The Pumping Lemma for Regular Languages

✦ What is it?
   ✗ A statement ("lemma") that is true for all regular languages

✦ Why is it useful?
   ✗ Can be used to show that certain languages are not regular
   ✗ How? By contradiction: Assume the given language is regular and show that it does not satisfy the pumping lemma

✦ What is the idea behind it?
   ✗ Any regular language \( L \) has a DFA \( M \) that recognizes it
   ✗ If \( M \) has \( p \) states and accepts a string of length \( \geq p \), the sequence of states \( M \) goes through must contain a cycle (repetition of a state) due to the pigeonhole principle! Thus:
   ✗ All strings that make \( M \) go through this cycle 0 or any number of times are also accepted by \( M \) and should be in \( L \).

Formal Statement of the Pumping Lemma

✦ Pumping Lemma: If \( L \) is a regular language, then there exists a number \( p \) (the "pumping length") such that for all strings \( s \) in \( L \) such that \( |s| \geq p \), there exist \( x, y, \) and \( z \) such that \( s = xyz \) and:
  1. \( xy^iz \in L \) for all \( i \geq 0 \), and
  2. \( |y| \geq 1 \), and
  3. \( |xy| \leq p \).

✦ More Plainly: \( p = \) number of states of a DFA accepting \( L \). Any string \( s \) in \( L \) of length \( \geq p \) can be expressed as \( s = xyz \) where \( y \) is not null (\( y \) is the cycle), \( |xy| \leq p \) (cycle occurs within \( p \) state transitions), and any "pumped up" string \( xy^iz \) is in \( L \) for all \( i \geq 0 \) (go through the cycle 0 or more times).

✦ Proved in 1961 by Bar-Hillel, Peries and Shamir.
The Pumping Lemma

✦ Proof on the board…(see page 79 in textbook)
  ➔ See how it applies to \( \{w \mid \text{number of 0's in } w \text{ is not divisible by 3} \} \)

✦ In-Class Examples: Using the pumping lemma to show a language \( L \) is not regular
  ➔ 5 steps for a proof by contradiction:
    1. Assume \( L \) is regular.
    2. Let \( p \) be the pumping length given by the pumping lemma.
    3. Choose cleverly an \( s \) in \( L \) of length at least \( p \), such that
    4. For any way of decomposing \( s \) into \( xyz \), where \( |xy| \leq p \) and \( y \) isn't null,
    5. We can choose an \( i \geq 0 \) such that \( xy^iz \) is not in \( L \).

Proving non-regularity as a Two-Person game

✦ An alternate view of using the pumping lemma to show a language \( L \) is not regular
  ➔ Think of it as a game between you and an opponent:
    1. You: Assume \( L \) is regular
    2. Opponent: Chooses some value \( p \)
    3. You: Choose cleverly an \( s \) in \( L \) of length \( \geq p \)
    4. Opponent: Breaks \( s \) down into some \( xyz \), where \( |xy| \leq p \) and \( y \) is not null,
    5. You: Need to choose an \( i \geq 0 \) such that \( xy^iz \) is not in \( L \) (in order to win (the prize of non-regularity)!).

✦ See how this works for showing \( \{0^n1^n \mid n \geq 0\} \) is not regular.
The Pumping Lemma Song (by Harry Mairson)

Any regular language \( L \) has a magic number \( p \)
And any long-enough word \( s \) in \( L \) has the following property:
Amongst its first \( p \) symbols is a segment you can find
Whose repetition or omission leaves \( s \) amongst its kind.

So if you find a language \( L \) which fails this acid test,
And some long word you pump becomes distinct from all the rest,
By contradiction you have shown that language \( L \) is not
A regular guy, resilient to the damage you have wrought.

But if, upon the other hand, \( s \) stays within its \( L \),
Then either \( L \) is regular, or else you chose not well.
For \( s \) is \( xyz \), and \( y \) cannot be null,
And \( y \) must come before \( p \) symbols have been read in full.

R. Rao, CSE 322  Source: http://www.cs.brandeis.edu/~mairson/poems/node1.html