CS 360
Fall 2021
The Pumping Lemma for Regular Languages

Jeffrey Shallit
School of Computer Science
University of Waterloo
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit
In order to *really* understand regular languages, we need to be able to tell when a language is *not* regular.

The main tool we will see in this course to do this is the *pumping lemma*.

The pumping lemma is an *implication*, not an equivalence. It says, “If a language $L$ is regular, then (some complicated condition) holds.”

So if the (complicated condition) *doesn’t* hold, we know $L$ is *not* regular.
What kinds of languages are not regular?

Of course, not all languages are regular.

What kinds of languages are \textit{not} regular?

One typical non-regular language is \( L = \{ a^i b^i : i \geq 0 \} \).

Our \textit{intuition} is that \( L \) is not regular because finite automata can’t count; they only have a finite number of states, so they can only “remember” finitely many bits. Recognizing \( L \) would seem to require counting the number of \textit{a}’s, which could be arbitrarily large.

But intuition like this \textit{doesn’t} constitute a formal proof. We need some way to convert our intuition that \( L \) is not regular into a rigorous proof.
Where our intuition might go wrong

To see how our intuition could fail, let’s look at another language where counting seems necessary to recognize it.

Define $|x|_w$ to be the number of occurrences of the string $w$ in $x$. For example, $|\text{banana}|_{\text{an}} = 2$.

Let’s look at the language

$$L_2 = \{x \in \{a, b\}^* : |x|_{ab} = |x|_{ba}\}.$$

Is $L_2$ regular? Our intuition might say “no”, because it would seem that we would have to count the number of $ab$’s in $x$ and compare it to the number of $ba$’s.
$L_2$ is actually regular

But $L_2 = \{x \in \{a, b\}^* : |x|_{ab} = |x|_{ba}\}$ is actually regular!

To see this, let’s keep track of both $c$, the last symbol seen, and the difference $d = |x|_{ab} - |x|_{ba}$. The trick is that this difference can only be 0, 1, or $-1$. So states are pairs $[c, d]$.

This gives the following DFA for $L_2$: 

![DFA Diagram](attachment:dfa-diagram.png)
Moral of the story

The moral is that we *can’t* always rely on our intuition to be a guide.

That’s why we need a formal proof!

So to claim a language is not regular, we need a rigorous proof.

The *pumping lemma* is one tool to provide that kind of proof.
Here is the statement of the pumping lemma. It’s rather long and complicated at first glance, so study it for a moment.

**Pumping Lemma.** Suppose $L$ is regular. Then
- there exists a constant $n > 0$ such that
- for all $z \in L$ with $|z| \geq n$
- there exists some factorization of $z$ into $z = uvw$, where $|uv| \leq n$ and $|v| \geq 1$ such that
- for all $i \geq 0$ we have $uv^i w \in L$.

In English, the pumping lemma asserts that if $L$ is regular, then every sufficiently long string $z \in L$ has a short prefix $uv$, with $v$ nonempty, such that we can “pump” the substring $v$ as many times as we like and still get a string in $L$. 
As an example of what the pumping lemma asserts, let’s apply it to a specific regular language: $L_3 = ad \cup bc(de)^*f$. This is regular because we gave a regular expression for it.

The pumping lemma says that if $z \in L_3$, and $z$ is long enough, then there’s some way to write $z = uvw$ so we can “pump” a nonempty $v$ and still get a string in $L_3$.

For example, suppose we chose $z = bcdedef$. Then we can write $z = uvw$ with $u = bc$, $v = de$, and $w = def$. Then $uv^i w \in L_3$ for all $i \geq 0$.

Notice that if we chose $z = ad$, we would be unable to pump, no matter how $u, v, w$ are chosen. That’s because $z$ is “too short”.

Example
Now that we’ve see how the conclusion of the pumping lemma holds when $L$ is regular, let’s see how we could use it to prove a language *not* regular.

One way is to assume that $L$ is regular, and use the pumping lemma to get a contradiction.

Start with $L = \{a^i b^i : i \geq 0\}$. Assume that $L$ is regular.

Then the pumping lemma asserts the existence of some constant $n$. Furthermore, it says that if we choose $z \in L$ such that $|z| \geq n$, then something holds. So we need to choose a $z$. The most obvious choice is $z = a^n b^n$. It’s in $L$, and its length is $2n \geq n$. 
Pumping lemma

Then the pumping lemma says that there’s some way to write $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$, such that something else holds.

So if $z = a^nb^n = uvw$ with $|uv| \leq n$, the only possible choice is that $u = a^j$, $v = a^k$, and $w = a^{n-j-k}b^n$ for some $j$ and $k$. And the length condition on $v$ assures us that $k \geq 1$.

Finally, the “something else” that holds is that $uv^iw \in L$ for all $i \geq 0$.

In particular, this must be true for $i = 0$. So $uv^0w = uv^0w = uw = a^j a^{n-j-k}b^n = a^{n-k}b^n$, which cannot possibly be in $L$ because $k \geq 1$. This gives us a contradiction: the pumping lemma asserted a string to be in $L$, but it’s not.

So our original assumption that $L$ is regular is wrong, and $L$ is not regular.
Proof of the pumping lemma

OK, now we’ve seen the statement, and a couple of examples. Now it’s time for the proof. Here’s the 30-second version:

1. A long $z \in L$ means a long acceptance path in a DFA.

2. A long acceptance path means some state is repeated, so there’s a loop in the acceptance path.

3. Then we can go around this loop as many times as we like to get longer and longer strings in $L$. See the diagram:
Proof of the pumping lemma

Now the 2-minute version of the proof:

If $L$ is regular, then it’s recognized by some DFA $M$.

Take a long string $z \in L$, write $z = a_1 a_2 \cdots a_r$ for some large $r$, and consider all the states encountered in the acceptance path for $z$. If $r$ is large enough compared to the number of states in $M$, then some state $q$ on the path must be repeated.

Let $u$ be the label of the path from $q_0$ to $q$, let $v$ be the label of the path from the first $q$ to the second $q$, and let $w$ be the label of the path from the second $q$ to a final state $q_f$.

Then $uv^i w$ is also accepted by $M$, because $u$ takes us to $q$; $v$ takes us back to $q$ (and so $v^i$ takes us back to $q$ over and over) and then $w$ takes us to $q_f$. 
Finally, the full formal proof, with more details:

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting $L$.

Let $n$, the pumping lemma constant, be $|Q|$, the number of states of $M$.

Suppose $z \in L$ and $|z| \geq n$. Write $z = a_1 a_2 \cdots a_r$, with $r \geq n$.
Then $\delta(q_0, z) = q_f$ for some final state $q_f$.

Define $q_j = \delta(q_0, a_1 a_2 \cdots a_j)$, the state of $M$ reached by reading the prefix of $z$ of length $j$. 
The pumping lemma proof

Then the total number of states reached while processing $z$ is $r + 1$: namely, $q_0$, $\delta(q_0, a_1), \ldots, \delta(q_0, a_1 a_2 \cdots a_r)$.

Now $r \geq n$, so $r + 1 \geq n + 1$. By the pigeonhole principle, some state is repeated. Even more is true: among the $n + 1$ states $\delta(q_0, a_1 \cdots a_k)$, $0 \leq k \leq n$, some state must be repeated.

(Aside: if $k = 0$ then $a_1 \cdots a_k = \epsilon$.)

Suppose the repeated state is $q = \delta(q_0, a_1 \cdots a_k) = \delta(q_0, a_1 \cdots a_\ell)$ for some $0 \leq k < \ell \leq n$.

Let $u = a_1 \cdots a_k$, $v = a_{k+1} \cdots a_\ell$, and $w = a_{\ell+1} \cdots a_r$. Then

$$\delta(q_0, u) = q$$
$$\delta(q, v) = q$$
$$\delta(q, w) = q_f.$$
It now follows that $\delta(q_0, uv^i w) = q_f$ for all $i \geq 0$.

Hence $uv^i w \in L$ for all $i \geq 0$.

Since $\ell > k$, we have $|v| = \ell - k \geq 1$.

Since $\ell \leq n$ we have $|uv| \leq n$.

That completes the full proof of the pumping lemma.
The most common use of the pumping lemma is to prove a language $L$ not regular.

We already saw one way to do this: assume $L$ is regular, and then use the pumping lemma to get a contradiction.

Another way, which is logically equivalent, is to use the contrapositive of the pumping lemma.

What is it? With all those quantifiers, it might be hard to figure out.
Here it is:

Pumping Lemma, Contrapositive Version.

Let $L$ be a language.

If for all $n \geq 1$

there exists $z \in L$ with $|z| \geq n$

such that for all factorizations $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$

there exists $i \geq 0$ such that $uv^iw \not\in L$

then $L$ is not regular.
The Pumping Game

Another way to think about the contrapositive of the pumping lemma (CPL) is as a two-player game that you are playing against an infinitely powerful (but honest) adversary.

You are trying to “win” by proving that $L$ is not regular. The adversary is trying to stymy you and “win” by making your proof fail.

Your moves are given by the “there exists” lines of the CPL.

The adversary’s moves are given by the “for all” lines of the CPL.

A proof is a winning strategy. This means that no matter what the adversary does on its moves, at the very end you have to produce the $i$ such that $uv^iw \notin L$. 
Rephrased, the CPL looks like:

You choose $L$, the language you want to prove nonregular.

Adversary chooses some $n \geq 1$.

You choose $z \in L$ with $|z| \geq n$.

Adversary chooses a factorization $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$.

You choose an $i \geq 0$ such that $uv^i w \not\in L$.

To “win” and prove $L$ nonregular, you must always win no matter what the adversary does at every stage.
Let’s play this game on the language \( L = \{ww : w \in \{a, b\}^*\} \).

The adversary chooses some \( n \geq 1 \).

Now you have to choose \( z \in L \) with \( |z| \geq n \). Not every choice will work! Let’s choose \( z = a^n b a^n b \).

The adversary chooses a factorization \( z = uvw \) with \( |uv| \leq n \) and \( |v| \geq 1 \). By the structure of \( z \) and the bounds on \( |uv| \) and \( |v| \), the only possibility is \( u = a^j, v = a^k \), and \( w = a^{n-j-k} b a^n b \), for some \( j \geq 0, k \geq 1, \) and \( j + k \leq n \).

Now you have to choose \( i \) so that \( uv^i w \not\in L \). One choice that works is \( i = 2 \). Then \( uv^i w = a^j a^{2k} a^{n-j-k} b a^n b = a^{n+k} b a^n b \).

Since \( k \geq 1 \), it is clear that this word is not in \( L \). We “win” the game and \( L \) is nonregular.
Continuing our “game” analogy, proving that \( L \) is nonregular via the CPL is like announcing a “forced checkmate” in 4 moves in chess.

Here a “forced checkmate” means that no matter what your opponent does on its two moves, you win.

Of course, you have to make the right responses to the adversary’s moves. Not every single response will work!

For example, suppose in the previous proof, we chose a different \( z \). For example, we might have been tempted to choose \( z = a^n a^n = a^{2n} \).

This is a bad move and you can’t win with it. Let’s see why.
Pumping

Adversary chooses $n$.

If you choose $z = a^{2n}$...

Then the adversary can choose $u = \epsilon$, $v = aa$, and $w = a^{2n-2}$.

Then no matter what $i$ you pick, we have $uv^iw = a^{2n+2i-2}$, which is in $L$.

So you lose. Losing doesn’t mean $L$ is regular (we already proved it isn’t!). It means *this particular attempt* to prove $L$ nonregular failed.
So choosing $z$ wisely and $i$ wisely can be the difference between a pumping lemma proof succeeding or failing.

In the next lecture, we’ll see some strategies for choosing $z$ and $i$. 