University of Waterloo

CS 360: Introduction to the Theory of Computing

Fall 1998

Nine Errors Students Commonly Make When Applying the Pumping Lemma

The pumping lemma for regular languages is the following:

Lemma.
For all regular languages L, there exists a constant n (depending on L) such that for all $z \in L$, $|z| \geq n$, there exists a decomposition $z = uvw$, with $|uv| \leq n$, $|v| \geq 1$, such that for all $i \geq 0$, $uv^iw \in L$.

Note that the pumping lemma states a property of regular languages. Hence one cannot use it to prove that a language is regular, but one *can* use the contrapositive (or proof by contradiction) to prove that a language is *non-regular*. The contrapositive is:

If for all n, there exists a $z \in L$ with $|z| \geq n$ such that for all decompositions $z = uvw$ satisfying the conditions $|uv| \leq n$ and $|v| \geq 1$, there exists an $i \geq 0$ such that $uv^iw \notin L$, then L is non-regular.

The common way to employ the pumping lemma is as follows: you pretend that an "adversary" has chosen n. You must be prepared in what follows to handle *any* n. You then choose your string $z \in L$ with $|z| \geq n$. Note that your string should depend on n in some way. Now the adversary gets to pick any decomposition whatsoever $z = uvw$, subject to the conditions that $|uv| \leq n$, and $|v| \geq 1$. Now you get to pick the appropriate i such that $uv^iw \notin L$, thereby showing that L is not regular.

The following are the nine errors students commonly make in applying the pumping lemma:

Error 1. Choosing a string z that is not in L. For example, suppose

$$L = \{ww \ : \ w \in (\mathtt{a} + \mathtt{b})^*\}.$$

You might incorrectly choose $z = \mathtt{a}^n\mathtt{b}^n$, which is not in L. At this point it's easy to get a "contradiction": just pick $i = 1$; then $z = uv^iw \notin L$.

Error 2. Not handling all possible decompositions of the string z as uvw. For example, consider

$$L = \{ww \ : \ w \in (\mathtt{a} + \mathtt{b})^*\}$$

again. Suppose the adversary chooses n and you choose $z = \mathtt{a}^{2n}$. Then the adversary is supposed to choose a decomposition $z = uvw$. If, by mistake, you do not examine *all* *possible* decompositions of z, you might wrongly choose to look only at the decomposition specified by $u = \Lambda, v = \mathtt{a}, w = \mathtt{a}^{2n-1}$. In this case, you could choose $i = 0$, to get the string $uv^iw = \mathtt{a}^{2n-1} \notin L$, to get a "contradiction". But it isn't *really* a contradiction, since you haven't examined all possible ways the adversary could decompose z. In particular, the adversary could choose $u = \Lambda, v = \mathtt{aa}, w = \mathtt{a}^{2n-2}$, in which case $uv^iw \in L$ for all $i \geq 0$.

Error 3. Choosing a string z that is not specific enough. Remember: you get to choose *any* string in L, based on n, that is longer than n in length. Why make the adversary's job easy? The adversary wants to defeat you by picking a bad decomposition. Usually, the more *specific* you choose your string, the harder time the adversary will have.

For example, in the language L above, you might have been tempted to choose $z = xx$, where x was *any* string of length $\geq n$. Then you let the adversary break the string up as $z = uvw = xx$. By picking $i = 0$, you might conclude that $uw \neq xx$, and so obtain a "contradiction". But this is simply not true! It does *not* suffice to show that $uv^iw \neq xx$ for a *particular* x; you must show it for *all possible* x, since that is the meaning of not being in L.

In fact, this kind of argument *cannot* succeed with such a general choice of z. For if your string was, say, $z = \mathtt{a}^n\mathtt{a}^n$, then the adversary can choose $u = \Lambda$, $v = \mathtt{aa}$, and $w = \mathtt{a}^{2n-2}$. In this case, no matter what i you choose, the resulting string $uv^iw \in L$, and you cannot "win".

Moral of the story: construct your string z with care.

Error 4. Choosing a string z that does not depend on n. For example, in the language L above, suppose you picked $z = \mathtt{abab}$. The problem is that you don't know what n is; you must be able to account to for *all possible* values of n picked by the adversary. If the length of the string you picked is *not* a function of n, you are in trouble.

Error 5. Choosing a negative or fractional i. This is not allowed by the statement of the pumping lemma. In looking at uv^iw, you must choose an i that is an integer ≥ 0.

Error 6. Applying the pumping lemma to a regular language. For example, consider

$$L = \{\mathtt{0}^x\mathtt{1}^y \ : \ x + y \equiv 0 \pmod 4\}.$$

This language is regular, but you might be tempted to try to prove it is *not* regular via the pumping lemma. You might pick, for example, the string $z = 0^{4n+3}1$. Then let the adversary decompose z as $z = uvw$. Hence $u = 0^a$, $v = 0^b$, and $w = 0^c1$, where $a + b + c = 4n + 3$. Then you might assert, "We can choose i such that $uv^iw = 0^{4n+3+ib}1$, and then clearly for all b we have that $x + y = 4n + 3 + ib + 1$ is not a multiple of 4."

The problem with this claim is that it is false. For example, if $b = 4$, then $4n+3+ib+1$ is a multiple of 4 for all i.

Moral here: be careful about what you assert, and be fairly confident that the language is indeed non-regular before you begin your proof.

Error 7. Assuming that all long strings in a regular language L can be written as uv^iw for some $i \geq 2$. This is not necessarily true. For example, if $L = (\mathtt{0} + \mathtt{1} + \mathtt{2})^*$, then you might be tempted to conclude that there exist words u, v, w such that all sufficiently long strings in L can be written as uv^iw for some $i \geq 2$. This is simply false, as there exist strings in L that contain no substring of the form vv — this was first proved by the Norwegian mathematician Axel Thue in the early 1900's.

Thue's example also kills the same "theorem" when u, v, and w are allowed to lie in some *finite* set.

Error 8. Trying to use the pumping lemma to prove that a language is regular. The pumping lemma is a statement about a property of regular languages. It says, "If L is regular, then L has the following property." Hence one cannot use the pumping lemma to prove that a language is regular; one can only use it to prove a language is *non*-regular.

In fact, there are languages which are non-regular, but nevertheless satisfy the conclusions of the pumping lemma! One example is the following language:

$$L = \{a^ib^jc^k \ : \ i = 0 \text{ or } j = k\}.$$

Suppose $z \in L$ is the string chosen to pump. There are two cases.

Case 1: $z = b^jc^k$ for some integers j, k. Pick $n = 1$; hence we may assume either $j \geq 1$ or $k \geq 1$. Then there exists a decomposition $z = uvw$, where $u = \Lambda$, $v = b$ (if $j \geq 1$) or $v = c$ (if $j = 0$) $w =$ the rest of the string, and then $uv^iw \in L$ for all $i \geq 0$.

Case 2: $z = a^ib^jc^j$, for some integers i, j with $i \geq 1$. Pick $n = 1$. Then there exists a decomposition $z = uvw$, where $u = \Lambda$, $v = a$, and $w =$ the rest of the string, and $uv^iw \in L$ for all $i \geq 0$.

The moral of the story is that the ordinary pumping lemma is not powerful enough to be able to directly prove the non-regularity of certain non-regular languages. Other techniques are needed.

Error 9. Choosing a string $z = z(n)$, depending on n, in such a way that

$$\{z(n) \ : \ n \geq 1\}$$

is a regular language. This is a rather subtle error, and understanding the error requires a fairly deep understanding of the pumping lemma itself, so you may wish to skip this one first time around.

If you choose the string $z = z(n)$ to depend on n in such a way that

$$L_z = \{z(n) \ : \ n \geq 1\}$$

is itself regular, then the pumping lemma cannot succeed in proving L non-regular. For suppose it did. Then for each way of decomposing $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$, there would be a choice of $i \geq 0$ such that $uv^iw \notin L$. But since $L_z \subseteq L$, $uv^iw \notin L_z$. Hence by the pumping lemma, L_z itself would not be regular. But L_z is; a contradiction.

Hence one must choose the string $z = z(n)$ in a sufficiently "irregular" way to ensure that L_z itself is not regular. As an example, consider the language

$$L = \{ww \ : \ w \in (\mathtt{a} + \mathtt{b})^*\}.$$

One might be tempted to choose the string $z = z(n) = \mathtt{a}^{2n}$, which is certainly in L. However, the associated language is

$$L_z = \{\mathtt{a}^{2n} \ : \ n \geq 1\} = (\mathtt{aa})^+,$$

which is regular, so this choice for z cannot possibly succeed in proving that L is non-regular.