

# COL352: Assignment 2

Sachin 2019CS10722  
Saurabh Verma 2019CS50129  
Sriram Verma

March, 2022

## 1 Question 1

We say that a context-free grammar  $G$  is self-referential if for some non-terminal symbol  $X$  we have  $X \rightarrow^* \alpha X \beta$ , where  $\alpha, \beta \neq \varepsilon$ . Show that a CFG that is not self-referential is regular.

**Proof:** Suppose our grammar  $G$  has  $N$  symbols  $X_1, X_2, \dots, X_N$  and for each pair  $i, j$  there exist strings  $\alpha, \beta$  such that  $X_i \Rightarrow \alpha X_j \beta$ .

**Proof by induction(on number of non terminal symbols):**

**(Base Step:(N=1)**) Let  $G$  has only one  $n$  non terminal symbol  $X$ . Hence language is in the form  $X \rightarrow aX$  or  $X \rightarrow Xb$  a Finite state Language.Hence regular.

**Induction Hypothesis:(N|n)** Suppose grammar that is not self referential with less than  $n$  non terminal symbols is regular.

**Induction Step:(N=n)** Let  $G$  has  $n$  non terminal symbols  $X_1, X_2, \dots, X_n$ . Let for some particular  $i, j$ , there are no strings  $\alpha, \beta$  such that  $X_j \Rightarrow \alpha X_i \beta$ . Let  $L$  be language generated by this grammar  $G$ . Lets remove this rule  $X_j \Rightarrow \alpha$  from grammar  $G$  and replaced with a new rule with terminal symbol  $c$  such that  $X_j \Rightarrow c$ . Now this newly created grammar be  $G_1$  and language generated from this grammar be  $L_1$ . From hypothesis,  $L_1$  is regular and set from  $X_j$  is also finite. So,  $L$  is a finite state language.Hence  $L$  is regular.

## 2 Question 2

Prove that the class of context-free languages is closed under intersection with regular languages. That is, prove that if  $L_1$  is a context-free language and  $L_2$  is a regular language, then  $L_1 \cap L_2$  is a context-free language. Do this by starting with a DFA

Let us suppose there is a CFL  $L$  and a regular langauge  $R$ . The pushdown automata that accepts  $L$  be  $P=(S_1, \Sigma, \Gamma, \delta_1, s_1, F_1)$  and the DFA accepting  $R$  be  $D=(S_2, \Sigma, \delta_2, s_2, F_2)$ . Now we have to show that the language  $L \cap R$  is CFL. To show this it is enough to provide a PDA that accepts it. So we will construct such a PDA to prove that  $L \cap R$  is CFL.

**To Prove:**  $L \cap R$  is CFL.

**Proof:** We will the above hypothesis by construction. The main idea behind the construction of the PDA is that we will run both the original PDA  $P$  for  $L$  and the DFA  $D$  in parallel on the input string and will only accept when the we reach an accepting state both in  $P$  and  $D$ . The construction of the PDA is described below:

**Construction:** Let the PDA which accepts  $L \cap R$  be  $M=(S, \Sigma, \Gamma, \delta, (s_1, s_2), F)$ . Here  $S$  is  $S_1 * S_2$  and  $F$  is  $F_1 * F_2$ . The transition function  $\delta$  is described as follow:

For all the transitions  $((p_1, a, \alpha), (p_2, \beta)) \in \delta_1$  and  $(q_1, a, q_2) \in \delta_2$  add the transition  $((p_1, q_1), a, \alpha), ((p_2, q_2), \beta))$  in  $\delta$ .

Also, for all the transitions  $((p_1, \epsilon, \alpha), (p_2, \beta)) \in \delta_1$  and  $\forall q \in S_2$  add the transition  $((p_1, q), a, \alpha), ((p_2, q), \beta))$  in  $\delta$ .

Here  $p_1, p_2 \in S_1$   $q_1, q_2 \in S_2$   $a \in \Sigma$   $\alpha, \beta \in \Gamma$

The accepting condition is that the final state reached after reading the input must belong to  $F$ .

Now our claim is that PDA  $M$  exactly recognises every string that is in  $L \cap R$ .

**Claim:** The PDA  $M$  constructed above exactly recognises strings in  $L \cap R$ .

**Proof:** We will have to show two things first that every string in  $L \cap R$  is accepted by  $M$ . Lets prove this. Choose any string  $w \in L \cap R$ . Then its run on the DFA  $D$  would be something like  $s_2, q_1, \dots, q_k$  where  $q_k \in F_2$ , also  $w$  would take the PDA  $M$  from start configuration  $(s_1)$  to an accepting configuration  $(q_k')$  in some steps. By the way we have constructed the PDA  $M$  the computations of  $M$  and  $D$  will happen in parallel. So  $(s_1, s_2)$  is the start configuration of the PDA. First state in the tuple denotes the state that would have been in the PDA  $P$  and second state denotes the state that would have been in the DFA  $D$  after reading input upto some point. So when the PDA gets to wun on  $w$ . It would take  $M$  from from  $(s_1, s_2)$  to  $(q_k, q_k')$ . Now this is an accepting configuration in  $M$  by the way we defined  $F(F_1 * F_2)$ . So all the string  $w \in L \cap R$  are accepted by  $M$ .

Also we have to prove that all the strings say  $w$  that are accepted by  $M$  should also be present in  $L \cap R$ . We will accept  $w$  if it takes PDA  $M$  from  $(s_1, s_2)$  to  $(q_1, q_2)$ ,  $q_1 \in F_1$  and  $q_2 \in F_2$ . Also we have showed that  $M$  is parallely running  $P$  and  $D$  where first state in the tuple means the state reached in  $P$  and second

state means the state reached in D after reading the input upto that point. So after reading w if the M is in state  $(q_1, q_2)$ , it would mean that after reading w P would have been in  $q_1$  and D would be in  $q_2$ .  $q_1 \in F_1$ , so  $w \in L$  also  $q_2 \in F_2$  so  $w \in R$  which implies  $w \in L \cap R$ .

Thus we have successfully constructed a PDA M which accepts  $L \cap M$ . Thus CFL's are closed under intersection with regular languages. Hence proved.

### 3 Question 3

Given two languages  $L, L'$ , denote by

$$L||L' := \{x_1y_1x_2y_2 \dots x_ny_n \mid x_1x_2 \dots x_n \in L, y_1y_2 \dots y_n \in L'\}$$

Show that if  $L$  is a CFL and  $L'$  is regular, then  $L||L'$  is a CFL by constructing a PDA for  $L||L'$ . Is  $L||L'$  a CFL if both  $L$  and  $L'$  are CFLs? Justify your answer.

For the first part we have to show that  $T=L||L'$  is a CFL if  $L$  is CFL and  $L'$  is regular. Since  $L$  is CFL we are given a PDA  $P=(S_1, \Sigma, \Gamma, \delta_1, s_1, F_1)$  which recognizes it. Also we have a DFA  $D=(S_2, \Sigma, \delta_2, s_2, F_2)$  which recognises  $L'$ . These are the things given to us.

**To Prove:**  $T$  is CFL.

**Proof Idea:** To show that  $T$  is CFL we would have to construct a PDA say  $R$  which accepts it. Now the main idea behind the working of this machine is that after reading alphabet of the input tape it would first check whether the position is odd or even (this can be done with an extra variable in the state) If the position is even then production rules of PDA  $P$  apply or else the transition rules of  $D$  apply. In the transitions we also have to update the position counter which tell us whether we are at odd or even location. So basically we are running PDA  $P$  on odd position and DFA  $D$  on even positions. The final accepting condition would be that position reached be even and states in both PDA  $P$  and DFA  $D$  are final. The specific construction of  $R$  is given as follows.

**Construction:**  $R=(S, \Sigma, \Gamma, \delta, s, F)$ . Here  $S$  is  $\{0, 1\} * S_1 * S_2$ ,  $F = \{1\} * F_1 * F_2$ . The start state  $s=(1, s_1, s_2)$  The transition function  $\delta$  is described as below  
 $\delta((1, p_1, q_1), a, \alpha)$  would contain  $((0, p_2, q_1), \beta)$  if  $\delta_1(p_1, a, \alpha)$  contained  $(p_2, \beta)$ .  
 $\delta((0, p_1, q_1), a, \alpha)$  would contain  $((1, p_1, q_2), \alpha)$  if  $\delta_2(q_1, a) = \{q_2\}$ .

Here  $p_1, p_2 \in S_1$   $q_1, q_2 \in S_2$   $a \in \Sigma$   $\alpha, \beta \in \Gamma$

The accepting condition is that the final state reached after reading the input must belong to  $F$ .

Now our claim is that PDA  $R$  exactly recognises every string that belongs to  $T$ .  
**Claim:**  $R$  recognises the language  $T$ .

**Proof:** We have to show two things first that any string accepted by  $R$  belongs to  $T$  and every string belonging to  $T$  is accepted by  $R$ .

Let us consider any string  $w$  and  $w' |w| = |w'| = n$ .  $t = w||w'$ . Let the run of  $P$  be  $(q_1, \epsilon), (q_2, \alpha_2), (q_3, \alpha_3), \dots, (q_{n+1}, \alpha_{n+1})$  for  $w$  and the run of  $D$  on  $w'$  be  $p_1, p_2, p_3, \dots, p_{n+1}$ .  $q_1 = s_1$  and  $q_2 = s_2$ . Now what would be the run of  $w||w'$  on  $R$ ? Start configuration of  $R$  would be  $((1, q_1, p_1), \epsilon)$ . Now the way we defined our transition function on odd positions it follows the transition rules of  $P$  (and only second variable of the state tuple i.e. state of  $P$  would be modified) and on even position rules of  $D$  (and only third variable of the state tuple i.e. state of  $D$  would be modified). We also know rules of  $D$  not make any changes in the stack. Thus after reading  $2*i | i > 0$  length of input of  $t$  the configuration that  $R$  would be in is  $((1, q_{i+1}, p_{i+1}), \alpha_{i+1})$  and after reading  $(2*i + 1) | i \geq 0$  configuration would be  $((0, q_{i+1}, p_i), \alpha_{i+1})$ . Now when all the input is read  $i=n$

so the configuration that R would be in is  $((1, q_{n+1}, p_{n+1}), \alpha_{n+1})$ . If  $w \in L$  then  $q_{n+1} \in F_1$  and  $w' \in L'$  then  $p_{n+1} \in F_2$ . So when  $t=w||w'$  run on R final state  $(1, q_{n+1}, p_{n+1}) \in F$ . So t is accepted. So every string in T is recognised by R. Now lets prove the other way.

Suppose R accepts a string t so final state  $(1, q_{n+1}, p_{n+1}) \in F$  which would mean  $q_{n+1} \in F_1 \rightarrow w \in L$  and  $p_{n+1} \in F_2 \rightarrow w' \in L'$ . Thus  $t=w||w', t \in T$ .

Thus we have show that any string accepted by R belongs to T and every string belonging to T is accepted by R. So our claim that R recognizes T is true.

Since we have constructed a PDA R for the language  $T=L||L'$ , we have sucessfully proved that T is indeed a CFL.

Now let us see the second part of the question. Is  $L||L'$  a CFL if both L and  $L'$  are CFLs? We will show that  $L||L'$  is not a CFL if L and  $L'$  are CFLs.

**To Prove:**  $L||L'$  is not a CFL if L and  $L'$  are CFLs.

**Proof:** Giving a counter example will show that its not the case. Let us take two context free language and then we will show that  $L||L'$  is not a CFL.  $L=\{a^n b^n \mid n \geq 0\}$ . CFG for L is  $G=(\{S\}, \{a, b\}, \{S \rightarrow aSb \mid \epsilon\}, S)$ . Thus L is CFL. Let  $L' = \{a^n b^{3n} \mid n \geq 0\}$  CFG for  $L'$  is  $G' = (\{S\}, \{0, 1\}, \{S \rightarrow 0S111 \mid \epsilon\}, S)$ . Thus  $L'$  is CFL.

Let us take the length of the string  $4k$  where  $k \geq 0$  ( Thats the length of the string nedeed because  $L'$  contains string of the multiple of 4 and for  $L||L'$  length of string of L and  $L'$  must be same). So string in L is of the form  $a^{2k}b^{2k}$  and string in  $L'$  is of the form  $0^k1^{3k}$ . Hence we see  $L||L'$  is  $(a0)^k(a1)^k(b1)^{2k}$ . Now our proof boils down to showing that  $(a0)^k(a1)^k(b1)^{2k}$  is not CFL.

**Claim:**  $T=(a0)^k(a1)^k(b1)^{2k}$  is not CFL.

**Proof:** We will use pumping lemma for CFL's. Let us consider the langauge is CFL. . Let p be the pumping length. Consider  $z=(a0)^p(a1)^p(b1)^{2p} \in T$ . Since  $|z| > p$ , there are u, v, w, x, y such that  $z = uvwxy$ ,  $|vwx| \leq p$ ,  $|vx| > 0$  and  $uv^iwx^iy \in L$  for all  $i \geq 0$ . Since  $|vwx| \leq p$  vwx would either contain a0,a1,b1 if vwx lies entirely in one of the 3 partitions else it could contain either a01,ab1 if vwx spans two partitions. In all these cases vwx will never contain all the four non-terminal symbols. Hence for every split if we pump up there would be imbalance in atleast one of the symbols. Hence the given language is not a CFL.

Since we have given a counter example in which the  $L||L'$  is not a CFL given L and  $L'$  are CFL we have sucessfully proved the hypothesis.

## 4 Question 4

For  $A \subseteq \Sigma^*$ , define  $\text{cycle}(A) = \{yx \mid xy \in A\}$  For example if  $A = \{aaabc\}$ , then  $\text{cycle}(A) = \{aaabc, aabca, abcaa, bcaaa, caaab\}$  Show that if  $A$  is a CFL then so is  $\text{cycle}(A)$

Let us suppose that we do have a CFG  $G = (V, T, P, S)$  for the language  $A$  in Chomsky normal form. We know for a fact that since  $A$  is CFL it will have a CFG. Now to prove that  $\text{cycle}(A)$  is also a CFL. So we will construct a CFG for  $\text{cycle}(A)$  that would show that  $\text{cycle}(A)$  is CFL.

**To Prove:**  $\text{Cycle}(A)$  is CFL.

**Proof Idea:** Let us consider any string  $w$  in language  $A$  of the form  $x_1x_2$ . For any CFG  $G'$  to accept  $w' = x_2x_1$  there should exist a parse tree for  $w'$  in  $G'$ . We will try to construct a parse tree of  $w'$  from parse tree of  $w$  from leftmost non terminal  $b$  of  $x_2$  by somehow backtracking  $x_2x_1$  from  $b$ . And add required transitions step by step.

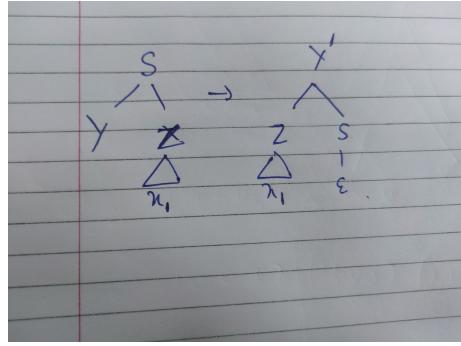
Now, let's see what are the production rules we require:-

1. A new start state  $S_0$   
To accept new strings. As previous start state is limited to  $A$ .
2.  $S_0 \rightarrow S$  and All productions of  $G$ .  
For satisfying the case when  $x_1$  or  $x_2 = \epsilon$
3. state  $X' \forall X \in G$ .  
These states will denote the reverse of the corresponding states of  $G$ . (to backtrack and get  $a, b$  formed till any transition  $S \rightarrow_* aXb$  in  $G$ )
4.  $S_0 \rightarrow aX'$  for any production rule of the form  $X \rightarrow a \in G$   
This is to start backtracking from left most non terminal of  $x_2$
5.  $S' \rightarrow \epsilon$   
This is to end backtracking (at the end we will reach at root when got complete string  $x_2x_1$ ).
6.  $Y' \rightarrow ZX' \forall X \rightarrow YZ$   
This is to backtrack and get complete string on the right of  $Y$ .

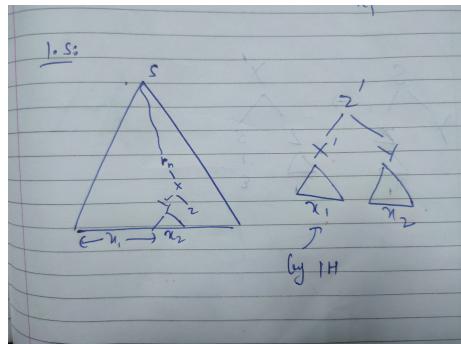
**Proof:**

Proving this inductively on number of parents of  $X$  from its path from  $S$  (lets say  $k$ ):

- **Base case:**  $k = 0$  hence,  $X = S$   
Only string on the right of  $Y$  is subtree of  $Z$  (let's say  $z$ )  
Then using above production rule we can generate  $z$  from  $Y'$  as follows:



- **I.H.:** Let's assume the statement holds  $\forall k \geq n$
- **I.S.:** Now, if  $k = n + 1$  Let's say subtree(z) = x1 and remaining string on right of Y is x2. Using the transition  $Y' \rightarrow ZX'$ , x1 can be generated from Z, and by induction hypothesis (as  $k = n$  for  $p_n$ ) we can generate x2 from  $X'$ .



Hence done

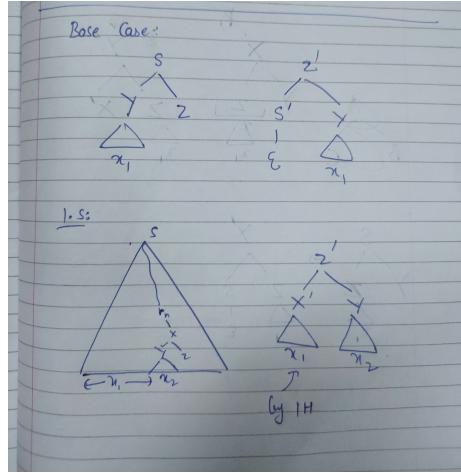
$$7. Z' \rightarrow X'Y \quad \forall X \rightarrow YZ$$

This is to backtrack and get complete string on the left of Z.

**Proof:**

Proving this inductively on number of parents of X from its path from S (lets say k):

Proof is very similar to the last one. Just now we are backtracing on the left side, still following is the diagram representation of the proof:



Now let's use the above added production rules and grammar to construct a CFG for cycle(A)

Let  $G'$  be the grammar formed by adding above production rules and states to  $G$ .

Lets say any string  $x_1x_2 \in A$   
 $\Rightarrow \exists$  a parse tree P for  $x_1x_2$  in  $G$

Let's try to construct a parse tree for  $x_2x_1$  in  $G'$

Suppose left most non terminal of  $x_2$  is  $b$  then start from the production rule  $S_0 \rightarrow bp_k$  where  $p_k$  is immediate parent of  $b$  in P

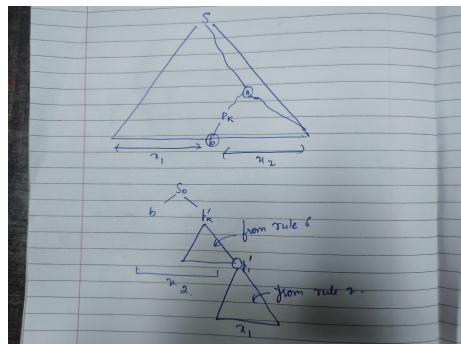
Also lets say  $p_1$  is the highest parent of  $b$  in P such that  $b$  is in the left subtree of  $p_1$ .

Then assuming subtree of  $p_1$  be the independent tree, we can construct  $x_2$  from  $p_k$  by rule 6.

But we will end at  $p_1$  to be the rightmost child of  $S_0$

From  $p_1$  we can generate  $x_1$  by rule 7.

Consider the following figure for detailed explanation:



Hence we have constructed a CFG such that its accepts  $cycle(A)$ , hence it is CFL.

## 5 Question 5

Let

$$A = \{wtw^R \mid w, t, \in \{0, 1\}^* \text{ and } |w| = |t|\}$$

Show that A is not a CFL.

We will prove that A is not CFL by contrapositive of pumping lemma. i.e. we need to show the following:-

$$\forall p \geq 0$$

$$\exists s \in A : |s| \geq p$$

$$\forall uvxyz = s : |vy| > 0, |vxy| \leq p$$

$$\exists i : uv^i xy^i z \notin A$$

Let  $s = 0^n 1^{\frac{n}{2}} 0^{\frac{n}{2}} 0^n$  ( $w = w^R = 0^n, t = 1^{\frac{n}{2}} 0^{\frac{n}{2}}, n = \text{any even number more than } p$ )

Now, lets divide s in 3 parts = wab, where  $a = 1^{\frac{n}{2}}$  and  $b = 0^{\frac{n}{2}} 0^n$

Considering all partitions of s = uvxyz such that  $|vy| > 0, |vxy| \leq p \leq n$

1. Case 1:  $vxy \subseteq w$

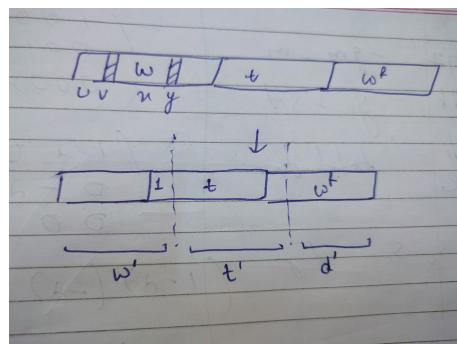


Figure 1: Case 1

Considering  $s' = uxz$  i.e.  $i = 0$

For  $s'$  to be in A, its should be divided into 3 halves =  $w't'd'$  of equal length such that first and third are reverse.

Since, we have removed characters from w only,  $w'$  will have all the characters of w left after removal of v,y and also some 1's from t for it to have same length as other subparts of equal length (see fig 1).

Hence,  $1 \in w'$ , but  $d' \subset w^R = 0^n \Rightarrow 1 \notin d' \Rightarrow d' \neq w'^R$

Hence,  $s' \notin A$

2. Case 2:  $vxy \subseteq b$

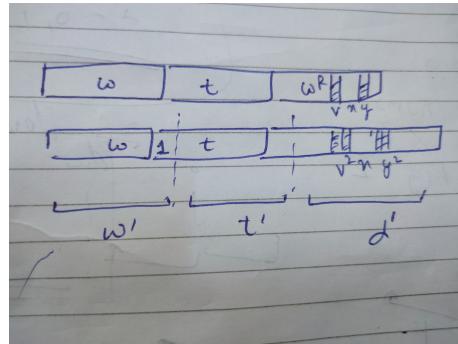


Figure 2: Case 2

Considering  $s' = uv^2xy^2z$  i.e.  $i = 2$

Let,  $|s| = l, |s'| = l', l' < l$

$$|w| = \frac{l}{3} < \frac{l'}{3}$$

$\Rightarrow w \subset w' \Rightarrow w'$  will overflow towards t  $\Rightarrow 1 \in w'$

But,  $d' \subset w'^R$  (after pumping) that only has 0 (see fig 2)

$$\Rightarrow d' \neq w'^R$$

Hence,  $s' \notin A$

3. Case 3:  $vxy \cap a \neq \phi$

(a)  $w \cup vxy = \phi$

Considering  $s' = uv^2xy^2z$  i.e.  $i = 2$

This case is similar to case 2, as length of string is increased but w remains unchanged, hence  $w'[1:n] = w = 0^n$  and  $w'[n+1] = 1$ . But  $(n+1)^{th}$  of d' from last = last bit of t = 0.  $\Rightarrow d' \neq w'^R$

Hence,  $s' \notin A$

(b)  $w \cup vxy \neq \phi$

Considering  $s' = uxz$  i.e.  $i = 0$

This case is similar to case 1, as length of string is decreased (all from w). Hence,  $1 \in w'$ , but  $1 \notin d'$  (from case 1)  $\Rightarrow d' \neq w'^R$

Hence,  $s' \notin A$

Hence A is not CFL.

## 6 Question 6

Prove the following stronger version of pumping lemma for CFLs: If  $A$  is a CFL, then there is a number  $k$  where if  $s$  is any string in  $A$  of length at least  $k$  then  $s$  may be divided into five pieces  $s = uvxyz$ , satisfying the conditions:

1. for each  $i \geq 0$ ,  $uv^i xy^i z \in A$
2.  $v \neq \epsilon$ , and  $y \neq \epsilon$ , and
3.  $|vxy| \leq k$ .

The given lemma differs from standard pumping lemma at point 2 only, so let's start the proof in a similar way to that of pumping lemma.

Let  $G = GFG$  for CFL  $A$

$b =$  maximum number of non terminals in the right side of any production rule.

$\Rightarrow$  in parse tree each node will have at most  $b$  childs.

$\Rightarrow$  a parse tree of height  $h$  corresponds to string of length at most  $b^h$

$\Rightarrow$  if  $|s| \geq b^h + 1$  then height of its parse tree is at least  $h + 1$

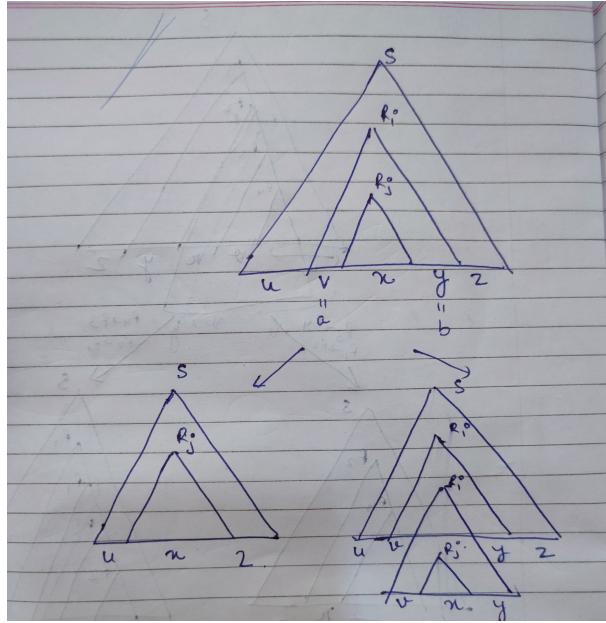
Now, set pumping length  $p = b^{4|V|+1}$ , where  $|V| =$  no of non terminals.  
 $\forall s : |s| \geq p = b^{4|V|+1}$  will have height of parse tree at least  $4|V| + 1$   
Assume longest path  $l$  from start state.  $|l| \geq 4|V| + 1 \Rightarrow$  no of non nodes in  $l \geq 4|V| + 2$   
 $\Rightarrow$  no of non lead nodes (non terminals) in  $l \geq 4|V| + 1$

Now, by pigeonhole principle one non terminal is repeated at least 4 times in  $l$ . Let lowest such non terminal to be  $R$ .

Lets name the 4 occurrences of  $R$  to be  $R_1, R_2, R_3, R_4$  from top to bottom. Now, 2 cases are possible:

1. **Case 1:**  $\exists i, j : R_i \rightarrow_* aR_j b (a \neq \epsilon, b \neq \epsilon)$

Lets consider the following partition of  $s = uvxyz$ :



i.e.  $a = v, b = y, \text{subtree}(R_j) = x$  and remaining  $u$  and  $z$  respectively.

Now, to pump down replace  $R_i$  by  $R_j$  giving  $uxz \in A$

to pump up  $k$  times, replace  $R_j$  by  $R_i$   $k$  times giving  $uv^kxy^kz \in A$

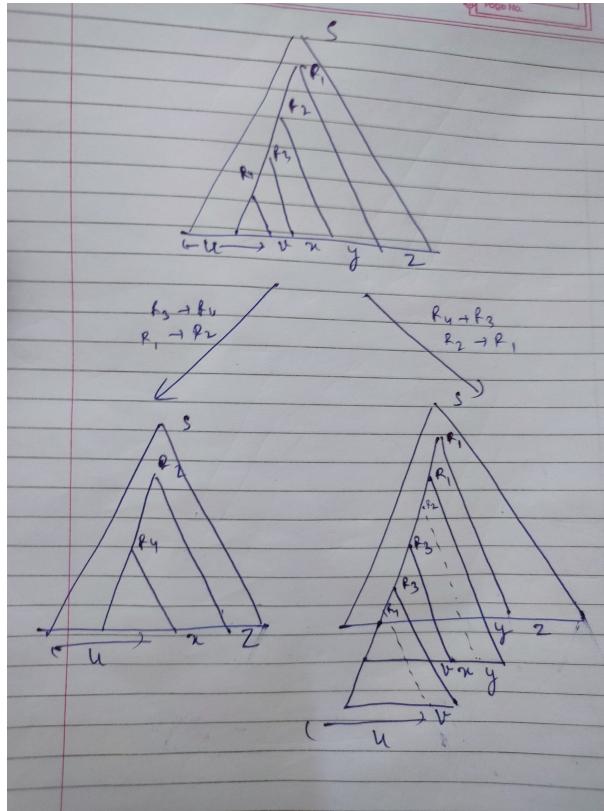
Also,  $v = a \neq \epsilon, y = b \neq \epsilon$

Moreover, since we have taken lowest repeating non terminal R, and we choose all 4  $R_i$  in the lower  $4|V| + 1$  non terminals, hence string generated by taking  $R_1$  as root (let w)  $|w| \leq b^{4|V|+1} = p$  and  $|vxy| \leq w \leq p$

Hence done.

## 2. Case 2: $\forall i \in \{1, 2, 3\} : R_i \rightarrow_* R_{i+1}b (b \neq \epsilon)$

Lets consider the following partition of  $s = uvxyz$ :



To pump down replace  $R_3$  by  $R_4$  and  $R_1$  by  $R_2$  giving  $uxz \in A$

To pump up k times, replace  $R_4$  by  $R_3$  k times,  $R_2$  by  $R_1$  k times giving  $uv^kxy^kz \in A$ .

Moreover, since we have taken lowest repeating non terminal R, and we choose all 4  $R_i$  in the lower  $4|V| + 1$  non terminals, hence string generated by taking  $R_1$  as root (let w)  $|w| \leq b^{4|V|+1} = p$  and  $|vxy| \leq w \leq p$   
Hence done.

### 3. Case 3: $\forall i \in \{1, 2, 3\} : R_i \rightarrow_* aR_{i+1} (a \neq \epsilon)$

This case is similar to last case just role of v is replaced by y. Performing the similar operations as that of last case will give  $uv^kxy^kz \in A \forall k \geq 0$

These are the only 3 cases possible since both v and y cannot be  $\epsilon$  simultaneously due to normal variant of pumping lemma.

## 7 Question 7

Give an example of a language that is not a CFL but nevertheless acts like a CFL in the pumping lemma for CFL (Recall we saw such an example in class while studying pumping lemma for regular languages).

Consider the following languages:-

1.  $L_1 = ab^n c^n d^n$

**Claim:**  $L_1$  is not CFL

**Proof:** Suppose  $L_1$  is CFL, consider the language  $L' = L_1 \cap b^* c^* d^* = b^n c^n d^n$

Then,  $L'$  would also be regular since it is intersection of a CFL and a regular language (proved in ques 2 that intersection of regular language and CFL is CFL). But it is proved in class that  $L'$  is not a CFL. Hence by contradiction  $L_1$  is not CFL.

2.  $L_2 = a^{k_1} b^{k_2} c^{k_3} d^{k_4} : k_1 \neq 1$

$L_2$  is CFL because it is union of 2 regular languages :  $b^* c^* d^* \cup a^2 a^* b^* c^* d^*$  that is regular and all regular languages are CFL.

3.  $L_3 = L_1 \cup L_2$

$L_3$  is not a CFL as  $L_3 \cap ab^* c^* d^* = L_1$  that is not a CFL and if  $L_3$  were CFL, it should have been CFL by closure of union on CFL.

Now, lets try to apply pumping lemma on  $L_3$ .

Let  $p = 2$

Consider  $\forall s \in L_3$

There are only 2 choices, either  $s$  is in  $L_1$  or  $L_2$  (as both have no intersection).

Lets consider both of the cases seperately.

1.  $s \in L_1$

Consider the partition of  $s = uvxyz$ , where  $u = v = x = \epsilon, y = a, z = b^n c^n d^n (n > 0 as |s| \geq 2)$

Now,  $\forall i \geq 0 : s' = uv^i xy^i z = a^i b^n c^n d^n$

If  $i = 1$  then  $s' = s \in L_1 \Rightarrow s' \in L_3$  otherwise  $s'$  is of the form  $a^{k_1} b^{k_2} c^{k_3} d^{k_4} : k_1 \neq 1$  i.e.  $s' \in L_2 \Rightarrow s' \in L_3$

$\Rightarrow \forall i \geq 0 s' \in L_3$

Hence  $L_3$  satisfies pumping lemma.

2.  $s \in L_2$

This case is simple as  $L_2$  is CFL, it should satisfy pumping lemma. Hence we are done.

Hence provided an NCFL that satisfies pumping lemma.