Pumping Lemma

How to prove that a language is regular?
- build a DFA or NFA
- build a regular expression
- use closure properties to reduce to other languages known to be regular

How to prove that a language $L$ is not regular?
- use closure properties to reduce from another language that is not regular
  eg., if $K$ is not regular and $L = \overline{K} = \{0, 1\}^* \setminus K$, then $L$ is not regular either
- but to get started, we can use the Pumping Lemma

Key point: DFAs are finite, i.e., have a finite number of states

This is useful, e.g., $L = \{\text{any particular finite set of strings}\}$ is regular (e.g., the texts of all books in the Library of Congress) by closure under complement and intersection, you can also add or remove any finite set of strings from a regular language, and still have a regular language.

But it is also their Achilles heel.

Finite set of states $\iff$ Finite memory

Any long enough input string will cause the machine to cycle.
Example: \[ L = \{ \epsilon, 0^n 1^n \mid n \geq 0 \} \]
\[ = \{ \epsilon, 01, 0011, 000111, \ldots \} \]
\[ \text{is not a regular language.} \]

Here is a computer program that tries to decide this language:

\[ n = 0 \]
while (input remains and next bit is 0):
\[ n += 1 \] and step forward
while (input remains and next bit is 1):
\[ n -= 1 \] and step forward
if (no input remains and \[ n = 0 \]):
\[ \text{ACCEPT} \]
else \[ \text{REJECT} \]

On any real computer, this program will not accept exactly \[ L \].

For long enough input strings, the register storing \[ n \] will overflow
and only works for small-enough inputs.

32 bits \( \Rightarrow \text{overflow after } 2^{32} \)
1 TB \( \Rightarrow \text{overflow after } 2^{8 \times 2^{20}} \)

We will see that finite-state machines suffer the same problem:

Proof: Assume \( L \) is regular. Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a DFA
accepting \( L(M) = L \). Let \( N = |Q| \).

Consider the strings \( \epsilon, 0, 00, 000, \ldots, 0^N \). Since there are
\( N+1 \) different strings and only \( N \) states, the machine must take too
of them to the same place, i.e., \( \exists m, n \in \{0, 1, 2, \ldots, N\}, m \neq n, \text{s.t.} \)
\[ \hat{\delta}(q_0, 0^m) = \hat{\delta}(q_0, 0^n) \].

By assumption, \( 0^m 1^m \in L(M) \) so \( \hat{\delta}(q_0, 0^m 1^m) \in F \).
\[ \Rightarrow \hat{\delta}(q_0, 0^m 1^m) = \hat{\delta}(\hat{\delta}(q_0, 0^m), 1^m) = \hat{\delta}(q_0, 0^m 1^m) \in F \]

Thus \( 0^m 1^m \in L(M) \), a contradiction since \( 0^m 1^m \notin L \).
This argument generalizes.

**Theorem ("Pumping Lemma")**: Let $L$ be a regular language. Then there exists a constant $n$ such that:

- For every $w \in L$ of length $|w| \geq n$,
  - $w$ can be written $w = xyz$ such that
    1. $xy^jz \in L$ for all $j \geq 0$
    2. $|y| > 0$
    3. $|xy| \leq n$.

\[
\begin{array}{cccc}
  & x & y & z \\
\end{array}
\]

"pumping": \[
\begin{array}{cccc}
  & x & 1 & z \\
  & x & y & y & y & y & y & \cdots \\
\end{array}
\]

Note: $|y| > 0$. (If $|y| = 0$ were a possibility, then the claim would be vacuous: $x \epsilon z = x \epsilon z + x$.)

Third condition says that "pumping" can start near the beginning of $w$. (This is often not needed, but is sometimes convenient.)

Remember this theorem, especially the first two conditions.

**Proof**: We prove.

**Lemma**: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $L(M) = L$.

Then the above conditions hold with $n = |Q| = \# \text{ of states}$.

If $L$ has no strings of length $\geq n$, then claim is vacuous.

Otherwise let $w \in L$ with $|w| \geq n$.

\[
\begin{array}{cccc}
  q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \xrightarrow{w_3} \cdots \xrightarrow{w_n} q_n \\
\end{array}
\]

Let $q_\delta = \delta(q_0, w_1, w_2, \ldots, w_\delta)$.

Since only a different states, two $q_\delta$ must be the same, i.e.

\[
q_\delta = q_\delta \text{ for some } 0 \leq \delta \leq n.
\]

Let $x = w_1, \ldots, w_i$, $y = w_{i+1}, \ldots, w_\delta$, $z = w_{\delta+1}, \ldots, w_n$

Since $q_i = q_\delta$.

\[
\begin{array}{ccc}
  q_0 \xrightarrow{x} q_i \xrightarrow{y} q_\delta \xrightarrow{z} q_n \\
  q_0 \xrightarrow{x} q_i \xrightarrow{y} q_\delta \xrightarrow{z} q_\delta \\
  q_0 \xrightarrow{x} q_i \xrightarrow{y} q_\delta \xrightarrow{z} q_n \\
\end{array}
\]

The loop can be repeated as often as we like (or 0 times). $\Box$
Corollary: $\Sigma^*\{1\}$ is not regular.

Proof by Pumping Lemma:

Assume otherwise. Let $N$ be the constant we get.

$w = 0^N1^N \in L$

$w = xy^z$ with $|xy| \leq N$, $|y| > 0$

thus $x = 0^i$, $y = 0^j$, $j > 0$, $z = 0^{N-i-j}$.

Apply the pumping zero times.

$\Rightarrow x \in L$

$= 0^i 0^{N-i-j} \in L = 0^N \in L$

$\Rightarrow \parallel$

Note third condition not necessary in this case:

00 ... 011 ... 1

Pumping $y$ two or more times would still give a contradiction.

Corollary: $\{w \in \{0,1\}^* | |w| = n \}$ is not regular.

Proof attempt 1: Let $N$ be const from pumping lemma, assuming $|w|

consider $w = 0^N1^N \in L$.

By lemma, can split $w = x y z$ s.t. $x y^z \in L$

But we do not get to choose this decomposition?

Could be: $x = \varepsilon$, $y = 0^j$, $z = 0^N - 1$.

Then $x y z = 0^N 0^1 \in L$. Not a contradiction?

Need to choose a better string to start with.

Proof attempt 2:

Try $w = 0^N1^N \varepsilon$

By pumping can split it up $w = x y z$.

What if $y = 0^1$, $x = 0^N1^N - 1$, $z = 1^N - 1$?

$\Rightarrow x y^z \in L$ not a contradiction?

Need to use the third condition.

Proof: $w = 0^N1^N$, and as before.

Alternative proof: By muntseh with $L(0^* 1^*)$...
Some good examples:

1. \( L = \bigcup_{n \geq 0} \Sigma^n \) is not regular

   Proof: Assume otherwise. Let \( h: \Sigma \rightarrow \{0,1\} \) be \( h(a) = 0 \iff a \).
   Then \( h(L) = \{0^n | n \text{ is prime}\} \).
   If \( L \) is regular, then so is \( h(L) \).
   Now apply pumping lemma \( \Rightarrow \) some \( N \)
   start with \( 0^p : p > N \) a prime
   \[
   \begin{array}{c}
   \text{pumping} \\
   m + k + l = p
   \end{array}
   \]
   \[
   h(L) = 0^m(0^k)^{m+k}0^l \in h(L)
   = 0^{(k+1)(m+1)} \in h(L)
   \]

2. \( \{0^m | n \geq m\} \)
   Proof: Reverse 1. Then pump.

3. \( \{0^n | n \geq 0\} \cup \{1^n | n \geq 0\} \)

   Proof: Assume otherwise.
   homomorphism to eliminate \( 0 \).
   Complement the language, intersect with \( a^*c^* \)
   gives \( \{a^*c^* \} \), homomorphism.

   Intuition is that if the DFA needs to count the language is not regular. But
   \[
   \frac{1}{2}(L) = \emptyset \neq 1 \times 1 \text{ for some } s \text{ with } 1w1 = 1x1, x \in \Sigma \}
   \]
   is regular.
   \[\text{If } L \text{ is regular, then so is } h(L) \]
   Proof: E-NFA \( N = (Q \times \Sigma, \epsilon, \delta, q_0, F) \)
   \[\rho(q_0, \epsilon) = \{q_0\} \times F \]
   \[\rho((q, \epsilon), a) = \delta(q, a) = q \times F \]

5. quotient \( \Sigma^* : w \in \Sigma^L \)

6. derivative \( \Sigma^*: w \in \Sigma^L \) (using closure under reversal and quotient)
   \[
   \Sigma^* = \{x | \text{number of 0l substrings = number of 0l substrings} \}
   \]
   is regular,
   no counter necessary. L can have two 0l's without a 10.