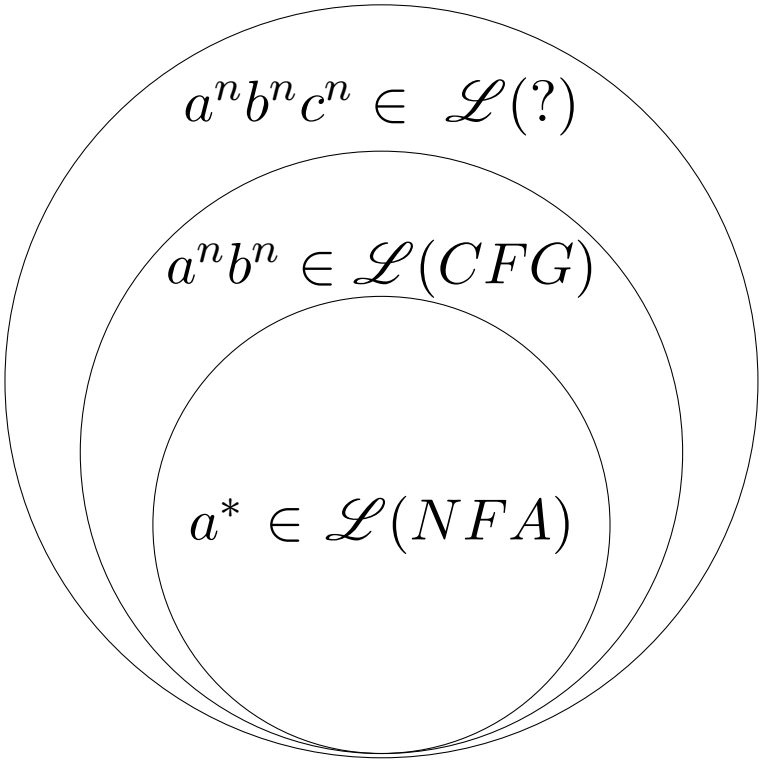
CS 4510 Automata and Complexity September 25th 2023

Lecture 9: The Pumping Lemma for Context Free Languages

Lecturer: Abrahim Ladha *Scribe(s): Rishabh Singhal*

1 Introduction

Lets give a high level picture of the space of languages as we know it so far. In the beginning, there was only the regular languages which had things like a^*. Next we found a class of languages called the CFLs which included things which were specifically not regular, like $\{a^n b^n \mid n \in \mathbb{N}\}$. Today, we will show the existence of languages outside of $\mathscr{L}(CFG)$, with our first example being $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.



We will prove that $L = \{a^n b^n c^n | n \in N\} \notin \mathscr{L}(CFL)$, but for now, lets just assume so. Consider the following two languages. $L_1 = \{a^m b^n c^n \mid m, n \in \mathbb{N}\}$ and $L_2 = \{a^n b^n c^m \mid m, n \in \mathbb{N}\}$. Certainly they are context-free, since we can produce grammars for each. Here is the grammar for L_1:

- $S \to AT$
- $A \to aA \mid \varepsilon$
- $T \to bTc \mid \varepsilon$

And similarly, for L_2:

- $S \to RC$
- $C \to cC \mid \varepsilon$
- $R \to aRb \mid \varepsilon$

However, notice that $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$. The number of a's equals the number of b's, but then the number of b's must equal the number of c's. So transitively, we get our triple threat language.

Assume to the contrary CFLs are closed under intersection. Since L_1, L_2 are CFLs, so should be $L_1 \cap L_2$. But $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ a language we are going to pump and prove to not be context-free, a contradiction. Assume to the contrary CFLs are closed under complement, then $\overline{L_1}$, $\overline{L_2}$ would be context-free, and so would $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$. However, we will prove it is not, a contradiction. CFLs are closed under union, Kleene star, and concatenation, but not under intersection or complementation.

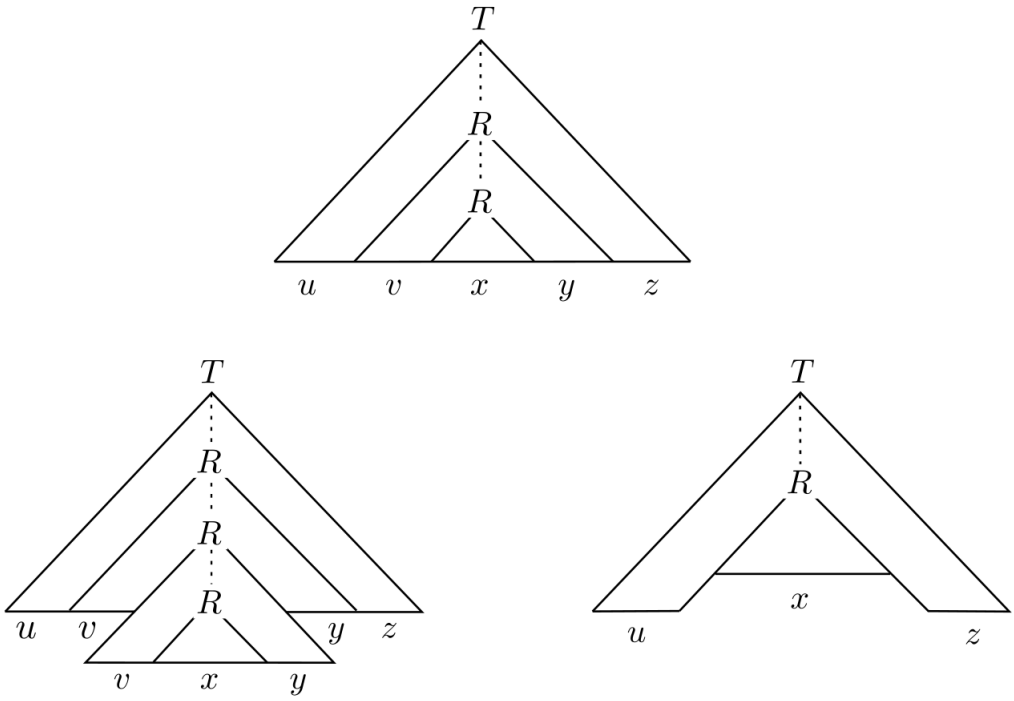
2 Parse Trees

A grammar is a finite device, like any other, but it has the ability to produce infinitely long strings. There must be some periodicity to this device. We derive pigeonhole conditions and prove a pumping lemma for the context-free languages. Recall that parse trees exist. This is what we will do our combinatorial "pumping" on. Consider the production of a super long word with a huge parse tree relative to the size of the grammar. If the string is long enough, then a non-terminal is repeated twice on some path from the start non-terminal, by the pigeonhole principle. I claim we can perform the following surgery on the parse tree. We may cut and copy the different subtrees with our repeated non-terminal. If some $uvxyz \in L$, then $uv^i xy^i z \in L$ as well.

Now lets derive the exact conditions for this to be true. Let b be the maximum length of the RHS of the largest rule of G. For any parse tree of G, a single step from the start has at most b children. Two steps has at most b^2, and so on. It may help to recall the derivation of the Master Theorem from Algorithms. If the tree height is h, the maximum length of the string it can derive is $|w| \leq b^h$. Conversely, if a generated string has length $b^h + 1$, its parse tree has height $h + 1$. Choosing a string of length greater than or equal to $b^{|V|} + 1$ will guarantee that our height is $\geq |V| + 1$, which by the pigeonhole principle, implies some non-terminal is repeated. For convention set $p = b^{|V|+1}$, so our parse tree has height greater than or equal to $|V| + 1$. Its true that we could satisfy this with just $b^{|V|} + 1$, but since $b^{|V|+1} > b^{|V|} + 1$, our tree height is still $\geq |V| + 1$. We maintain this convention to make applying the lemma easier. Since G only has $|V|$ non-terminals, some non-terminal is repeated twice (or more) by the pigeonhole principle along a path from the start non-terminal.

- We want to produce a string such that its parse tree has height sufficient to be pumped, so we may choose any $|s| \geq p$.
- For $s = uvxyz$ as shown in the figure, we want v, y to both not be empty, so we may pump something nontrivial. We condition this as requiring $|vy| > 0$.

- There may be many more repeated non-terminals along the height of our tree, but we want to focus on to the last repeated one. In the picture we have $R \stackrel{*}{\Rightarrow} vRy \stackrel{*}{\Rightarrow} vxy$. We want this tree to be at most $|V| + 1$ high, so we require that $|vxy| \leq b^{|V|+1} = p$.



3 Recipe of Pumping Lemma

Recall that like the pumping lemma for regular languages, it is easy to miss one case here or there, or misapply it. Following this recipe exactly is the best way to ensure your proof is correct.

1. Assume the contrary L is context-free with pumping length p.
2. Choose some $s \in L$, $|s| \geq p$
3. List cases $\forall u, v, x, y, z$ with $s = uvxyz$ subject to $|vxy| \leq p$ and $|vy| > 0$.
4. For each case choose an i such that $uv^ixy^iz \notin L$
5. Conclude that the language must not be context-free.

Also unlike the pumping lemma for regular languages, we are not going to be able to list out every case mathematically, using superscripts as lengths. Instead, you have to think logically. Construct several "meta-cases" in which each sub-case is easy to argue. The meta-cases should be defined so that its obvious any case must fall into one meta-case or the other.

4 Examples

Lets do an example.

4.1 $\{a^n b^n c^n \mid n \in \mathbb{N}\}$

- Assume to contrary L is context-free with pumping length p.
- Choose $s = a^p b^p c^p$. Certainly $s \in L$ and $|s| \geq p$.
- We have two general cases:
  - v **and** y **each contain only one type of symbol.** There is three of $\{a, b, c\}$, only two of the $\{v, y\}$. So $uv^2xy^2z \notin L$ cannot contain an equal number of a's, b's, and c's. Some letter will not increase.
  - **If either** v **or** y **contains more than one symbol**. then uv^2xy^2z contains symbols out of order, and thus, not be in our language.
- It follows that L is not a CFL.

Lets give some remarks about the proof. First, for choosing s, before for regular languages, a bad choice of s just meant we had many cases but it could still be correct. For pumping context-free languages, a bad choice of s likely will not allow you to even complete the proof. Many obvious or first choices end up being pumpable, when we want to show they are not. Try to choose a string which is barely in the language. One where even the smallest perturbation brings it out. Secondly, Notice that we didn't really have to apply the fact that $|vxy| \leq p$. This is sometimes too fine-grained. This would eliminate the case where v may contain a's at the same time y may contain c's. Our meta cases are so general, these are absorbed. Finally, notice how we defined our meta cases as logical complements of each other. Either both v and y contain one symbol, or maybe one of them is a mix of symbols. Any possible cases must fall into one of those two huge general categories. If we tried to do it any other way, we may get dozens of cases, many of which are identical.

4.2 $\{a^i b^j c^k \mid 0 \leq i \leq j \leq k \in \mathbb{N}\}$

- Assume to the contrary that L is regular with pumping length p. Choose $s = a^p b^p c^p$. Clearly $s \in L$ and $|s| \geq p$. We have a few cases
- **both** v **and** y **are of only one type of letter.** We need to pump up or down depending upon what the letters actually are, so we divide into further sub cases.
  - If a's do not appear we pump down. Choose $i = 0$, so uv^0xy^0z has more a's than b's.
  - If b's do not appear:
    - But a's appear choose $i = 2$, so uv^2xy^2z has more a's than b's.
    - But c's appear choose $i = 0$, so $u\ v^0x\ y^0z$ has more b's than c's.
  - If c's do not appear choose $i = 2$, so uv^2xy^2z has more a's or b's than c's.

- **if v or y contain more than one type of symbol** we choose $i = 2$, as uv^2xy^2z will be out of order.
- We conclude that L is not CFL

Convince yourself that the cases listed are total. By out of order, we mean for example that $v = a^k b^j$. Then $uvv... = a^{p-k}a^k b^j a^k b^j ... = a^p b^j a^k b^j ...$ so somehow this string contains a's after the b's, and wouldn't be in the language.

4.3 $\{ww \mid w \in \Sigma^*\}$

There are many bad choices of s for this language. Choosing a few may lead you to a good one. I have the premonition to know we are going with a good choice of s here.

- Assume to contrary L is context-free with pumping length p. choose s = $0^p1^p0^p1^p$ and confirm $s \in L$ and $|s| \geq p$.
- **If vxy is all in the first or last half**, Choose $i = 2$. let $uv^2xy^2z = w_1w_2$ with $|w_1| = |w_2|$. If vxy was in the first half, w_1 begins with 0 and w_2 begins with 1. If vxy was in the second half, w_1 ends with 0 but w_2 ends with 1. For either of those $w_1 \neq w_2$.
- vxy **straddles the midpoint.** Pump down so $uv^0xy^0z = 0^p1^k0^j1^p$ where k, j cannot both be p. So now $0^p1^k0^j1^p \neq ww$ unless $k = j = p$ and this is impossible since $|vy| > 0$.
- We conclude that L is not CFL