Lecture 11

Proving that languages are not context-free

In this lecture, we will show how to prove that languages are not context-free. We will essentially have only one tool for this: the "pumping lemma for context-free languages". This is a lemma that is similar in structure to the pumping lemma for regular languages, but which has a slightly different statement; it gives a property that is satisfied by all context-free languages, and thus, to show that a language is not context-free, it suffices to show that it does not satisfy the property described in the the lemma.

Of course, once we've managed to show that some specific languages are not context-free, we can always show that other languages are not context-free using closure properties. So in a sense, there will be two ways of showing that languages are not context-free: the pumping lemma for context-free languages, and closure properties.

11.1 The pumping lemma for context-free languages

Let us state the pumping lemma for context-free languages. Note that this is a completely separate result from the pumping lemma we've seen for regular languages, and it will require a different proof. However, they are both called "pumping lemmas" because the statements are somewhat analogous.

Theorem 11.1 (Pumping lemma for CFLs). *Let A be a context-free language over an alphabet Σ. Then there is a natural number $n \in \mathbb{N}$ such that for all strings $w \in A$ with $|w| \geq n$, there exist strings $x, u, y, v, z \in \Sigma^*$ such that the following conditions hold.*

1. $w = xuyvz$
2. $uv \neq \epsilon$
3. $|uyv| \leq n$
4. *For all $i \in \mathbb{N}$, we have $xu^iyv^iz \in A$.*

As in the case of regular languages, we will call the number n the *pumping length* of the context-free language A. (We will often use the term "pumping length" to refer to the minimum value of $n \in \mathbb{N}$ for which the lemma is satisfied for the context-free language A.)

Let's compare the pumping lemmas for the regular and context-free languages. In both cases, we can think of it as a game we play against the lemma. First, we choose a language A. The pumping lemma then responds by giving a natural number $n \in \mathbb{N}$. Next, we choose a string $w \in A$. The pumping lemma responds by giving a decomposition of w into substrings. Finally, we check whether the conditions of the lemma are satisfied by these substrings. The lemma says that these conditions can always be satisfied if A is regular or context-free (depending on whether we are talking about the pumping lemma for regular languages or for context-free languages). Therefore, if we can show that the conditions are not satisfied for our choice of w, then we've shown that the language A is not regular or not context-free.

Recall that for the regular languages, we had a decomposition $w = xyz$, and the condition was that we could "pump" the string y so that xy^iz is in the language for all $i \in \mathbb{N}$. For context-free languages, the situation is slightly more complex: instead of a decomposition into three parts $w = xyz$, we have a decomposition into five parts $w = xuyvz$. Also, instead of pumping one substring, we pump two of them at the same time: the guarantee is that $xu^iyv^iz \in A$ for all $i \in \mathbb{N}$. If the substring u and the substring v of w gets repeated the same number of times i, then the lemma guarantees that the resulting string is still in the context-free language A.

Both pumping lemmas would be pointless if the string to be pumped could be empty. That's why, for the regular-language pumping lemma, we have the condition $y \neq \epsilon$. For the context-free language pumping lemma, we have the condition $uv \neq \epsilon$, which is equivalent to saying that at least one of u and v is not the empty string.

Finally, Both pumping lemmas tell us something about where the substrings that can be pumped can be found. The regular-language pumping lemma says $|xy| \leq n$, which means that the string y which can be pumped is always found within the first n characters of the string w (here n is the pumping length for the regular language). The context-free-language pumping lemma says $|uyv| \leq n$, which says that the two strings u and v that can be pumped together are always within n characters of each other in w (where n is the pumping length for the context-free language).

11.2 Proving the pumping lemma for CFLs

Recall that to prove the pumping lemma for regular languages, we used the fact that each regular language is recognized by a DFA; since a DFA must have only a finite number n of states, if we run this DFA on a string w of length n or longer, there will be some state which it enters twice. This "loop" in the run of the DFA on the string w can be repeated any number of times to get other accepting paths in the DFA, and hence other strings in the language. This is how we showed that some substring y of w can be "pumped", proving the lemma.

How do we make such an argument for context-free languages? CFLs do not, in general, have DFAs that recognize them. Instead, they have CFGs that generate them. In fact, we've shown in a previous lecture that each context-free language is generated by some CFG in Chomsky normal form; this is the property we will use.

To be explicit, let A be any context-free language over an alphabet Σ, and let G be a CFG for A in Chomsky normal form. Let k be the number of variables of G, and set $n = 2^k$. We will show that this n works in the lemma (which means that the minimum pumping length of A is at most 2^k).

Let w be any string in A of length at least n. Since w is in A, there is some parse tree for w with respect to the CFG G. Also, since $|w| \geq n = 2^k$, we have $w \neq \epsilon$, so this parse tree has the form of a binary tree of variables, with the leaves being labeled by alphabet symbols. If we remove the leaves, we get a binary tree: each node is labeled by a variable, and has either 2 children or 0. The new leaves will now be labeled with variables (since we deleted the old leaves the were labeled with alphabet symbols). Moreover, there is one leaf in this binary tree for each symbol of w, and hence there are $|w|$ leaves in total.

Since the tree is a binary tree and it has $|w|$ leaves, it must have depth at least $\log_2|w|$. In other words, there is some path from the root to a leaf in this tree that is of length $\log_2|w|$ or longer, meaning there are at least $1+\log_2|w|$ nodes on this path (including both the root and the leaf). Since $|w| \geq 2^k$, we have $\log_2|w| \geq k$, so the number of nodes on this path is at least $k+1$. In particular, since there are only k variables and each node in this tree is labeled by a variable, some variable must be used twice in this path. More specifically, some variable must be used at least twice within the last $k+1$ nodes of this path. Let this variable be Y. In Chomsky normal form, the start variable S cannot occur on the right hand side of any rule, and hence if a variable Y occurs twice in a path of the parse tree of w, we must have $Y \neq S$.

We've identified two occurrences of Y in the parse tree of w, with one of them being a descendant of the other. Both occurrences are within the last $k+1$ nodes of the longest path in the parse tree of w. This means that if you look at the sub-tree of the topmost of the two occurrences, this sub-tree can have depth at most k (that is, at most $k+1$ nodes can occur on any path from the root of the sub-tree, which is labeled Y, to a leaf of the sub-tree). This means that the sub-tree has at most 2^k leaves. Let s be the string that the first occurrence of Y turns into in the parse tree of w; then $|s| \leq 2^k$, since each leaf turns into exactly one alphabet symbol. Also, we have $Y \Rightarrow_G^* s$, and $w = xsz$ for some strings $x, z \in \Sigma^*$.

Now consider the second of the two occurrences of Y we've identified; this is a descendent of the first. Let y be the string that this second occurrence of Y turns into in the parse tree of w; then $s = uyv$ for some strings $u, v \in \Sigma^*$. Also, we can write

$$Y \Rightarrow_G^* uYv \Rightarrow_G^* uyv$$

to describe the derivation of s from the first occurrence of Y. Note that since none of the variables can turn into ϵ (since G is in Chomsky normal form), it's not possible for both u and v to be ϵ, since that would force the two occurrences of Y to be the same node. Hence we have $|uv| > 0$, as well as $|uyv| = |s| \leq 2^k = n$. Since $w = xsz$, we also have $w = xuyvz$.

Finally, note that we have

$$S \Rightarrow_G^* xYz, \qquad Y \Rightarrow_G^* uYv, \qquad Y \Rightarrow_G^* y.$$

We can generate strings using the grammar G by starting with $S \Rightarrow_G^* xYz$, applying the path $Y \Rightarrow_G^* uYv$ some number of times, say i times, and finishing with $Y \Rightarrow_G^* y$; this gives a derivation for the string xu^iyv^iz in the grammar G, for every $i \in \mathbb{N}$ (including $i = 0$). This completes the proof of the pumping lemma for context-free languages.

11.3 Using the pumping lemma

Let's now use the pumping lemma for CFLs to show certain languages are not context-free.

Problem 11.2. *Let $A = \{0^n1^n2^n : n \in \mathbb{N}\}$. Show that A is not context-free.*

To show this, we use the pumping lemma. Suppose by contradiction that A was context-free, and let n be its pumping length. Pick $w = 0^n1^n2^n$; then $|w| \geq n$. The pumping lemma then gives us a decomposition $w = xuyvz$ such that $|uyv| \leq n$, $|uv| > 0$, and such that $xu^iyv^iz \in A$ for all $i \in \mathbb{N}$. Since $|uyv| \leq n$, this string cannot contain both a 0 and a 2, since the 0s and the 2s have distance n inside w. If uyv does not contain a 0, then xu^2yv^2z is a string that's longer than $3n$ but which has only n 0s, which means it cannot be of the form $0^m1^m2^m$ for any $m \in \mathbb{N}$; similarly, if uyv

does not contain a 2, then xu^2yv^2z is a string that's longer than $3n$ but which has only n 2s, which means it cannot be of the form $0^m1^m2^m$ for any $m \in \mathbb{N}$. This means that $xu^2yv^2z \notin A$, giving a contradiction. Thus A is not context-free.

Problem 11.3. *Let $B = \{0^n1^m0^n1^m : n, m \in \mathbb{N}\}$. Show that B is not context-free.*

To show this, suppose by contradiction that B was context-free, and let n be its pumping length (assume $n \geq 1$). Pick $w = 0^n1^n0^n1^n$. The pumping lemma gives us a decomposition $w = xuyvz$ with $|uyv| \leq n$; this means that u and v span at most two blocks of 0s or 1s in w. If either u or v straddles two different blocks, then xu^2yv^2z would have at least 6 blocks of alternating 0s and 1s, which would make it not in B. If u and v are in the same block, then xu^2yv^2z will three different blocks having length n and the fourth having length strictly greater than n (since $|uv| > 0$), which means that it cannot have the form $0^n1^m0^n1^m$ for any $n, m \in \mathbb{N}$, and hence is not in B. Finally, if u lies in one block and v lies in the subsequent block, then since either u or v is nonempty, either the two blocks of 0s in xu^2yv^2z have different lengths or else the two blocks of 1s in xu^2yv^2z have different lengths. In all cases, xu^2yv^2z cannot be in B, which contradicts the pumping lemma. Hence B is not context-free.

Problem 11.4. *Let $C = \{0^p : p \in \mathbb{N},\ p \text{ is prime}\}$. Show that C is not context-free.*

Once again we will use the pumping lemma. Suppose by contradiction that C was context-free, and let n be its pumping length. Pick a prime number $p \geq n$ (which is possible as there are infinitely many prime numbers), and let $w = 0^p \in C$. The pumping lemma gives us a decomposition $w = xuyvz$ with $|uyv| \leq n$ and $|uv| > 0$ such that $xu^iyv^iz \in C$ for all $i \in \mathbb{N}$. Let $k = |uv| \geq 1$. Then for each $i \in \mathbb{N}$, we have $xu^iyv^iz = 0^{p+(i-1)k}$, and since these strings are all in C, it must be the case that $p + (i-1)k$ is prime for all $i \in \mathbb{N}$. Pick $i = p + 1$. Then the number $p + (i-1)k$ is equal to $p + pk = p(k+1)$. Since $k \geq 1$, we have $k + 1 \geq 2$, and since p is prime we have $p \geq 2$. Therefore, the product p times $k+1$ is a factorization of the number $p + (i-1)k$ when $i = p+1$, so this number is not prime. This is a contradiction, so C is not context-free.

Problem 11.5. *Let $D = \{r\#s : r, s \in \{0,1\}^*, r \text{ is a substring of } s\}$. Show that D is not context-free.*

Suppose by contradiction that D was context-free, and let n be its pumping length. Pick $w = 0^n1^n\#0^n1^n \in D$. The pumping lemma gives us a decomposition $w = xuyvz$ such that $|uyv| \leq n$, $|uv| > 0$, and $xu^iyv^iz \in D$ for all $i \in \mathbb{N}$. Since $|uyv| \leq n$, the strings u and v can contain symbols from at most two consecutive blocks of 0s or 1s. Also, since $xu^2yv^2z \in D$ but since the strings in D contain exactly one copy of #, it must be the case that # does not lie in u or in v, so it is in one of x, y, or z. if # is in x, then u and v both contain parts of w that are after the # symbol, so the string xu^0yv^0z contains fewer symbols after the # symbol than before it; this means xu^0yv^0z cannot be in D. If # is in z, the string xu^2yv^2z contains more than $2n$ symbols before the # symbols and $2n$ symbols after the #, so it cannot be in D.

Finally, suppose that # is in y. In this case, u is a substring of the first block of 1s, and v is a substring of the second block of 0s. We split into two cases. If $u \neq \epsilon$, then xu^2yv^2z has its first block of 1s be of length strictly larger than n, but its second block of 1s is of length n; this means the string to the left of # cannot be a substring of the string to the right of #, so the whole string xu^2yv^2z cannot be in D. Finally, suppose that $u = \epsilon$. Then $v \neq \epsilon$, since $|uv| > 0$. In this case, the string xu^0yv^0z contains more symbols to the left of # than to the right of #, so it also cannot be in D. In all cases, some string of the form xu^iyv^iz is not in D, which contradicts the pumping lemma; hence D is not context-free.

11.4 Showing languages are not CFLs using closure properties

Just as we saw in the case of regular languages, we can also use closure properties to show that languages are not context-free. This essentially works by giving a *reduction*: we reduce the problem of showing some language A is not context-free to showing that some other language B is not context-free, by showing that if A were context-free then B would also be context-free. We then show that the language B is not context-free using the pumping lemma for context-free languages. The hope is that it is easier to show that B is not context-free using the pumping lemma than to show that A is not context-free using the pumping lemma.

The context-free languages do not have as many closure properties as the regular languages do. For instance, they are not closed under intersection, and intersection was a very useful closure property to use when showing that languages are not regular. Luckily, context-free languages are closed under intersection with a regular language; this will be a very convenient closure property to use.

Problem 11.6. *Let $A = \{w \in \{0,1,2\}^* : |w|_0 = |w|_1 = |w|_2\}$. Show that A is not context-free. (Recall that $|w|_0$ denotes the number of 0-symbols in w, and similarly for $|w|_1$ and $|w|_2$.)*

Consider the language $L = \{0^i 1^j 2^k : i, j, k \in \mathbb{N}\}$. This language has the regular expression $0^*1^*2^*$, so it is regular. Suppose by contradiction that A was context-free; then $A \cap L$ would also be context-free, since the intersection of a context-free language and a regular language is always context-free. Note that $A \cap L = \{0^n 1^n 2^n : n \in \mathbb{N}\}$. Since this language is context-free (under our assumption that A is context-free), let its CFG be G. Let G' be the CFG in which we replace the symbol 2 with the symbol 0 in all rules. Then G' generates the language $\{0^n 1^n 0^n : n \in \mathbb{N}\}$, so the latter is context-free. However, we already saw in Problem 11.2 that this language is not context-free; this gives a contradiction, so A is not context-free.

Note that we could have also shown that the language $\{0^n 1^n 2^n : n \in \mathbb{N}\}$ is not context-free by directly using the pumping lemma. In the future, we will assume that this language is not context-free without proof (it tends to come up repeatedly).

Problem 11.7. *Let $B = \{0^n 1^n 2^n : n \in \mathbb{N}\}$, and let $C = B^*$. Show that C is not context-free.*

As usual, we assume that C is context-free and try to get a contradiction. It might be tempting to take the intersection of C with B, but we have no control over whether the resulting language is context-free or not because B is not regular; for this reason, it will be hard to get a contradiction by examining $B \cap C$. Instead, we look at $C \cap L(0^*1^*2^*)$. The language $L(0^*1^*2^*) = \{0^i 1^j 2^k : i, j, k \in \mathbb{N}\}$ is regular, and since C is context-free by our assumption, we get that $C \cap L(0^*1^*2^*)$ is context-free.

Now it just remains to show that $C \cap L(0^*1^*2^*) = B$. To see this, note that any string in B is clearly in both C and $L(0^*1^*2^*)$. On the other hand, any string that is in both C and $L(0^*1^*2^*)$ must be in B^i for some i, and hence must have the form $0^{n_1}1^{n_1}2^{n_1}0^{n_2}1^{n_2}2^{n_2}\dots 0^{n_i}1^{n_i}2^{n_i}$; since the string is in $L(0^*1^*2^*)$, all the n_j must be 0 except for one, so the string must have the form $0^n 1^n 2^n$ for some $n \in \mathbb{N}$, and hence each string in $C \cap L(0^*1^*2^*)$ must be in B.

Finally, since we know that B is not context-free, we get a contradiction. This shows that C is not context-free.

11.4.1 Other useful closure properties

We mention two other useful closure properties for context-free languages, which can sometimes be used to show that a language is not context-free.

First, context-free languages are closed under symmetric difference with a finite set. That is, if A is context-free and B is finite, then $A \triangle B$ is context-free. To see this, note that $A \triangle B = (A \setminus B) \cup (B \setminus A)$. We have $A \setminus B = A \cap \overline{B}$; since B is finite, B is regular, so $\overline{B}$ is also regular, and since the intersection of a regular and a context-free language is context-free, we conclude $A \setminus B$ is context-free. Also, since B is finite, $B \setminus A$ is also finite, and finite languages are always regular (and hence always context-free). Finally, since context-free languages are closed under union, we conclude that $A \triangle B$ is context-free. This property is often useful.

Problem 11.8. *Let $A = \{0^n 1^n 2^n : n \in \mathbb{N}, n \geq 2\}$. Show that A is not context-free.*

Solution: if A were context-free, then $A \triangle \{\epsilon, 012\}$ would be context-free. But the latter is $\{0^n 1^n 2^n : n \in \mathbb{N}\}$, which we've shown is not context-free.

Another useful property is the closure of context-free languages under alphabet projections. That is, let A be a context-free language over the alphabet Σ, and let ϕ be a function from Σ to another (possibly smaller) alphabet Σ'. For each string $x \in \Sigma^*$, define x_ϕ to be $\phi(x_1)\phi(x_2)\ldots\phi(x_n)$, where $x_1, x_2, \ldots, x_n$ are the symbols of x. Define $A_\phi = \{x_\phi : x \in A\}$; this is a language over the alphabet Σ'. We claim that A_ϕ is also context-free (assuming A is context-free). The reason is that we can take any CFG G for A, and replace each alphabet symbol $c \in \Sigma$ that occurs in the rules of G with the symbol $\phi(c)$. This new CFG will then generate A_ϕ. This closure property can also be used to show that a language is not context-free.

Problem 11.9. *Let $B = \{w \in \{0,1\}^* : |w| \text{ is prime}\}$. Show that B is not context-free.*

Solution: consider the alphabet projection from $\{0,1\}$ to $\{0\}$ that maps $0 \to 0$ and $1 \to 0$. The projection of B under this alphabet change is the language $\{0^p : p \in \mathbb{N}, p \text{ is prime}\}$. If B were context-free, then so would this projected language, but we've already shown it is not context-free.

11.5 Further remarks

As we've mentioned, the context-free languages are not closed under intersection and complement. We now have the tools to prove this: let $A = \{0^n 1^n 0^m : n, m \in \mathbb{N}\}$. Then A is context-free: it is generated by the CFG

$$S \to XY;\ X \to 0X1|\epsilon;\ Y \to 0Y|\epsilon.$$

(Another way to see that A is context-free is to note that it is the concatenation of $\{0^n 1^n : n \in \mathbb{N}\}$, which we've already previously seen is context-free, and the regular language $L(0^*)$; since regular languages are context-free and since context-free languages are closed under concatenation, A is context-free.)

Since context-free languages are closed under reverse, $A^R = \{0^m 1^n 0^n : n, m \in \mathbb{N}\}$ is also context-free. However, the language $A \cap A^R$ is equal to $\{0^n 1^n 0^n : n \in \mathbb{N}\}$, which we've seen is not context-free. This means that the context-free languages are not closed under intersection: sometimes, the intersection of two context-free languages is not context-free.

What about complement? Well, if the context-free languages were closed under complement, they would also be closed under intersection, which we know they are not. This is because $A \cap B = \overline{\overline{A} \cup \overline{B}}$. In other words, if A and B are context-free and if complement preserved context-free-ness, then $\overline{A}$ and $\overline{B}$ would be context-free, and, since the context-free langauges are closed under union, $\overline{A} \cup \overline{B}$ would be context-free; but this would also imply that $\overline{\overline{A} \cup \overline{B}} = A \cap B$ would be context-free. We know that the latter fails for some choice of A and B, which means that sometimes, the complement of a context-free language is not context-free.