First we review the pumping lemma for regular languages.

**Definition 1.** We say that a language $L$ is **pumpable** iff

there exists an integer $p > 0$ such that
for all strings $s \in L$ with $|s| \geq p$,
there exist strings $x, y, z$ with $xyz = s$ and $|xy| \leq p$ and $|y| > 0$ such that
for every integer $i \geq 0$,
$xy^iz \in L$.

We proved this in class last time:

**Lemma 2** (Pumping Lemma for Regular Languages). *For any language $L$, if $L$ is regular, then $L$ is pumpable.*

Here is the contrapositive, which is an equivalent statement:

**Lemma 3** (Pumping Lemma (contrapositive form)). *For any language $L$, if $L$ is not pumpable, then $L$ is not regular.*

We will use the contrapositive form to prove that certain languages are not regular by showing that they are not pumpable. By definition, a language $L$ is not pumpable iff

for any integer $p > 0$,
there exists a string $s \in L$ with $|s| \geq p$ such that
for all strings $x, y, z$ with $xyz = s$ and $|xy| \leq p$ and $|y| > 0$,
there exists an integer $i \geq 0$ such that
$xy^iz \notin L$.

Here is a template for a proof that a language $L$ is not pumpable (and hence not regular). Parts in brackets are to be filled in with specifics for any given proof.

1. Given any $p > 0$,
   let $s = [describe some string in $L$ with length $\geq p]$.
2. Now for any $x, y, z$ with $xyz = s$ and $|xy| \leq p$ and $|y| > 0$,
   let $i = [give some integer $\geq 0$ which might depend on $p, s, x, y, and z]$.
3. Then we have $xy^iz \notin L$ because [give some reason/explanation].

Note:
• We cannot choose \( p \). The value of \( p \) could be any positive integer, and we have to deal with whatever value of \( p \) is given to us.

• We can and do choose the string \( s \), which may differ depending on the given value of \( p \) (so the description of \( s \) uses \( p \) somehow). We must choose \( s \) to be in \( L \) and with length \( \geq p \), however.

• We cannot choose \( x \), \( y \), or \( z \). These are given to us and could be any strings, except we know that they must satisfy \( xyz = s \), \( |xy| \leq p \), and \( |y| > 0 \).

• We get to choose \( i \geq 0 \) based on all the previous values.

**Example:** Let

\[ L = \{ w \in \{0,1\}^* \mid w \text{ has more 0's than 1's}\}. \]

We show that \( L \) is not pumpable using the template:

Given any \( p > 0 \),

- let \( s = 0^p1^{p-1} \). (Clearly, \( s \in L \) and \( |s| \geq p \).)
- Now for any \( x, y, z \) with \( xyz = s \) and \( |xy| \leq p \) and \( |y| > 0 \),
- let \( i = 0 \).

Then we have \( xy^iz = xy^0z = xz \notin L \), which can be seen as follows: Since \( |xy| \leq p \) it must be that \( x \) and \( y \) consist entirely of 0’s, and so \( y = 0^m \) for some \( m \), and we further have \( m \geq 1 \) because \( |y| > 0 \). But then \( xz = 0^{p-m}1^{p-1} \), and so because \( p-m \leq p-1 \), the string \( xz \) does not have more 0’s than 1’s, and thus \( xz \notin L \).

**Exercises**

1. (Exercise 4.1.1 (selected items)): Prove that the following are not regular languages. For each, show that the given language is not pumpable. You may use the template given above.

   (a) The set of strings of balanced parentheses. These are the strings of characters "(" and ")" that can appear in a well-formed arithmetic expression.
   (b) \( \{0^n1^n \mid n \geq 1\} \).
   (c) \( \{0^n1^m2^n \mid n \text{ and } m \text{ are arbitrary integers}\} \).
   (d) \( \{0^n1^{2n} \mid n \geq 1\} \).

2. (! Exercise 4.1.2 (selected items)): Prove that the following are not regular languages. [Do as many as you can.]

   (a) \( \{0^n \mid n \text{ is a perfect square}\} \).
   (b) \( \{0^n \mid n \text{ is a perfect cube}\} \).
   (c) \( \{0^n \mid n \text{ is a power of 2}\} \).
   (d) The set of strings of 0’s and 1’s that are of the form \( ww \), that is, some string repeated.
   (e) The set of strings of 0’s and 1’s of the form \( w\overline{w} \), where \( \overline{w} \) is formed from \( w \) by replacing all 0’s by 1’s, and vice versa; e.g., \( 01100 = 100 \), and 011100 is an example of a string in the language.
3. Recall that a string $x$ is a *subsequence* of a string $y$ (written $x \preceq y$) if the symbols of $x$ appear in $y$ in order (although not necessarily contiguously). For language $L \subseteq \Sigma^*$, define

$$\text{SUBSEQ}(L) := \{ x \in \Sigma^* : (\exists y \in L)[x \preceq y] \},$$

that is, SUBSEQ($L$) is the set of all subsequences of strings in $L$. For example, if $L = \{aabc, cab\}$, then $\text{SUBSEQ}(L) = \{\epsilon, a, b, c, aa, ab, ac, bc, aab, aac, abc, aabc, ca, cb, cab\}$.

Show that if $L$ is regular, then SUBSEQ($L$) is regular. [Hint: Two methods will work here: (1) transforming a regular expression for $L$ into a regular expression for SUBSEQ($L$); (2) transforming an $\epsilon$-NFA for $L$ into an $\epsilon$-NFA for SUBSEQ($L$). By the way, it is known that if $L$ is any language whatsoever, then SUBSEQ($L$) is regular, but the proof of this fact is not constructive.]

4. (! (not in the text; optional)) Let $x$ and $y$ be any two strings. A *merge* of $x$ and $y$ is any string obtained by merging the symbols of $x$ with those of $y$ in some arbitrary way, maintaining the order of the symbols from each string. More exactly, if $|x| = m$ and $|y| = n$, then a string $z = z_1 \cdots z_k$ is a merge of $x$ and $y$ if and only if

- $k = m + n$,
- there exist $1 \leq i_1 < i_2 < \cdots < i_m \leq k$ such that $x = z_{i_1} z_{i_2} \cdots z_{i_m}$,
- there exist $1 \leq j_1 < j_2 < \cdots < j_n \leq k$ such that $y = z_{j_1} z_{j_2} \cdots z_{j_n}$, and
- $\{i_1, \ldots, i_m\} \cap \{j_1, \ldots, j_n\} = \emptyset$.

For example, there are five different merges of the strings $ab$ and $bc$:

$$abab \ abca \ baab \ acab \ bca$$

Let $A$ and $B$ be any languages over the same input alphabet $\Sigma$. Define

$$A \text{ merge } B := \{ z \in \Sigma^* \mid z \text{ is a merge of some } x \in A \text{ and some } y \in B \}.$$  

Show that if $A$ and $B$ are regular, then so is $A \text{ merge } B$. *Hint:* Given a DFA for $A$ with $r$ many states and an DFA for $B$ with $s$ many states, you can construct an NFA for $A \text{ merge } B$ with $rs$ many states.

5. (* Exercise 4.4.1(a,b)): Do Exercise 4.4.1(a,b).

6. (Exercise 4.4.2(a,b)): Do Exercise 4.4.2(a,b).

7. Consider the palindrome grammar of Figure 5.1. Give leftmost and rightmost derivations of the strings 1001001 and 110011.

8. Consider the simple expression grammar $G = (\{E, I\}, T, P, E)$, where the productions of $P$ are given in Figure 5.2. Give two different leftmost derivations for the string $a1 + b0 * a00$.

9. Describe briefly in words the language $L(G)$, where $G = (\{A, B\}, \{a, b, c\}, P, A)$ is a context-free grammar and the productions in $P$ are

$$A \rightarrow aAc$$
$$A \rightarrow B$$
$$B \rightarrow \epsilon$$
$$B \rightarrow Bc$$
10. Do the grammars with productions given by

\[
\begin{align*}
A & \rightarrow \epsilon \\
A & \rightarrow Aa
\end{align*}
\]

and

\[
\begin{align*}
A & \rightarrow \epsilon \\
A & \rightarrow aA
\end{align*}
\]

describe the same language? Explain.

11. Do Exercise 5.1.1(a,b,c).

12. Do Exercise 5.1.2(a,b,c).

13. Do Exercise 5.1.7(a,b).

14. Consider the grammar of Exercise 5.1.8. Show that \textit{abba} is generated by the grammar but \textit{aba} is \textit{not} generated by the grammar. (This is a special case of the full exercise.)

15. Do Exercise 5.2.1(a,b,c).

16. Do Exercise 5.3.1.

17. (Optional) The grammar of the last exercise has one variable and three productions. There is an equivalent grammar with one variable and only two productions. Can you find it?