4.3 CFG = PDA

Since we know by Definition 11 that a language is context-free if there exists a context-free grammar generating the language, we can combine the previous two lemmas to get the main result of this section.

Theorem 23. *A language A is context-free if and only if there exists a pushdown automaton $\mathcal{M}$ such that $L(\mathcal{M}) = A$.*

Proof. ($\Rightarrow$): Follows from Lemma 19.

($\Leftarrow$): Follows from Lemma 21. □

We can think of this result as the final piece to obtain the context-free analogue of Kleene's theorem for the regular languages. Since context-free grammars generate context-free languages, and since context-free grammars can be converted to pushdown automata (and vice versa), both models correspond to the exact same language class. Unfortunately, this result doesn't get a nice name like Kleene's theorem did, but perhaps the lack of a name is justified when you consider the diagram we get isn't as interesting as the one we had for the regular languages:[7]

CFG ⟷ PDA

However, we're not yet finished. As a consequence of the equivalence between context-free grammars and pushdown automata, we obtain an important corollary that relates the class of context-free languages to the class of regular languages.

Corollary 24. *Every regular language is also a context-free language.*

Proof. Every regular language is recognized by some finite automaton. Since a finite automaton is a pushdown automaton that does not use the stack, every regular language is also accepted by some pushdown automaton. Therefore, every regular language is context-free. □

Of course, we already know that there exist some context-free languages that are not regular, so this inclusion only works in one direction.

5 Proving a Language is Non-Context-Free

At the end of our discussion on regular languages, we saw that there exist certain languages that are nonregular, and we also saw that we can prove a language is nonregular by using the pumping lemma. One of the biggest obstacles we observed that results in a language being nonregular was, broadly speaking, having to count or otherwise keep track of symbols. Fortunately, by augmenting our machine model with a stack and creating a pushdown automaton, we were able to overcome this obstacle. Surely, this means that we can now recognize any language we want, right?

Well, not exactly. While the stack goes a long way in helping us to recognize more than just the class of regular languages, it isn't the magic solution we need in order to recognize *any* language. Consider, for example, the language

$$L_{\mathtt{a=b=c}} = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}.$$

We know that a pushdown automaton can accept words of the form $\mathtt{a}^n\mathtt{b}^n$ by pushing one symbol to the stack for each a that is read, and then popping one symbol from the stack for each b that is read. When it comes to recognizing words of the form $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, however, we run into a problem: after we read all of the bs in the word, our stack will be empty and we will therefore have forgotten the value of n by the time we have to count the cs! We also can't cheat our way around this problem by, for example, pushing two symbols to the stack for each a we read; if we try that, then reading either b or c would require us to pop the same symbol, and we can draw a conclusion that such an approach would result in the pushdown automaton accidentally accepting words where bs and cs are either out of order or having mismatched counts.

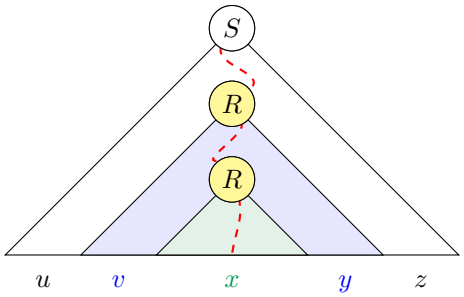
[7] Not exactly a *Scutum Fidei* as before, but maybe a *Gladius Fidei*?

Thus, there do indeed exist languages that are not context-free, and so we require a technique to prove the non-context-freeness of a language. Fortunately, we're mostly familiar with such a technique already: the pumping lemma for regular languages is a special case of the more general pumping lemma for context-free languages.

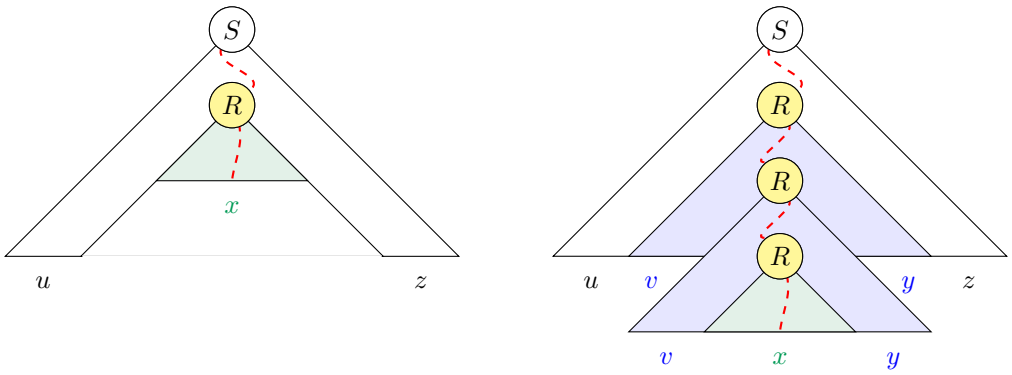
5.1 The Pumping Lemma for Context-Free Languages

You might be wondering at this point what we mean by the pumping lemma for regular languages being a "special case". Since regular languages are, in a sense, simpler than context-free languages, our formulation of the pumping lemma for regular languages was accordingly simpler: pumping the middle portion of any sufficiently-long word in a regular language results in us obtaining another word that also belongs to the language. Pumping this middle portion of the word essentially corresponds to us traversing a loop somewhere in the finite automaton recognizing the language.

With context-free languages, however, we can't just pump one portion of the word. To understand why not, recall that we can represent the derivation of a word in a context-free language using a parse tree. If our word is sufficiently long,[8] then the parse tree will be rather deep. Then, since we have only a finite number of both rules and nonterminal symbols, the pigeonhole principle tells us that there must exist some path from the root of the parse tree to a leaf of the parse tree where some nonterminal symbol R appears more than once along that path.



Suppose, as we did in the illustration, that we decompose our word w into five parts, denoted $uvxyz$. If we then take the subtree rooted at the first occurrence of R, we can repeat this subtree zero or more times by appending the subtree to the other occurrences of R. In essence, we are pumping the segment of the subtree between both occurrences of R when we perform this repetition, and as a result, we are pumping the v and y portions of the word simultaneously.



With this idea in mind, the formal statement of the pumping lemma for context-free languages is quite similar to that of the pumping lemma for regular languages, modulo the appropriate changes.

[8] "Sufficiently long" in this context is measured in terms of both the maximum number of symbols on the right-hand side of any rule of the context-free grammar and the size of the set of nonterminal symbols.

Lemma 25 (Pumping lemma for context-free languages)**.** *For all context-free languages L, there exists $p \geq 1$ where, for all $w \in L$ with $|w| \geq p$, there exists a decomposition of w into five parts $w = uvxyz$ such that*

1. *$|vy| > 0$;*
2. *$|vxy| \leq p$; and*
3. *for all $i \geq 0$, $uv^i xy^i z \in L$.*

Proof. Let $G = (V, \Sigma, R, S)$ be a context-free grammar such that $L(G) = L$, and let $b \geq 2$ denote the *branching factor* of G; that is, the maximum number of terminal and nonterminal symbols occurring on the right-hand side of any rule of G. If the height of some parse tree of G is h, then the length of any word derived using that parse tree will be at most b^h.

Let $n = |V|$ denote the number of nonterminal symbols of G, and take $p = b^{n+1}$. By our earlier observation, any word generated by G whose parse tree contains no path with a repeated nonterminal must have length at most b^n. Moreover, since $b \geq 2$, we must have that $b^{n+1} > b^n$.

Let w be any word in $L(G)$ where $|w| \geq p$, and let T be a parse tree for w of minimal size. We know, again by our earlier observation, that T must have a height of at least $n + 1$. Choose some path in T with length at least $n+1$, and take R to be the deepest nonterminal in the parse tree that occurs more than once in this path.

Decompose the word w into five parts, $uvxyz$, such that the subtree rooted at the upper occurrence of R has height at most $n + 1$ and the parts u and z are outside of the subtree rooted at the upper occurrence of R. We must have that $|vy| > 0$, since otherwise there would exist a smaller parse tree for w, which contradicts our assumption that T was of minimal size. Furthermore, the yield of this subtree, vxy, is a subword with length at most $p = b^{n+1}$. Finally, the word uxz is in L since we could replace the subtree rooted at the upper occurrence of R with the subtree rooted at the lower occurrence of R that yields only the part x, and for $i \geq 1$, all words of the form $uv^i xy^i z$ are in L since we can place copies of the subtree rooted at the upper occurrence of R at each subsequent occurrence of R. Therefore, all three conditions of the pumping lemma are satisfied. □

5.2 Using the Pumping Lemma

Just like before, we can write a proof that some language is non-context-free by simply following a common set of steps. As a consequence, all non-context-freeness proofs share a similar structure. To see such an example of a proof, let's revisit the language we introduced at the beginning of this section.

Example 26. Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, and consider the language $L_{\mathrm{a=b=c}} = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$. We will use the pumping lemma to show that this language is non-context-free.

Assume by way of contradiction that the language is context-free, and let p denote the pumping constant given by the pumping lemma. We choose the word $w = \mathtt{a}^p\mathtt{b}^p\mathtt{c}^p$. Clearly, $w \in L_{\mathrm{a=b=c}}$ and $|w| \geq p$. Thus, there exists a decomposition $w = uvxyz$ satisfying the three conditions of the pumping lemma.

Observe that the first condition of the pumping lemma requires that *either* part v or part y is nonempty; potentially both could be nonempty. We consider two cases, depending on the contents of the parts v and y of the word w:

1. Both part v and part y contain some number of a single alphabet symbol; that is, v contains only `a`s, only `b`s, or only `c`s, and likewise for y. (Note that v and y do not need to contain the *same* alphabet symbol; for example, v could contain only `a`s and y could contain only `b`s.)

   In this case, pumping v and y once to obtain the word uv^2xy^2z results in the word containing unequal numbers of `a`s, `b`s, and `c`s. This violates the third condition of the pumping lemma.

2. Either part v or part y contains some number of multiple alphabet symbols; that is, either v or y contains both `a`s and `b`s, or both `b`s and `c`s.

In this case, pumping v and y once to obtain the word uv^2xy^2z results in the word containing symbols out of order. This violates the third condition of the pumping lemma.

In all cases, one of the conditions of the pumping lemma is violated. As a consequence, the language cannot be context-free.

By a similar line of reasoning, we can show that the language $L = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m \mid m, n \geq 1\}$ is non-context-free, since there's no way for us to separate the counts of as/cs and bs/ds using only one stack.

Recall that, before, we used the pumping lemma for regular languages to show that the language $L_{\text{pal}} = \{ww^{\text{R}} \mid w \in \Sigma^*\}$ was nonregular. We can easily show that the language of palindromes is context-free. However, suppose we modify the language of palindromes so that the reversed occurrence of the word w is instead just a repetition of w. This "doubled" language has an almost identical structure to the language of palindromes, but it happens to be non-context-free!

Example 27. Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, and consider the language $L_{\text{double}} = \{ww \mid w \in \Sigma^*\}$. We will use the pumping lemma to show that this language is non-context-free.

Assume by way of contradiction that the language is context-free, and let p denote the pumping constant given by the pumping lemma. We choose the word $s = \mathtt{a}^p\mathtt{b}^p\mathtt{a}^p\mathtt{b}^p$. Clearly, $s \in L_{\text{double}}$ and $|s| \geq p$. Thus, there exists a decomposition $s = uvxyz$ satisfying the three conditions of the pumping lemma.

Observe that the second condition of the pumping lemma requires that $|vxy| \leq p$. Here, we will consider two cases, depending on the contents of the middle portion vxy of the word s:

1. If vxy occurs entirely within the first half of s (that is, within the first occurrence of $\mathtt{a}^p\mathtt{b}^p$), then as a consequence of the fact that $|vxy| \leq p$, we must have one of the following subcases: vxy contains all as, vxy contains all bs, or vxy contains both as and bs, where all as occur before bs.

   In any of these three subcases, pumping v and y once to obtain the word uv^2xy^2z results in the first half of the word differing from the second half of the word. We can make an analogous argument if vxy occurs entirely in the second half of s. This violates the third condition of the pumping lemma.

2. If vxy straddles both halves of s, then v and y must contain different symbols as a consequence of the fact that $|vxy| \leq p$. Pumping v and y down to obtain the word $uv^0xy^0z = uxz$ results in the first half containing fewer bs than the second half, and the second half containing fewer as than the first half. This violates the third condition of the pumping lemma.

In all cases, one of the conditions of the pumping lemma is violated. As a consequence, the language cannot be context-free.

Summary

Let's revisit our diagram by adding to it the class of context-free languages. Of course, knowing now that there exist languages that aren't context-free, our diagram must not be complete just yet. We still have some language classes to add, and we will be taking care of those classes in the lectures to come.