Lecture 10

Ben Rosenberg

June 22, 2021

Recall: to prove that a language L is not regular:

1. Recognize a property that must be satisfied by **all** elements of L
2. Assume that L is regular, name the positive constant of the Pumping Lemma
3. Select an element $w \in L$ that is long enough to pump
4. For every admissible pumping decomposition $w = xyz$

- Select i such that xy^iz violates the recognized property, which means that $xy^iz \in L$

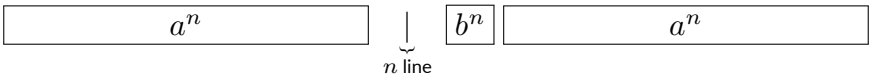
Consider $\Sigma = \{a, b\}$ and $p \equiv$ the set of palindromes over Σ. Then, the property is that the strings are palindromic. Let $n > 0$ be the constant of the P.L.

How to fail: select a long palindrome. The thing is, a^{n+1} is a palindrome, and remains so when it is pumped even though it is long enough to pump because $|a^{n+1}| > n$.

Back to the solution: recall that we want some string for which we can do this:

Figure 1: Pumping

Select a string: $w = a^nba^n$. We have $|w| = 2n + 1 > n$ and so w must pump.



Where is the pumping window? To the left of the n-line.

We say that the P.W. is a^j for some $j > 0$.

Pumping up once gives w_1 with $n + j$ a's; $w_1 = a^{n+j}na^n$.

Since the $(n + 1)$st symbol from the right is b and the $(n + 1)$st symbol from the left is a, w_1 cannot be a palindrome.

(Note that the pumping window cannot be of zero length by definition.)

Another example: Suppose that we have the set of strings $L_S = \{a^n | n = k^2, k \in \mathbb{N}\}$.

So, $L_S = \{\lambda, a, aaaa, a^9, a^{16}, a^{25}, \dots\}$.

We want to prove that L_S is not regular. The property that we will use is that the number of a's is a square of a natural number.

Assume for the sake of contradiction that L_S is regular. Let $\alpha > 0$ be the constant of the P.L. Select a string $w = a^{n^2}$ where $n > \alpha$.

Wherever the pumping window is, it is equal to a^j for some j such that $1 \leq j \leq \alpha$ (in accordance with the constraints provided by the pumping lemma).

Puump w up once, and obtain $w_1 = a^{n^2+j}$. We need to prove that $n^2 + j$ is not a square of a natural number.

We know that $n^2 < n^2 + j < n^2 + \alpha$ by the constraints on j. This in turn is strictly less than $n^2 < n$ because of our choice of n as greater than α. Then, this is strictly less than $n^2 + 2n + 1$ by arithmetic. Since this is $(n+1)^2$, we have $n^2 < |w_1| < (n+1)^2$. And so, if $|w_1$ is a square, then it cannot be the square of some natural number.

The length of a string in a regular language has the form $\alpha n = \beta$, for $n \geq 0$ and constants α and β. This comes from the Kleene star.

The difference between a regular language and, say, the language defined above as L_S, is in the difference between any two consecutive possible lengths. Between n^2 and $(n+1)^2$ in L_S, we have a difference of $2n + 1$ which grows unboundedly. Any such function besides linear functions will grow unboundedly as the difference between points becomes unbounded. We should know that a language is not regular when its size does not honor this pattern.

Pumping lemma for context-free languages

Regular languages pump because the automata that represent it are finite, and have only so many states. Eventually, a state will be repeated.

Similarly, a context-free language will pump because its grammar has only so many variables; waiting long enough will lead to a variable being repeated in the derivation.

Theorem:

Let L be a context-free language. Then:

$(\exists k > 0)(\forall w \in L)(|w| > k \implies ((\exists x, y_1, t, y_2, z \in \Sigma^*)(w = xy_1ty_2z \wedge |y_1ty_2 \leq k \wedge |y_1y_2| > 0 \wedge ((\forall i \geq 0)(xy_1^ity_2^iz \in L)))))$

In effect, the pumping window is split in general, and the two pieces pump in sync. (This comes from telescoping.)

Example: $L_3\{a^nb^nc^n | n \geq 0\}$. Intuitively, this cannot be telescoping because there are three variables and not two – the "parentheses are unmatched."

Proof that L_3 is not context-free: Assume that it is. The property is that the number of a's is equal to the number of b's and the number of c's.

Let $k > 0$ be the constant of the P.L. Select $w = a^nb^nc^n$ for some $n > k$. Where is the pumping window?

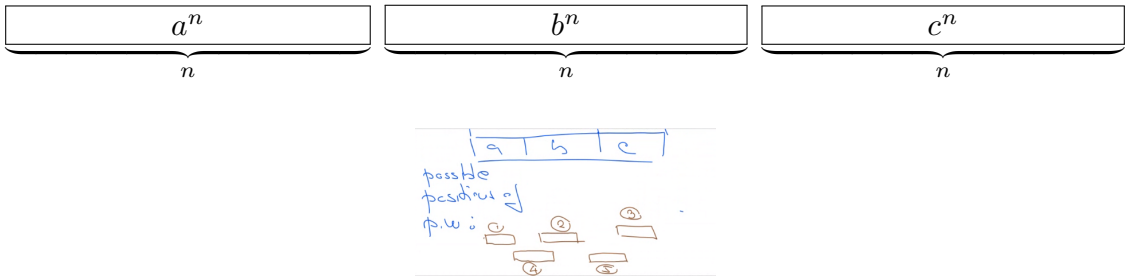


Figure 2: Enumeration of possible positions of the pumping window

By our own choice, we cannot pump up a's, b's, and c's at the same time because the pumping window cannot exceed a length of k and $n > k$.

Pumping up once gives a string not in L_3 because there are either too many a's, or b's, or c's, or too few c's, or too few a's.

We have enumerated all 5 cases, each of which results in a language that is not context-free by our definition of the PL for context-free languages.

Recall: The class of regular languages is closed under the operations of union, concatenation, the Kleene star, intersection, and complement.

The class of context-free languages is closed under regular operators, and we had an algorithm that we applied on grammars.

Recall: $L_3 = \{a^n b^n c^n\}$ is not context-free (we just proved it using the pumping lemma).

But, for example, $L_1 = \{a^n b^n c^k | n, k \geq 0\}$ is context-free. So is $L_2 = \{a^n b^k c^k | n, k \geq 0\}$.

It is the case, however, that $L_3 = L_1 \cap L_2$.

Theorem: The class of context-free languages is *not* closed under intersection.

What about the complement operation?

Theorem: The class of context-free languages is *not* closed under complement.

Proof: Find a context-free language whose complement is not context-free. As it happens, $\overline{L_3}$ is context-free, and L_3 is not context-free.

Another proof: Reduction. Recall: set operations.

$$L_1 \cap L_2 \xrightarrow{\text{DeMorgan's Laws}} \overline{\overline{L_1} \cup \overline{L_2}}$$

Let L_1 and L_2 be any pair of context-free languages. If the complement of a context-free language was always context-free, then $\overline{L_1}$, $\overline{L_2}$, $\overline{L_1} \cup \overline{L_2}$, and $\overline{\overline{L_1} \cup \overline{L_2}}$ would be context-free. (Context-free languages are closed under union.) But then, the intersection $L_1 \cap L_2$ would also always be a context-free language, but sometimes, it is not as we know from the previous proof.

This constitutes a "reduction" from the proof that context-free languages aren't closed under intersection to a proof that context-free languages aren't closed under complement. (Aside: this is not a real reduction. Real reductions are significantly more complicated and are used to solve algorithmic problems, not prove corollaries.)

"How to fail"

$R = \{a^n b^k c^j | n, j, k \geq 0\}$ is regular.

But $R_1 = \{a^n b^k c^j | n, j, k \geq 0\}$ is not, as there is telescoping. We can use the pumping lemma with the property that there is an equal number of b's and c's. Call the P.L. constant α. and select the string $b^m c^m, m > \alpha$. We've already done this – make too many b's after pumping and then the language no longer satisfies the property.

What about $R_2 = \{a^{n+2} n^k c^k | n, k \geq 0\}$? This is not regular either. Using the P.L., take the same property of # b's = # c's, constant α, and select string $w = aab^m c^m, m > \alpha$. This string is obtained by setting $n = 0$. Now, we want to pump w. Where is the pumping window? It can be anywhere from aa to anywhere else. The issue is if we are made to pump a and a alone, as pumping this makes good strings.

And so, we need to find a good proof.

Let languages $A_1 = aab^*c^*$, and let $A_2 = \{aab^kc^k | k \geq 0\}$. Then, $A_2 = A_1 \cap R_3$.

Proof by P.L. that A_2 is not regular: straightforward property #b's = #c's

String: $aab^mc^m, m > \alpha$

Pumping must be in a's because A_2 has exactly two a's. And so, A_2 is not regular and therefore, R_3 must not be regular because the class of regular languages is closed under intersection and we know A_1 to be regular.

This is another "reduction."

Theorem: The intersection of a context-free language with a regular language is context-free. This should be learned, albeit without proof.

Turing machines

Turing machines look somewhat like finite automata, with a state box and a tape with a beginning. The difference is that the tape continues forever, and the head of the machine is a read/write/left/right head, while the head of a finite automaton was a read/right -only head. There are also symbols called "blanks", which take the place of a symbol and are present in the "tail end" of the string out to $\aleph_0$.

We will continue with the actual definition of Turing machines tomorrow.