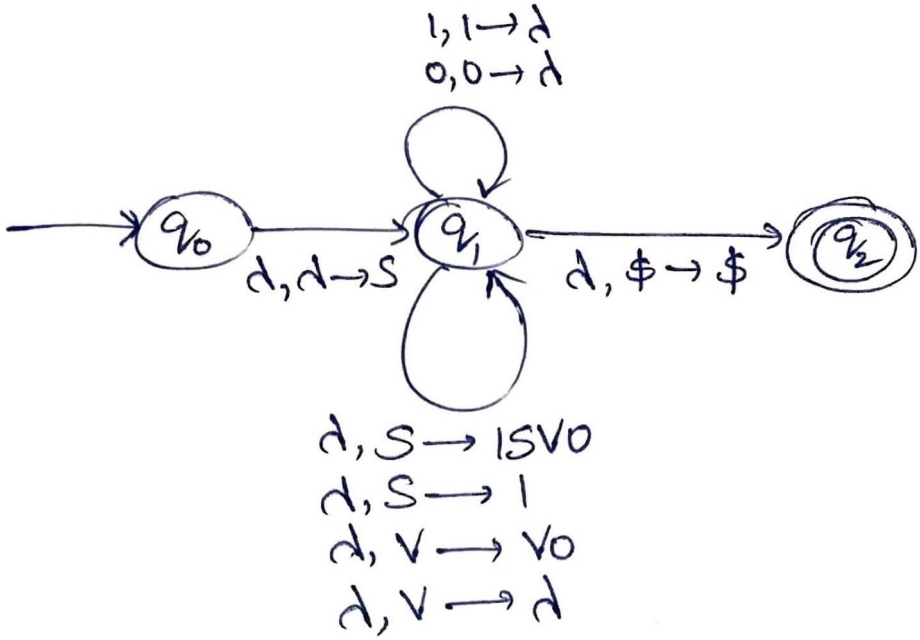
(3) Then, pop the top terminal if it matches input.



for every production, $A \longrightarrow v$ (v: string of terminals and non-terminals)

add transition $\varepsilon, A \longrightarrow v$

So the PDA M with $L(M) = L(G)$ is



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

Assume L is a regular language. Then pumping lemma must hold on L. Consider p as critical length.

Consider a string $S \in L$ as follows.

$S = 0^{2p} w$ where w starts with 1 and $|w| = p$

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

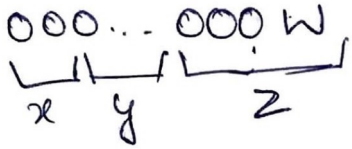
$\because$ L is regular, $\exists\ x, y, z \in \{0,1\}^*$ such that

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

ii) $|y| > 0$

iii) $|xy| \leq p$

Consider $y = 0^k$ $\therefore$ $k > 0$, $|x| + k \leq p$



For $i = 0$, $xy^iz = 0^{2p-k}w \in L$

But $2p - k < 2|w| = 2p \Rightarrow 0^{2p-k}w \notin L$

We got this contradiction because of assuming L is regular.

$\therefore$ L is not regular.

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

Assume L is regular language.

Then pumping lemma holds on L. Assume the critical length as p.

Then, every string $S \in L$ such that $|S| \geq p$ 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) $|xy| \leq p$

Consider, string s as $0^{p!}1^{p!}2^{(p+1)!}$

$$|s| = 2p! + (p+1)! \geq p$$

Now, split s into x, y, z such that $y = a^q$, $q > 0$

and $|xy| \leq p$

$$\Rightarrow q + |y| \leq p \Rightarrow q \leq p$$

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

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

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

$$(p+1)! - 2p! = (p+1-2)p! = (p-1)p!$$

$\because 0 < q \leq p$, q divides $p! \Rightarrow q$ divides $(p-1)p!$

Hence, a valid i exists such that $k = n+m$, for xy^iz

But, as per the assumption that L is regular

this should not happen. Hence, the assumption is wrong

and $\boxed{L = \{0^n 1^m 2^k \mid k \neq n+m\} \text{ is not regular}}$

5. To prove:

$$\left\{\begin{array}{l}\text{Set of deterministic}\\ \text{context free languages}\\ \text{(DCFL's)}\end{array}\right\} \subset \left\{\begin{array}{l}\text{class of context}\\ \text{free languages}\\ \text{(CFL's)}\end{array}\right.$$

Method:

Every DPDA is a PDA. Languages accepted by PDA are CFL's and those accepted by DPDA are DCFL's. If we can show there exists a CFL which is not accepted by a DPDA, the above statement is proved.

Proof:

Consider a language $L = \{a^n b^n \mid n>0\} \cup \{a^n b^{2n} \mid n>0\}$

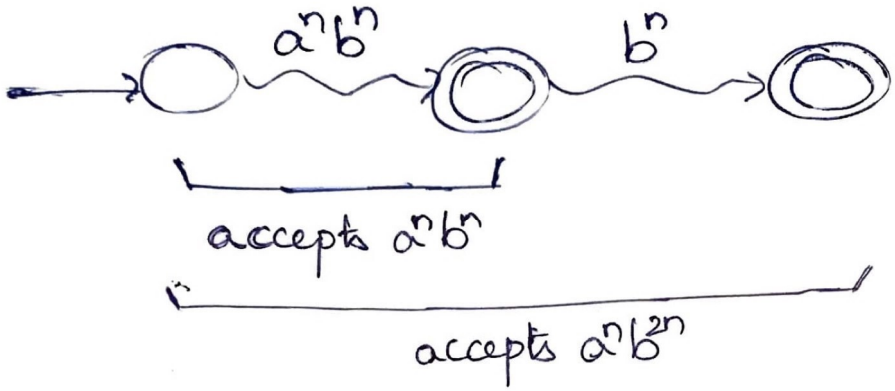
L is a context free language since it can be generated by the below context free grammar

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

Assumption: L is a deterministic CFL.

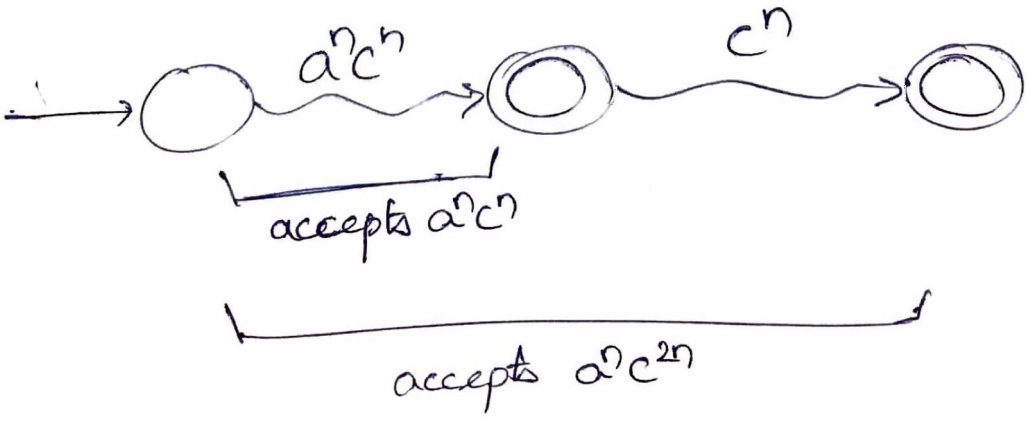
Hence, there exists a DPDA M that accepts L.

A DPDA M with $L(M) = \{a^n b^n \mid n>0\} \cup \{a^n b^{2n} \mid n>0\}$ will have a path as shown due to determinism.



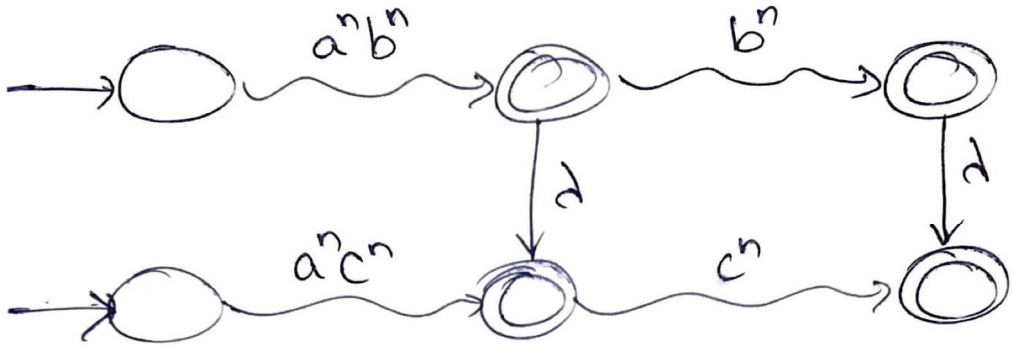
Now, consider another language $L_1 = L \cup \{a^n b^n c^n \mid n>0\}$ We can construct a PDA for L_1 through the following procedure:

Step 1: In the DPDA M that accepts L, replace all b's with c's. This will generate a new DPDA M' with $L(M') = \{a^n c^n \mid n>0\} \cup \{a^n c^{2n} \mid n>0\}$, and will contain the following path.



Step 2:

Connect the final states of M and M' as shown below,



This is a PDA that can accept $L_1 = L \cup \{a^n b^n c^n \mid n>0\}$

$\because L \cup \{a^n b^n c^n \mid n>0\}$ is accepted by a PDA, it must be a context free language.

But, we know $\{a^n b^n c^n \mid n>0\}$ is not a CFL. Hence, $L \cup \{a^n b^n c^n \mid n>0\}$ should not be a CFL.

We obtained this contradiction due to the assumption that $\{a^n b^n \mid n>0\} \cup \{a^n b^{2n} \mid n>0\}$ is a DCFL.

Hence, the assumption is wrong

$\therefore$ There is a CFL which is not accepted by any DPDA.

Hence, as per the method stated at the beginning, set of DCFLs is proper subset of class of CFLs. Hence Proved!

6. We need to design a turing machine for the language

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

The idea for turing machine for language $L' = \{a^n b^n \mid n > 0\}$ can be extended for language L.

The algorithm for turing machine of L' is :

① Match all a's with b's

② Repeat:

Replace leftmost a with x

Find leftmost b and replace it with y

Until:

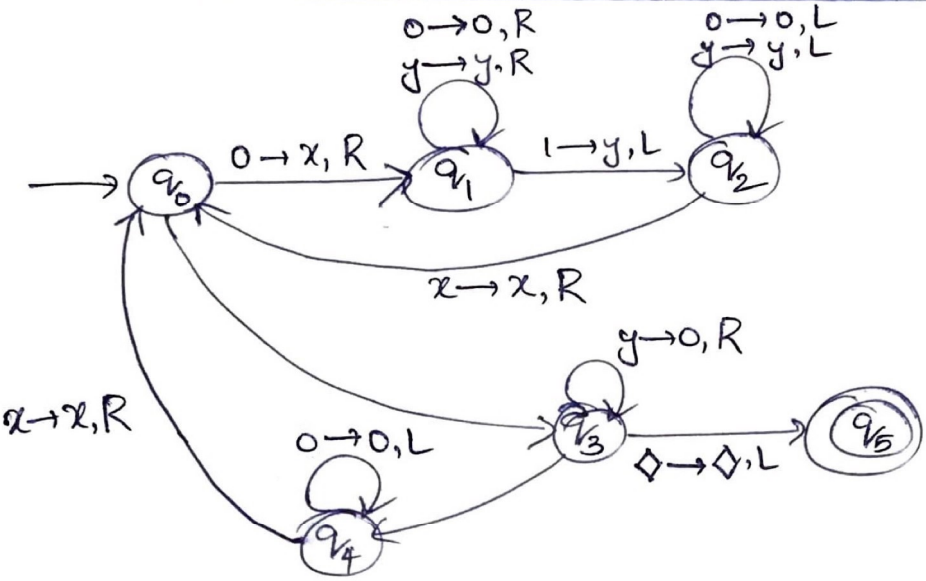
No a's or b's are remaining

If any a or b is leftover, input is rejected.

At the end of the algorithm, a's are replaced by x and b's by y. Now, for language L,

$$0^n \underbrace{1^n 1^n \ldots 1^n}_{c \text{ times}}$$

If the above algorithm is performed once, result is $x^n y^n \underbrace{1^n \ldots 1^n}_{(c-1) \text{ times}}$. If we replace, all y's by 0's and for $x^n 0^n 1^n$ repeat the procedure for c times, we can identify strings that belong to L.



This is the turing machine for $L = \{0^n 1^{cn} : n, c \in \mathbb{N}\}$

---

7. Given Statement: "Does the given turing machine accept a regular language"

To prove: Stated problem is undecidable

Proof:

The statement can be modelled in the form of a Turing Machine. Its language comprises of all Turing Machines that accept a regular language.

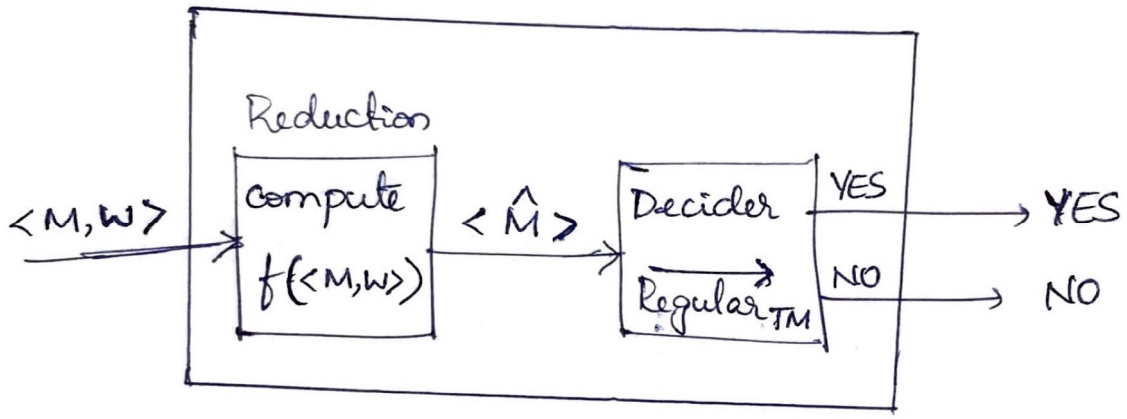
Corresponding Language:

$$Regular_{TM} = \{\langle M \rangle : M \text{ is a turing machine that accepts a regular language}\}$$

Now, we need to prove that $Regular_{TM}$ is undecidable.

For this, we will reduce A_{TM} (membership problem) to $\overline{Regular_{TM}}$ (regular language problem)

Membership Problem Decider

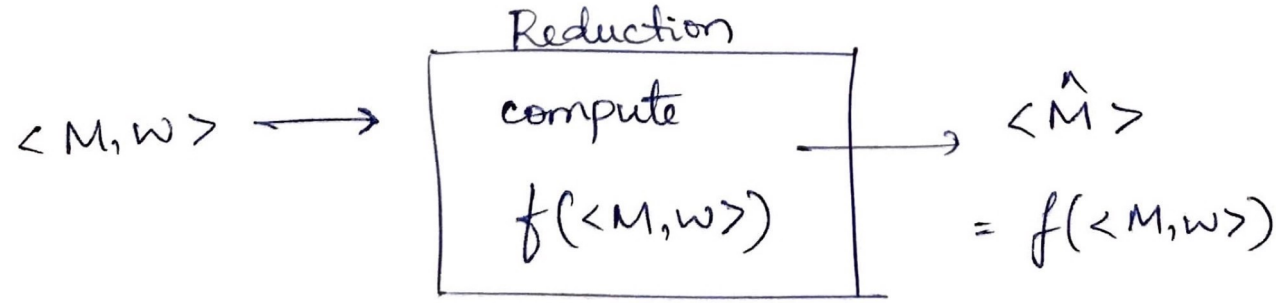


Given the reduction,

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

This is a contradiction since A_{TM} is undecidable

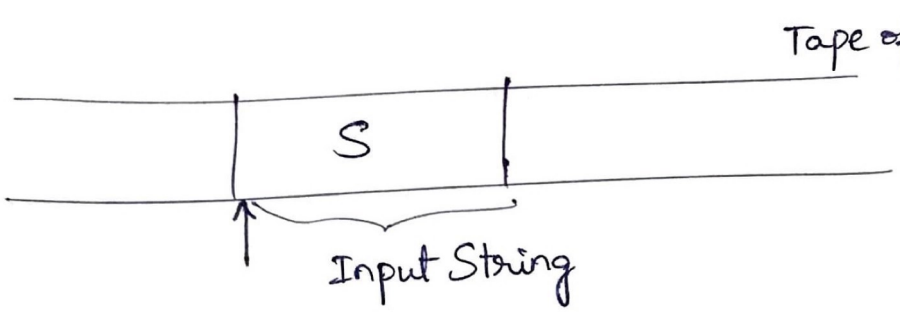
∴ We only need to build the reduction

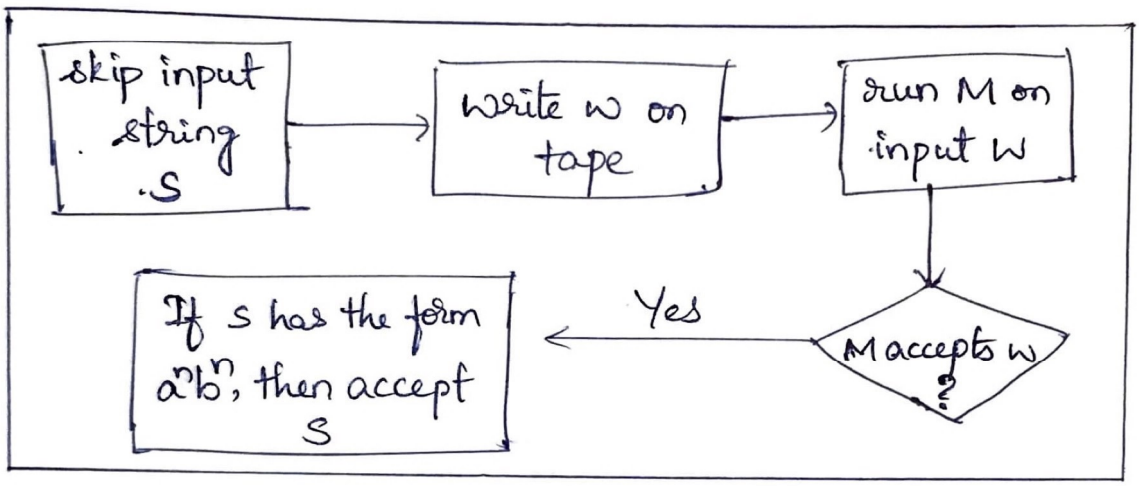


so that

$$\langle M,w\rangle \in A_{TM} \iff \langle \hat{M} \rangle \in \overline{Regular_{TM}}$$

Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:





M accepts $w \implies L(\hat{M}) = \{a^n b^n \mid n \geq 0\}$

↳ not regular

M doesn't accept $w \implies L(\hat{M}) = \phi$ (regular)

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

Equivalently,

$$\langle M, w \rangle \in A_{TM} \iff \langle \hat{M} \rangle \in \overline{Regular_{TM}}$$

So, the stated problem "does a turing machine accept a regular language" is undecidable.

Hence Proved

8. Given grammar G is $S \rightarrow Aba$

$A \rightarrow aab$

$B \rightarrow Ac$

Now, $B \rightarrow Ac$ is a useless production. It can be removed

Grammar G is $S \rightarrow Aba$

$A \rightarrow aab$

$S \rightarrow Aba$ is invalid under Chomsky Normal Form (CNF) after converting it to valid form

G is $S \rightarrow AT_1$

$A \rightarrow aab$

$T_1 \rightarrow ba$

$A \rightarrow aab$ is invalid under CNF, after converting it to valid form

G is $S \rightarrow AT_1$

$A \rightarrow T_a T_2$

$T_1 \rightarrow ba$

$T_2 \rightarrow ab$

$T_a \rightarrow a$

$T_1 \rightarrow ba$ is invalid under CNF, after converting to valid form,

G is $S \rightarrow AT_1$

$A \rightarrow T_a T_2$

$T_1 \rightarrow T_b T_a$

$T_2 \rightarrow ab$

$T_a \rightarrow a$

$T_b \rightarrow b$

$T_2 \rightarrow ab$ is invalid under CNF, after converting to valid form.

G is $S \rightarrow AT_1$

$A \rightarrow T_aT_2$

$T_1 \rightarrow T_bT_a$

$T_2 \rightarrow T_aT_b$

$T_a \rightarrow a$

$T_b \rightarrow b$

∴ Chomsky normal form of G is $S \rightarrow AT_1$

$A \rightarrow T_aT_2$

$T_1 \rightarrow T_bT_a$

$T_2 \rightarrow T_aT_b$

$T_a \rightarrow a$

$T_b \rightarrow b$

9. Given: language $L = \{0^n 1^j : n = j^2\}$

To prove: L is not context free

Proof:

Assume, L is a context free language, then pumping lemma must hold on L. Consider 'n' as critical length.

Now, $0^{m^2}1^m \in L$ and $|0^{m^2}1^m| \geq m$

$\therefore \exists\ u, v, x, y, z$ such that $uvxyz = 0^{m^2}1^m$

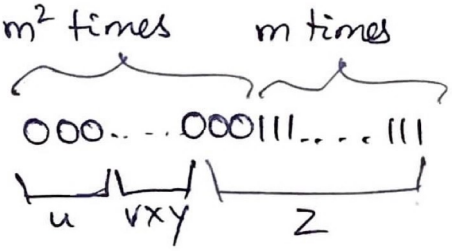
$|vxy| \leq m$

$|vy| \geq 1$

$\forall\ i \geq 0,\ uv^ixy^iz \in L$

Consider u, v, x, y, z as follows

Case (i):



Consider, $\left.\begin{matrix} v = 0^{k_1} \\ y = 0^{k_2} \end{matrix}\right\}$ s.t $\left\{\begin{matrix} k_1, k_2 \geq 1 \\ |vxy| \leq m \quad |vy| \geq 1 \end{matrix}\right.$

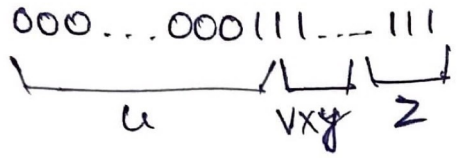
$\Rightarrow 1 \leq k_1 + k_2 \leq m$

For $i=2$, $uv^ixy^iz = 0^{m^2+k_1+k_2}1^m \in L$

but $m^2 + k_1 + k_2 \neq m^2$

$\therefore$ Pumping lemma doesn't hold in this case

Case(ii):



$k_1, k_2 \geq 1$ $\quad v = 1^{k_1}$, $y = 1^{k_2}$ s.t $\quad k_1, k_2 \geq 1$, $|vxy| \leq m$, $|vy| \geq 1$

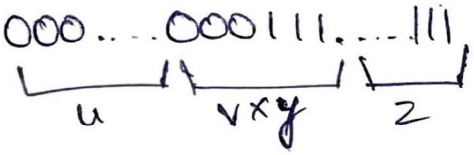
$\Rightarrow 1 \leq k_1 + k_2 \leq m$

If $i=0$, $uv^i xy^i z = 0^{m^2} 1^{m-k_1-k_2} \in L$,

but $m^2 \neq (m-k_1-k_2)^2$

∴ Pumping Lemma doesn't hold in this case.

Case(iii):



Subcase①: If v spans both 0 and 1, then if pumping is done, the resultant string has 0's and 1's mixed up. So, it doesn't belong to L. Same holds for y.

Subcase (2): $v = 0^{k_1}$ $y = 1^{k_2}$ such that

$k_1, k_2 \geq 1$, $|vxy| \leq m$, $|vy| \geq 1$

$\Rightarrow 1 \leq k_1 + k_2 \leq m$

For $i=0$, $uv^i xy^i z = a^{m^2-k_1} b^{m-k_2} \in L$

But, $(m-k_2)^2 \leq (m-1)^2$ $[\because k_2 \leq 1]$

$= m^2 - 2m + 1 < m^2 - k_1$ $(\because k_1 < m)$

$\Rightarrow (m^2 - k_1) \neq (m-k_2)^2$

$\therefore$ Pumping Lemma doesn't hold in this case.

$\because$ Pumping Lemma didn't hold in any case,

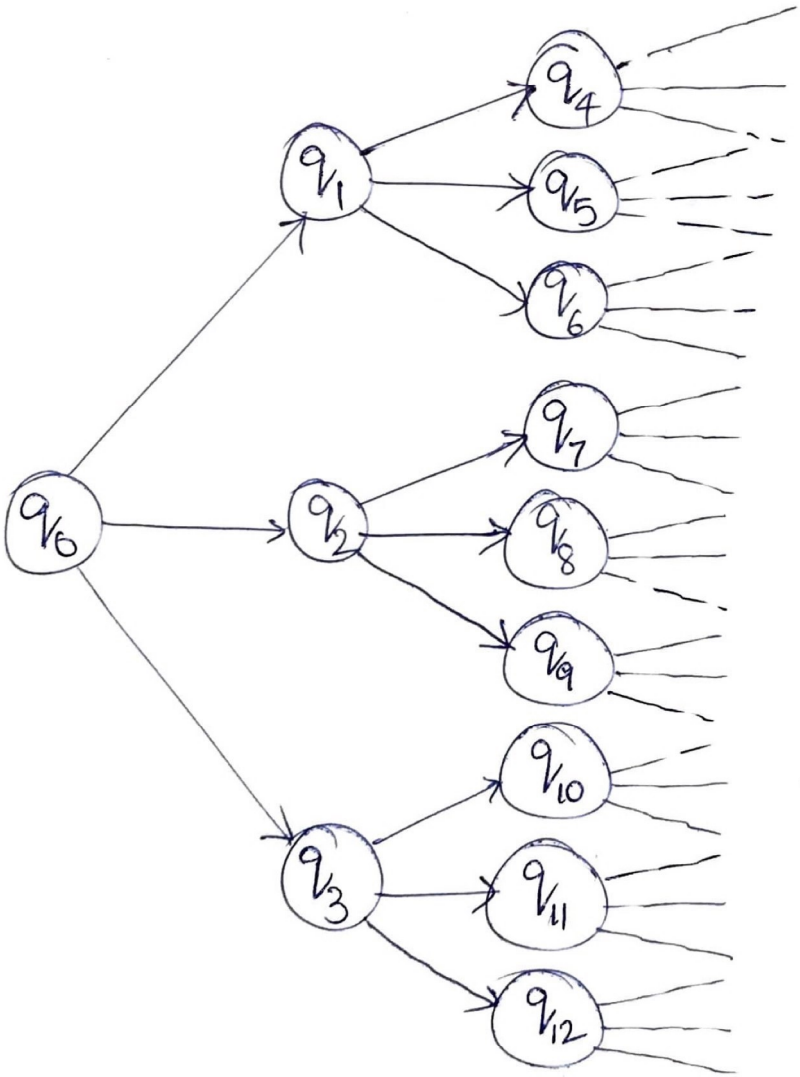
$L = \{0^n 1^j : n = j^2\}$ is not context-free. Hence Proved!

10. Given: A non-deterministic turing machine M takes k steps to solve a problem.

To prove: A standard turing machine takes $O(\alpha^{kn})$ steps [α, n are independent of k]

Proof:

Consider non-deterministic TM, M as follows.

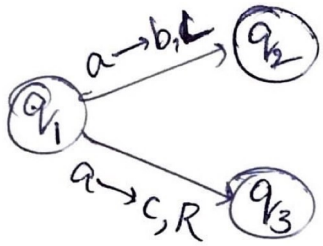


Initial State | Step 1 | Step 2 | ------ | Step k

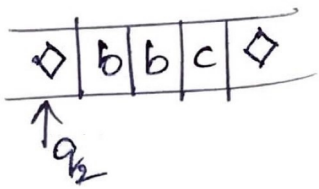
The accepting state is found in Step-k states.

Each step of the non-deterministic TM happens in a multi-dimension tape. i.e., all states at the same level occur parallely.

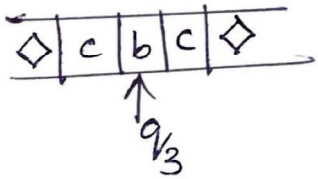
For example, in the following non-deterministic TM,



At Time 1, there are 2 choices



Choice 1



Choice 2

We can infer that, in a non-deterministic TM, no. of steps is no. of input size.

Now, in a standard turing machine,

Total no. of steps = Total no. of nodes in the initial tree

$$\Rightarrow \text{Total no. of computations} = 1 + b + b^2 + \dots + b^k$$

$$= \frac{b^k - 1}{b - 1} = O(b^k)$$

where b is maximum branch size.

Time for processing one node (starting from root) = $O(k)$

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

$= O(\alpha^{\log_\alpha k})O(b^k)$ $[\because k = \alpha^{\log_\alpha k}]$

$= O(\alpha^{\log k})O((\alpha^{\log_k b})^k)$ $[\because b = \alpha^{\log_\alpha b}]$

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

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

$= O(\alpha^{O(k)})$

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

· where α, n are constants

Hence proved that if a non-deterministic TM takes k step to solve a problem, then a standard turing machine takes $O(\alpha^{kn})$ steps [α, n are independent of k]