Introduction to Automata and Theory of Computation
COL352 - Assignment 3

Tanishq Dubey (2019CS51077), Satyam Kumar Modi
(2019CS50448), Rupanshu Shah (2019CS10395)

February 2022

Contents

1 Self-Referential CFG

Let G = (V, Σ, P, S) be a CFG that is not self-refrential,i.e. there does n't exist any non-terminal X for which $X \stackrel{*}{\Rightarrow} \alpha X \beta$ holds for any α, $\beta \neq \epsilon$.

We need to prove that L(G) is regular. For this, we prove that the CFG G can generate only finite strings and this will imply that L(G) is regular.

- **Base Case** For $n = 1$, we can have only a single production rule of the form $S \rightarrow a$ for some non-terminal S and terminal a. In this case only a is the string that the grammar can produce. Note that a production rule of type $S \rightarrow AS$ or $S \rightarrow SA$ is not possible because the grammar G is not self-refrential.
- Assume the induction to hold for n = m, i.e. we have total n production rules and we can have a maximum of k possible strings that the grammar can generate.
- **Inductive step** For n = m+1, i.e. we have total $n + 1$ production rules. The first rule must be of the form $S \rightarrow AB$ or $S \rightarrow a$ where S is the start state.
  - **Case-I** We have the rule $S \rightarrow AB$. Note that the rest m production rules cannot involve S in the right hand side of the rule because that will result in G being self-refrential. Now, the grammar G can be splitted into two grammar, first(say G_1) which has A as the start state and second(say G_2) which has B as the start state and the same production rules remain except $S \rightarrow AB$. Both G_1 and G_2 can't be self-refrential otherwise this would imply G is self-refrential. Now, since both of the grammars have n production rules, they must be generating finite strings (say k_1 and k_2). Clearly, we can have a maximum of $k_1 k_2$ possible strings that G can generate.
  - **Case-II** We have the rule $S \rightarrow a$ where a is a terminal symbol. Clearly, we can have only one possible string a. Thus, the grammar G generates finite strings.

  Thus, the induction holds and hence all non self-refrential grammar generate only finite no. of strings. This implies L(G) is finite and hence it is regular.

2 Intersection of a CFL and Regular Language

Let L_1 be a CFL and L_2 be the regular language. Since, L_1 is a CFL, we have a pushdown automata corresponding to L_1, say $P = (Q_1, \Sigma, \Gamma, \delta_1, q_0^1, F_1)$ which accepts by transitioning to final state and L_2 have a DFA $M = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$. To show that $L_1 \cap L_2$ is a CFL, we show the existence of a CFG for this. We construct the new PDA P' as follows:
$Q' = \{(q_1, q_2) | q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$
$\delta((q_1, q_2), a, A) = \{((q', \delta_2(q_2, a)), B) \mid (q', B) \in \delta_1(q_1, a, A)\}$
$q_0 = (q_0^1, q_0^2)$
$F = \{(q_1, q_2) | q_1 \in F_1 \text{ and } q_2 \in F_2\}$
The alphabet Σ and stack symbols Γ remains same.
Acceptance by the PDA P' is by acceptance by final state.

Proof: $L(G') = L_1 \cap L_2$

- **Case-I** First, we prove that $L(G') \subseteq L_1 \cap L_2$. Take a string $x \in L(G')$, we show that $x \in L_1 \cap L_2$. Since the PDA P' accepts x, it must have reached to some final state (q_1, q_2) where $q_1 \in F_1$ and $q_2 \in F_2$ in the run of the PDA P'. Lets say we have the following states to which the transition happens in the run of the PDA P', $(q_{i_1}, q_{j_1}), (q_{i_2}, q_{j_2}), .., (q_{i_k}, q_{j_k})$ where $q_{i_1}, .., q_{i_k} \in Q_1$, $q_{j_1}, .., q_{j_k} \in Q_2$ and $q_{i_1} = q_0^1$, $q_{j_1} = q_0^2$ and $q_{i_k} = q_1$, $q_{j_k} = q_2$. The states $q_{j_1}, .., q_{j_k}$ marks the run of the DFA M through the string x. Since, $q_{j_k} \in F_2$, DFA M must accept the string x. Also, the states $q_{i_1}, .., q_{i_k}$ mark the run of the PDA P through the string x and since $q_{i_k} \in F_1$, the PDA P must accept the string x. Thus, we have proved that $x \in L_1 \cap L_2$. This implies $L(G') = L_1 \cap L_2$.

- **Case-II** Now, we prove that $L_1 \cap L_2 \subseteq L(G')$. Let string $x \in L_1 \cap L_2$, we prove that $x \in L(G')$. Since, $x \in L_1$, lets say we have the following states through which the PDA P runs, $q_{i_1}, .., q_{i_k}$ where $q_{i_1} = q_0^1$ and $q_{i_k} \in F_1$. Similiarly, we have the following states through which the DFA M goes in the run of x, $q_{j_1}, .., q_{j_k}$ where $q_{j_1} = q_0^2$ and $q_{j_k} \in F_2$. When the string x is ran through the PDA P', the states to which it transitions must be $(q_{i_1}, q_{j_1}), (q_{i_2}, q_{j_2}), .., (q_{i_k}, q_{j_k})$ because of the way δ is defined for PDA P' and since we land up in a final state which is (q_{i_k}, q_{j_k}) as $q_{i_k} \in F_1$ and $q_{j_k} \in F_2$. (Note that there would be no problem with the stack, as the stack symbol in each transition remains the same as in PDA P). This proves that $x \in L(G')$. This implies $L_1 \cap L_2 \subseteq L(G')$.

From the above two cases, we have $L(G') = L_1 \cap L_2$.

3 Concatenation CFL

Let L be a CFL and L' be a Regular Language. Let M be an NPDA for L. Let A be a DFA for L'.
We create an NPDA for $L||L'$ by making the accept state of M as the start state of A.

More formally, Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ be an NPDA for L. Let $A = (Q', \Sigma', q_0', F', \delta')$ be a DFA for L'.
Define NPDA for $L||L'$: $M" = (Q", \Sigma", \Gamma", \delta", q_0", \perp", F")$ where:
$\Gamma" = \Gamma$,
$q_0" = q_0$,
$\Sigma" = \Sigma \cup \Sigma'$,
$\perp" = \perp$,
$Q" = Q \cup Q'$ where $q_i = q_0' \ \forall q_i \in F$,
$F" = F'$,
$\delta" = \delta \cup \delta_1$ where $\delta_1 \subseteq Q" \times \Sigma" \times \Gamma" \times Q" \times \Sigma"^*$ and
$\delta_1(q_j, \lambda, \perp") = (q_k, \perp") \iff \delta'(q_j, \lambda) = q_k; \ \lambda \in \Sigma'$.
This NPDA $M"$ first simulates NPDA M and then non-deterministically simulates DFA A after reaching accept state of NPDA M.

Let $w \in L||L'$. This implies $w = w_1 \cdot w_2$ where $w_1 \in L, \ w_2 \in L'$. Thus there is a run of NPDA $M"$ which on reading the string up-to w_1 goes to configuration $(q_0', \perp)$ and thereafter on reading remaining string w_2 goes to $(q_f, \perp)$ where $q_f \in F"$. Therefore, w is accepted by NPDA $M"$.

Let $w \notin L||L'$ This implies w cannot be broken into w_1,w_2 where $w_1 \in L_1, \ w_2 \in L_2$. We claim that there is no accepting run of NPDA $M"$ on string w.
Proof is by contradiction.
Let us assume there is an accepting run. Since $q_i = q_0' \ \forall q_i \in F$, therefore, each accepting run must pass through configuration $(q_0', \perp)$. Let w_1 be the string read up-to $(q_0', \perp)$, and q_2 be the remaining string of the accepting run. Thus we have $w = w_1 \cdot w_2$ where $w_1 \in L, \ w_2 \in L'$, a contradiction. Hence, there is no accepting run of NPDA $M"$ on string w i.e, NPDA $M"$ rejects string w

Therefore, NPDA $M"$ recognizes exactly the language $L||L'$. This implies $L||L'$ is a CFL.
Hence proved.

If L, L' both are CFLs, then also $L||L'$ is a CFL. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ be an NPDA for L. Let $M' = (Q', \Sigma', \Gamma', \delta', q_0', \perp', F')$ be an NPDA for L. Define NPDA for $L||L'$: $M" = (Q", \Sigma", \Gamma", \delta", q_0", \perp", F")$ where:
$\Gamma" = \Gamma \cup \Gamma'$,
$q_0" = q_0$,
$\Sigma" = \Sigma \cup \Sigma'$,
$\perp" = \perp' = \perp$,

Q" $= Q \cup Q'$ where $q_i = q'_0$ $\forall q_i \in F$,
F" $= F'$,
δ" $= \delta \cup \delta'$ This NPDA M" first simulates NPDA M and then non-deterministically simulates NPDA M' after reaching accept state of NPDA M. By same arguments as above, NPDA M" recognizes exactly the language $L||L'$. This implies $L||L'$ is a CFL.
Hence proved.

4 Cycle CFL

Let A be an arbitrary CFL with NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ that accepts by empty stack. We construct the NPDA $M' = (Q', \Sigma', \Gamma', \delta', q_0', \perp', F')$ for *cycle(A)* as follows:
We introduce a new dummy symbol Ω in the stack alphabet. Let $w \in A$. Consider any partition $w = w_1 \cdot w_2$. The word $w' = w_2 \cdot w_1 \in cycle(A)$. Now, when reading the word w', the NPDA M' non-deterministically speculates the symbol at the top of the stack if the NPDA M had read string w_1. Let's say the speculated symbol is λ_0. The NPDA M' initially adds $\lambda_0\Omega\lambda_0$ to the stack i.e., the contents of the stack initially are $\lambda_0\Omega\lambda_0\perp$. Now the NPDA M' simulates NPDA M and starts reading the word till the top of the stack becomes Ω.
Ω acts as a splitter to split the speculated symbols from the rest of the contents of the stack

When the top of the stack becomes Ω, it non-deterministically takes 2 paths:

1) It again speculates non-deterministically and adds one more letter λ_1 to both sides of the symbol Ω (first pop Ω then push λ_1, then push Ω and then again push λ_1). So, that the stack contents become $\lambda_1\lambda_0\Omega\lambda_1\lambda_0\perp$. Then it repeats the process of simulating NPDA M and reading remaining string.

2) It speculates that w_2 has ended and starts recognizing w_1 by simulating NPDA M and reading the remaining string. If the NPDA M pushes symbol λ_k to the stack and the first symbol right to Ω is λ_k, then pop λ_k. Continue this popping till the end of string or if there is no matching symbol at any point.
At the end of the word w' if the stack is $\Omega\perp$, then pop Ω and accept the string (acceptance by empty stack).

In essence, for $w' = w_2 \cdot w_1 \in cycle(A)$ where $w = w_1 \cdot w_2 \in A$, the NPDA M' speculates the word w_1' and after reading w_2 it checks if the speculated word concurs with the rest of the string w_1.

If $w' = w_2 \cdot w_1 \in cycle(A)$, then there exists a speculated word w_1' which concurs with w_1, and hence the NPDA M' accepts w'.

If $w' \notin cycle(A)$, then there is no break-up $w' = w_2 \cdot w_1$ such that $w_1 \cdot w_2 \in A$. We claim that NPDA M' does not accept such string w'.
Proof is by contradiction. Let us assume NPDA M' accepts string w'. This implies, there exists some speculated word w_1' such that $w' = w_2 \cdot w_1$ and w_1' concurs with w_1. This implies $w_1 \cdot w_2 \in A$, a contradiction. Therefore, NPDA M' does not accept such string w'.

Thus, NPDA M' recognizes exactly the language $cycle(A)$. Therefore, CFLs are closed under *cycle* operation.
Hence proved.

5 Pumping Lemma

Consider the language L = $\{a^i b^j c^k d^l \mid$ either $i = 0$ or $j = k = l$ $\}$. First, we prove that the language L is not a CFL. For this, let us assume that we have a CFG for L with k non-terminals. Now, take a string $z = a^3 b^{2k} c^{2k} d^{2k}$, we can divide z as $uvwxy$ such that vwx lies within b^{2k} and $|vx| \neq 0$. Note that uwy in this case does n't lie in L as it would have less number of $b's$ in comparison to c and d. Thus, this contradicts that L have a CFG. So, we have proved that L is not a CFL.
Now, we show that the pumping lemma is satisfied in this case. We have the pumping length for this language as p = 3.

- **Case-I** If the string x = $a^i b^j c^k d^l$ such that $i \neq 0$ and $j = k = l$, then take $u = \epsilon$, $v = a$, $w = \epsilon$, $x = \epsilon$, $y = a^{i-1} b^j c^k d^l$. Note that here $|vx| = 1$, $|vwx| = 1 < p$(Pumping length). Now for all $n \geq 0$, we have $uv^n wx^n y = a^{n+i-1} b^j c^k d^l$. Clearly, $uv^n wx^n y$ belongs to the language L for all $n \geq 0$ as $j = k = l$. Thus, the pumping lemma is satisfied in this case.

- **Case-II** If the string x = $b^j c^k d^l$, i.e. $i = 0$. Here, take $u = \epsilon$, $v = b$, $w = \epsilon$, $x = \epsilon$, $y = b^{j-1} c^k d^l$.Note that here $|vx| = 1$, $|vwx| = 1 < p$(Pumping length). Now for all $n \geq 0$, we have $uv^n wx^n y = b^{n+j-1} c^k d^l$. Clearly, $uv^n wx^n y$ belongs to L as $i = 0$. Thus, the pumping lemma is satisfied in this case too.

So, we have proved that L is not a CFL but still the pumping lemma for CFL is satisfied here.

6 Reverse String CFL

Claim 5.1 Let us have the language $A = \{wtw^R | w, t \in 0, 1^*; |w| = |t|\}$. Then A is not a context free language.

Proof By Pumping Lemma:

Let us take a pumping length p, and the word $w = 0^{2p}0^p1^p0^{2p}$. Clearly, $|w| \geq p$ and $w \in A$. Now, we will take cases on $w = uvxyz$ where $|vy| > 0$ and $|vxy| \leq p$.

* **Case I:** vxy is a uniform string i.e all characters in it are same. Now, consider $w = 0^{3p}1^p0^{2p}$, we have three sub-cases -
  - First, we take vxy completely in 0^{3p}, now if we pump with $i = 2$, we get v^2xy^2. Clearly, the last $2p$ characters in $w' = uv^2xy^2z = 0^{3p+x}1^p0^{2p}$, where $x > 0$ are 0. Let us say this new string $\in A$, $w' = mnm^R$. Now, clearly, m^R cannot be of length more than $2p$ because then m must contain a 1 at its $2p + 1$ position, which is not possible since the $2p + 1$-th character in w' is clearly 0. Now, let us take $|m^R| = k \leq 2p$, then we have $m^R = m = 0^k$. Now, for w' to belong in A, we need $|m| = |n|$, but this is not true because $|m| = k \leq 2p$, but $|n| > 2p$, since n includes 0^p1^p and the extra pumped up characters. Hence, we conclude that $w' \notin A$.
  - Secondly, let us assume, vxy is completely in 1^p. Now, let us pump up with $i = 2$ and we have $w' = uv^2xy^2z$. The word will be of the form $w' = 0^{3p}1^{p+x}0^{2p}$, where $x > 0$. Let if possible $w' = nmn^R \in A$, then clearly, w^R cannot be of length $> 2p$. because if it was, it will contain a 1 at the $2p+1$-th position from the end, and by similar arguments as in the previous case we can conclude. Now, we have $n^R = n = 0^k$, where $k \leq 2p$. Again, using similar arguments in the previous case, we conclude that $|m| \neq |n|$. Hence, we have shown $w' \notin A$.
  - Thirdly, let us assume vxy is completely in the last 0^{2p}. We now have $w' = 0^{3p}1^p0^{2p+x}$, where $x > 0$. Now we can easily choose pumping constant i to be large enough for $2p+x \geq 3p$, and then we can proceed with the same arguments used in previous two cases by showing that if $w' = mnm^R$, then $|m| \leq 3p$, and if that is the case, then we cannot have $|m| = |n|$ which shows that w' cannot be in A.
* **Case II:** vxy is not uniform and contains both 0 and 1. We have two sub-cases-
  - First, vxy is straddled between the first 0^{3p} and 1^p. Now, if we pump any such vxy to v^2xy^2, using pumping constant $i = 2$ we get $w' = uv^2xy^2z = 0^{2p}0^{p+x}1^{p+y}0^{2p}$, where $x, y > 0$.

Therefore, let if possible $w' \in A$, $w' = nmn^R$. Now, clearly, n^R must have a length less than $2p$, otherwise n will need to have a 1 at $2p+1$ th place which is a contradiction. Thus, we conclude $|n| = |n^R| = k \leq 2p$. Now, clearly, the length of m is greater than $2p$ because because it contains at least 0^p1^p and the pumped up characters. Thus, we conclude that w' cannot be in A.

- Let us take the case where vxy is straddled between 1^p and 0^{2p}. After pumping up with some $i > 0$, we will get $w' = 0^{3p}1^{p+y}0^{2p+x}$, where $x, y > 0$. We can take i large enough such that $2p + x \geq 3p$ and then we can argue that if $w' = nmn^R$, then $n \leq 3p$ and hence we cannot have $|n| = |m|$. This is again a very similar argument as made in the above cases, hence we omit the description. Finally we can conclude w' does not belong to A.

Through the above cases and sub-cases we have shown that for $w = 0^{2p}0^p1^p0^{2p} \in A$ any u, v, x, y, x such that $w = uvxyz$ where $|vy| > 0$ and $|vxy| \leq p$, we can always find an i, such that uv^ixy^iz does not belong to A. Therefore, by pumping lemma for CFL, we conclude that A is not a CFL. ■.

7 Stronger Pumping Lemma

Claim 6.1: Let us say we have a context free grammar G in Chomsky-Normal form with $|V|$ non terminals, then for any word w with $|w| \geq 2^{2|V|}$, the height of the parse tree will at least be $2|V|+1$ and there will at least be one non-terminal A that is repeated thrice in the derivation of w.

Proof by Deduction:

Note that for a grammar in CNF form, the i^{th} level of the parse tree will have at most 2^i nodes where $i \geq 0$ is not the last level with all terminals. For, $i = h$, where h is the height of the parse tree, we know that this level will comprise of the word w and since CNF grammar only have transitions of the form $A \to a$, when RHS contains a terminal, we can conclude for height h, the maximum length the word can have is equal to the maximum possible nodes at level $h-1$ i.e 2^{h-1}. As a consequence, we can say that if the word length is $\geq 2^{h-1}$ then the parse tree must have at least h height. Now, we take $h = 2|V|+1$ and the first part of the claim follows.

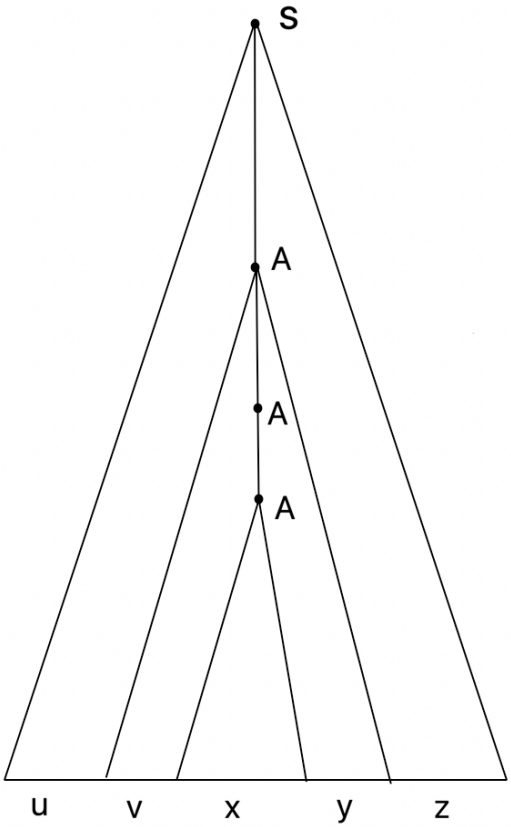
For the second part, assuming we have $|V|$ non-terminals in the grammar G and we have a string w of length $\geq 2^{2|V|}$, we take path P in the parse tree of w of maximum length. From above deduction, we know that P will have atleast a length $\geq 2|V|+1$. Thus, from pigeon hole principle, there must exist a non-terminal A in the path P that repeats thrice. ■

Claim 6.2: If L is a CFL with $|V|$ terminals, then there is a number $p = 2^{2|V|}$ where if w is any string in L of length at least p then w may be divided into five pieces $w = uvxyz$, satisfying the conditions that for each $i \geq 0$, $uv^ixy^iz \in L$ with $v \neq \epsilon$, and $y \neq \epsilon$ and $|vxy| \leq p$.

Proof:

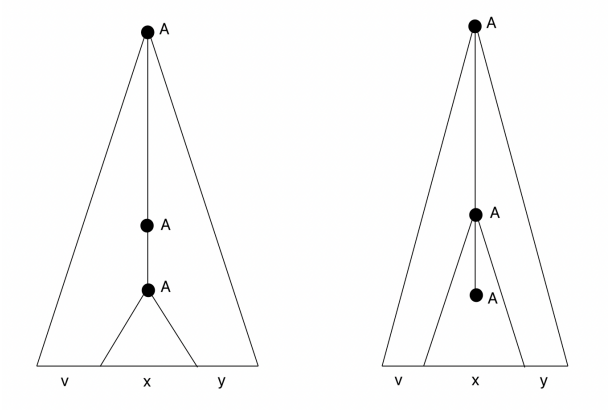
Existence of Split: Consider the diagram of the parse tree with a path containing three non-terminals A and the strings u, v, x, y, z. We can see that $A \xrightarrow{*} x$ is a valid derivation, we also note that $A \xrightarrow{*} vAy$ is also a valid derivation. Note that, A in the LHS in this case can be either the first one or the second one, as explained in the second diagram given on the next page. We can also clearly see that $S \xrightarrow{*} uAz$ is a valid derivation. Hence, using the following derivation with $S \xrightarrow{*} uAz \xrightarrow{*} uvAyz \xrightarrow{*} uv^2Ay^2z \xrightarrow{*} uv^3Ay^3z$ and so on. After $i \geq 0$ such steps we can use $A \xrightarrow{*} x$ to terminate the derivation. Hence, we have proved that there exists a way to split the word w int the way provided in the claim when $|w| \geq p = 2^{2|V|}+1$, such that $uv^ixy^iz \in L$.

Lower Bound: Now, to bound v and y, we will take the situation when they are as small as possible, hence, we consider the case where

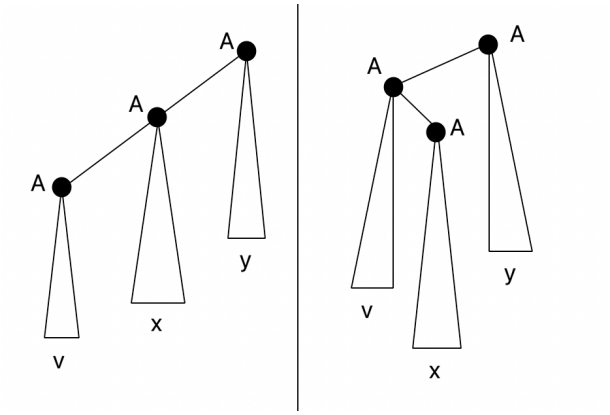


An example parse tree for w

the first and the third A are as close as possible in the parse tree, because if not, we can easily see that the length of v or x will be more. Now, there can be multiple cases and we will show in all such cases it is possible to create v, x, y split such that $|v| \neq \epsilon$ and $|y| \neq \epsilon$. Let us take the case where $A \to AB \to ACB$, i.e the A-s are the left children in the derivation as shown in the left of the bottom figure on the next page. Clearly, in this case as shown in the figure, we can find non zero length v, x, y with using the third A for x. Similarly, we have shown for the case where the first A is the left child and the second A is the right child. Clearly, in all 8 cases we have $|v|, |y| \neq \epsilon$.

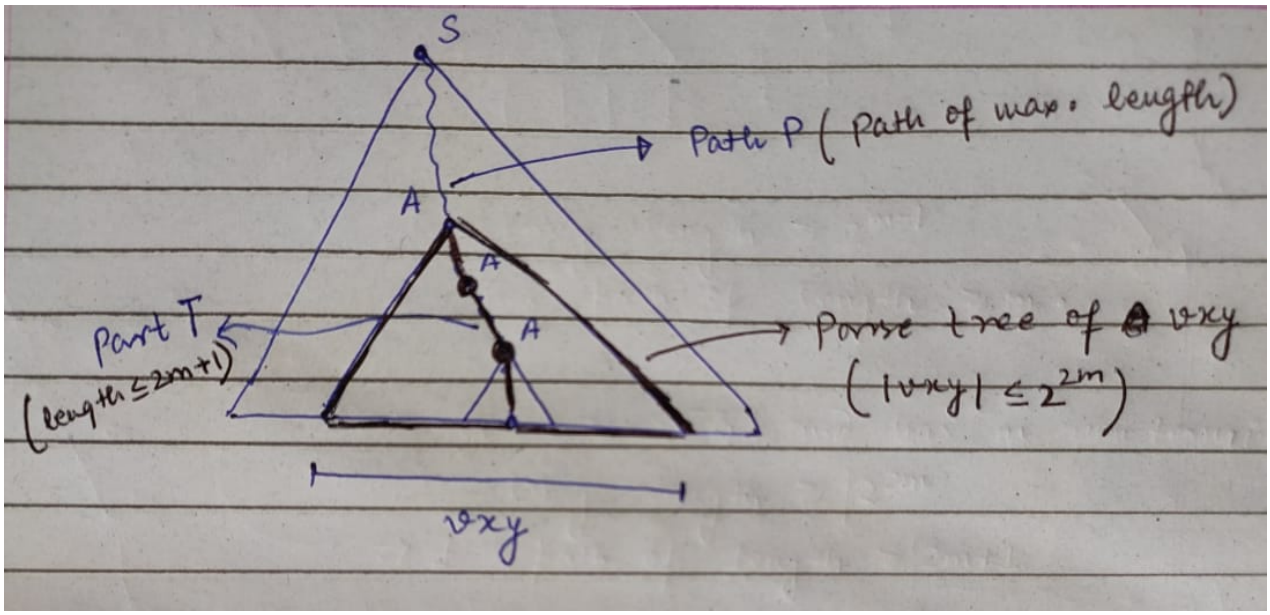


Possible parse trees for vxy



Lower bound on v and y

Upper Bound: We find out the non-terminal A that repeats thrice in the longest path P in the following way. We pick the bottom part T of P of length $2m + 1$ (such a part T exists as length of $P \geq 2m + 1$), from the pigeon hole principle some non-terminal A must have repeated thrice in this part T. Now, we can argue that length of such a bottom part where a non-terminal repeats thrice must be $\leq 2m + 1$. This implies that the parse tree that the first A in the path P produces must have a maximum height of $2m + 1$ and this imply that $|vxy| \leq 2^{2m}$. ■


S
Path P (path of max. length)
A
A
A
Part T
(length ≤ 2m+1)
Parse tree of vxy
(|vxy| ≤ 2^2m)
vxy