CSE 322 - Introduction to Formal Methods in Computer Science
The pumping lemma for CFLs

Dave Bacon
Department of Computer Science & Engineering, University of Washington

Way back a along time ago, we derive a little thing called the pumping lemma for regular languages. Is there an equivalent idea for context-free languages? The answer, it turns out, is yes. This is what we'll talk about today: the pumping lemma for context-free languages.

Instead of delaying. Let's just state the theorem outright:

> **Pumping Lemma for CFLs:** If L is a context-free language, then there is a number p where if s is any string in L, $s \in L$ of length at least p ($|s| \geq p$), then s may be divided up into five parts: $s = uvxyz$ which satisfy the conditions
>
> 1. for each $i \geq 0$, $uv^i xy^i z \in L$
> 2. $|vy| > 0$
> 3. $|vxy| \leq p$

Cool, now why the heck is this true? Let's give you a fairly informal idea of why it is true and then prove it more rigorously. This is a problem where a picture is worth a thousand words (or in this case a thousand mathematical statements?) This picture is given in Fig. 1:

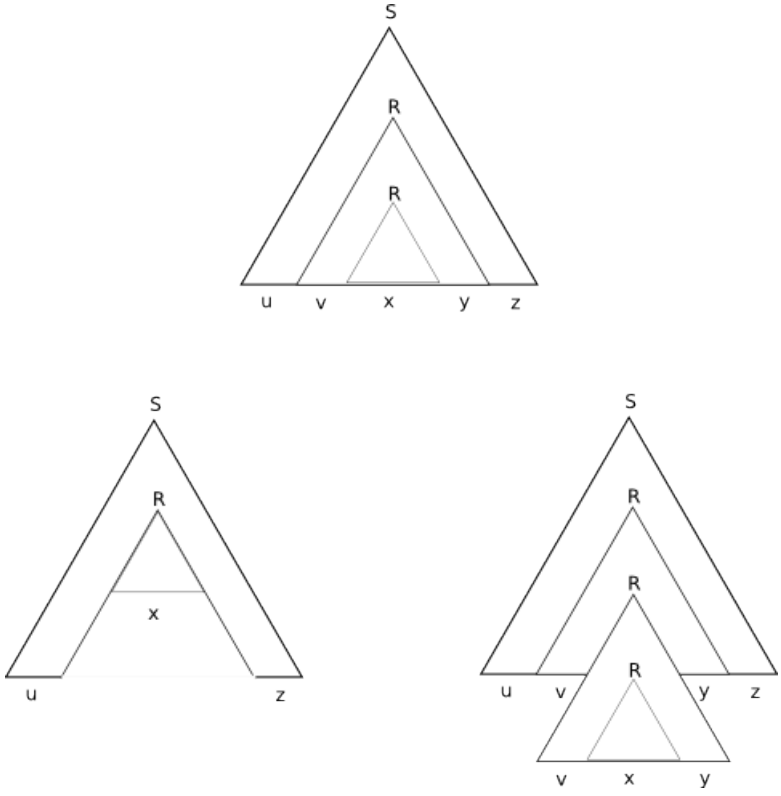

FIG. 1: The intuition behind the CFL pumping lemma.

Suppose that you have some long string s in a CFL L. Imagine the parse tree for this string under a CFG for this language. Suppose that the CFG has $|V|$ variables. Suppose the string is long enough that there is a path through the parse tree that is of length greater than $|V|$. If this is true, then one of the variables must be repeated in that path. If this is true, then we can divide up the string in a manner like that of Fig 1. From there you can get an idea about why the pumping lemma is true: you can perform a sort of surgery on the graphs such that the pumping

lemma holds. You can, for example, remove the part of the derivation originating from the first repeat of the variable and start with the part of the tree that comes from the second variable. This will give you the $i = 0$ case of part 1 of the CFL pumping lemma.

Okay now lets make this formal (words you dread, I'm sure.) Let G be a CFG for the CFL L of which we are going to prove the pumping lemma for CFLs.

First some observations. Suppose that b is the maximum number of symbols in the right-hand side of a rule of G. We will assume that b is greater than or equal to 2. If we think about parse trees generated by this grammar, any node in the tree can have at most b children. If the parse tree is of height h, then the string generated can be of length at most b^h. Or, to put this another way, if a generated string is at least $b^h + 1$ long, then each of its parse trees must be at least height $h + 1$ high.

Let $G = (V, \Sigma, R, S)$. We are going to prove the pumping lemma by setting p, the pumping length, to be $b^{|V|+1}$. So if a string is of length at least p, then we know that its height must be at least $|V| + 1$ (since $b^{|V|+1} \geq b^{|V|} + 1$.) This is great! Because it means that we can use the pigeon hole principle.

Okay now lets show how we can pump any string of length greater than or equal to p. For the string s, let τ be a parse tree for s. Actually don't just make it any parse tree, make it the parse tree with the smallest number of nodes (if the grammar is ambiguous.) We know that the parse tree must be at least $|V| + 1$ high, so it must contain a path from the root to a leaf of length at least $|V| + 1$. That path has at least $|V| + 2$ nodes, one at a terminal, and all of the rest are variables. Thus, since there are only $|V|$ variables there must be at least one variable, call it R, which is repeated during this path (pigeon hole principle.) Lets assume for convenience that this R is the variable that repeats among the lowest $|V| + 1$ variables in the path.

Okay, now we have a path and two repeated variables on this path. We can then divide up s into five chunks, as described in Fig. 1: $s = uvxyz$. Each occurrence of R has a subtree under it, generating a part of the string. The part under the lower occurrence of R we call x. The upper occurrence generates vxy. Note that R generates both x and vxy. If we take the upper part of the tree and replace the $R \stackrel{*}{\Rightarrow} vxy$ with $R \stackrel{*}{\Rightarrow} x$, we will obtain a derivation of the string uxz. If we take the part of the tree $R \stackrel{*}{\Rightarrow} x$ and replace it by $R \stackrel{*}{\Rightarrow} vxy$, we obtain $uvvxyyz$. Clearly (formally by induction) we see that condition 1 holds.

Now why do the other conditions hold (2 and 3 from the lemma statement)? Well condition 2 says that both v and y are not ε. If v and y were both ε, then we could substitute the smaller tree for the larger tree in our parse tree. This would then produce a derivation of s which had fewer nodes. But this contradicts our assumption, that τ is the smallest tree from which we can generate s. Thus it must be true that both v and y are not ε.

Finally to get condition 3, we need to be sure that vxy has length at most $p = b^{|V|+1}$. Why is this true? In the upper occurrence of R we generate vxy. We chose R so that both occurrences fall within the bottom $|V| + 1$ variables on the path, and we chose the longest path in the parse trees, so the subtree originating at R which generates vxy is at most $|V| + 1$ high. A tree of this height can generate a string of length at most $b^{|V|+1}$ which is p.

I. EXAMPLE

Lets show how to prove a language is not context free using the pumping lemma for CFLs. Consider the language

$$L = \{a^i b^i c^i | i \geq 0\}$$

We will show that L is not context-free by showing that it cannot satisfy the pumping lemma.

Let p be the pumping length of the CFL pumping lemma. Given this length p, consider the string $w = a^p b^p c^p$. Clearly w is in L and is of length greater than or equal to p. Thus, if L were context-free it must be possible that all three conditions of the pumping lemma hold. We will now show that this is not possible (and therefore that L is not a CFL.)

We are dividing $a^p b^p c^p$ into a string of the form $uvxyz$. Consider the case where v and y contain only one type of alphabet symbol. Then uv^2xy^2z cannot contain equal numbers of as bs and cs because both v and y cannot be empty. Thus uv^2xy^2z is not in L. This contradicts condition 1. If either v or y contain more than one type of symbol, then uv^2xy^2z may contain equal numbers of as, bs, and cs, but they will not be in the correct order (since squaring one must produce a wrong order substring.) Hence uv^2xy^2z is not in L and condition 1 is violated.

Thus we have seen that no matter how we divide up the string into $uvxyz$, we always violate one of the three conditions. Therefore L does not satisfy the pumping lemma for CFLs. Therefore L is not a CFL.

II. ANOTHER EXAMPLE

Let $L = \{ww | w \in \{0,1\}^*\}$. Assume that L is context-free and thus the pumping lemma for CFLs holds. Let p be the pumping length.

Choose the string $s = 0^p1^p0^p1^p$. Let us show that this string cannot be pumped. First note that $|s| \geq p$ and $s \in L$. We are dividing s as $s = uvxyz$. Condition 3 of the pumping lemma says that $|vxy| \leq p$. First lets show that the vxy part of the string must straddle the middle of the s. Why? Suppose that vxy appears in the first half of the s. Then uv^2xy^2z would contain a 1 at the beginning of the second half of the string. But since s starts with 0 this will result in a contradiction with this string being in L. Similarly suppose that vxy appears in the second half of s. Then uv^2xy^2z would move a 0 into the end of the first half, which would again yield a contradiction with this string being in L. Thus vxy must straddle the middle of s. If it straddles the middle, vxy must be made up of 0^i1^j for $i + j \leq p$ and $vy \neq \varepsilon$. Thus uxz must be of the form $0^p1^k0^l1^p$. Both k and l cannot be p. This string is not of the form ww, so it is not in L. Thus we have arrived at a contradiction with the pumping lemma. Hence L is not context-free.