Nine Common Errors with the Pumping Lemma

Jeffrey Shallit
School of Computer Science
University of Waterloo
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit
Recall the pumping lemma for regular languages:

Lemma.

If \( L \) is regular, there exists a constant \( n \) (depending on \( L \)) such that for all \( z \in L \) with \( |z| \geq n \), there exists a factorization \( z = uvw \), with \( |uv| \leq n \), \( |v| \geq 1 \), such that for all \( i \geq 0 \) we have \( uv^i w \in L \).
The contrapositive is as follows:

\textit{If for all } n \textit{, there exists a } z \in L \textit{ with } |z| \geq n \textit{ such that for all factorizations } z = uvw \textit{ satisfying the conditions } |uv| \leq n \textit{ and } |v| \geq 1 \textit{, there exists an } i \geq 0 \textit{ such that } uv^i w \notin L \textit{, then } L \textit{ is non-regular.}
The pumping game

Recall the “game” interpretation of the contrapositive.

You are playing a four-step game against an adversary. The adversary is all-powerful, knows everything, but cannot cheat. Your goal is prove the language non-regular; your adversary is trying to prevent your proof from going through. You take turns choosing various objects:

- **Step 1:** adversary chooses $n$.
- **Step 2:** you choose $z \in L$ with $|z| \geq n$.
- **Step 3:** adversary chooses a factorization $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$.
- **Step 4:** you choose $i$. You “win” and show $L$ is not regular if $uv^i w \notin L$, no matter what the adversary did in steps 1 and 3. Otherwise you lose: your proof didn’t work.
Error #1. Choosing a string $z$ that is not in $L$.

Suppose

$$L = \{ww : w \in \{a, b\}^*\}.$$

You might incorrectly choose $z = a^n b^n$, which is not in $L$.

At this point it’s easy to “win” — just pick $i = 1$; then $z = uv^i w \not\in L$.

But you didn’t win—because you cheated by picking $z \not\in L$. 
Error #2. Not handling all possible factorizations of the string $z$ as $uvw$.

Consider

$$L = \{ww : w \in \{a, b\}^*\}$$

again.

Suppose the adversary chooses $n$ and you choose $z = a^{2n}$.

Then the adversary is supposed to choose a factorization $z = uvw$.

If, by mistake, you do not think about all possible factorizations of $z$, you might wrongly choose to look only at the factorization specified by $u = \epsilon, v = a, w = a^{2n-1}$.
Error #2. Not handling all possible factorizations of the string $z$ as $uvw$.

In this case, you could choose $i = 0$, to get the string $uv^i w = a^{2n-1} \notin L$ and “win”.

But you haven’t really “won”, because you didn’t handle all possible ways the adversary could factor $z$.

The adversary could have chosen $u = \epsilon, v = aa, w = a^{2n-2}$, in which case $uv^i w \in L$ for all $i \geq 0$.

You failed to consider this choice.
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 factorization. Usually, the more specific you choose your string, the less freedom the adversary will have to respond.

For example, in the language $L = \{ww : w \in \{a, b\}^*\}$ 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”.
Error #3. Choosing a string $z$ that is not specific enough.

But this is simply not true! It does \textit{not} suffice to show that $uv^iw \neq xx$ for a \textit{particular} $x$; you must show it for \textit{all possible} $x$, since that is the meaning of not being in $L$.

In fact, this kind of argument \textit{cannot} succeed with such a general choice of $z$.

For if your string was, say, $z = a^n a^n$, then as we saw, the adversary could choose $u = \epsilon$, $v = aa$, and $w = a^{2n-2}$.

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, and as specific as possible.
Error #4. Choosing a string $z$ that does not depend on $n$.

For example, in the language

$$L = \{ww : w \in \{a, b\}^\ast\}$$

suppose you picked $z = abab$.

The problem is that in the pumping game 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 adversary had picked $n = 5$, then your string $z$ would not be longer than $n$.

So your string $z$ must depend on $n$ in some way.

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 choice—a negative or fractional $i$—is not allowed by the statement of the pumping lemma.

In looking at $uv^iw$, you must choose an $i$ that is a non-negative integer.

Also, there is never any need to choose $i = 1$, since $uv^1w = uvw = z \in L$.
Error #6. Applying the pumping lemma to a regular language.

For example, consider

\[ L = \{0^x1^y : x + y \equiv 0 \pmod{4}\}. \]

This language is regular, but you might not realize it.

Then you might try to prove it is not regular via the pumping lemma.

You might pick, for example, the string \( z = 0^{4n+3}1 \).

Then suppose the adversary factors \( z \) as \( z = uvw \).

Hence \( u = 0^a, \ v = 0^b, \) and \( w = 0^c1 \), where \( a + b + c = 4n + 3 \).
Error #6. Applying the pumping lemma to a regular language.

Then you might assert, “We can choose $i$ such that $uv^i w = 0^{4n+3+(i-1)b}1$, and then clearly for all $b$ we have that $x + y = 4n + 3 + (i - 1)b + 1$ is not a multiple of 4.”

The problem with this claim is that it is false. For example, if $b = 4$, then $4n + 3 + (i - 1)b + 1$ 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 $uv^i w$ for some $i \geq 2$.

This is not necessarily true.

For example, if $L = \{0, 1, 2\}^*$, 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 in the form $uv^i w$ for some $i \geq 2$.

This is simply false, as there exist strings in $L$ that contain no nonempty substring of the form $xx$ — this was first proved by the Norwegian mathematician Axel Thue in 1906.
Error #7. Assuming that all long strings in a regular language $L$ can be written as $uv^iw$ for some $i \geq 2$.

Similarly, the pumping lemma proves that if a language is regular and infinite, it must contain a subset of the form $uv^*w$ for some nonempty string $v$.

But this is not true of all infinite languages.

For example, the set of all strings over 0, 1, 2 having no nonempty substring of the form $xx$ is infinite.
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!
Error #8. Trying to use the pumping lemma to prove that a language is regular.

One example is the following nonregular language:

\[ L = \{ a^j b^k c^\ell : j = 0 \text{ or } k = \ell \} . \]

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

**Case 1:** \( j = 0 \), so \( z = b^k c^\ell \) for some integers \( k, \ell \) with \( k + \ell \geq n \).
Consider the factorization \( z = uvw \), where \( u = \epsilon \), \( v = b \) (if \( k \geq 1 \)) or \( v = c \) (if \( k = 0 \)), \( w = \) the rest of the string. Then \( uv^i w \in L \) for all \( i \geq 0 \).

**Case 2:** \( j \neq 0 \), so \( z = a^j b^k c^k \), for some integers \( j, k \) with \( j \geq 1 \) and \( j + 2k \geq n \). Consider the factorization \( z = uvw \), where \( u = \epsilon \), \( v = a \), and \( w = \) the rest of the string. Then \( uv^i w \in L \) 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.

A very useful heuristic: 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 factoring $z = uvw$ 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 $L_z \subseteq L$, we know $uv^iw \notin L_z$.

Hence by the pumping lemma, $L_z$ itself would not be regular.

But $L_z$ is in fact regular — a contradiction.
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.

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

\[
L = \{ww : w \in \{a, b\}^*\}
\]

again.

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

However, the associated language is

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

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

So instead you would need to pick something like \( z = a^nba^n \).
I’ll end today’s lecture with a puzzle.

Suppose $L$ is a finite language.

Then $L$ is regular, so the pumping lemma applies to it.

But the pumping lemma says that there are $u, v, w$, with $v$ nonempty, such that $uv^i w \in L$ for all $i \geq 0$.

How can this be, if $L$ is finite?