Exercise

Prove, by using the pumping lemma, that the language
\[ L = \{ a^i b^j | j \geq i \} \]
is not regular.

Solution

Remember that the pumping lemma tells us that if the words of a
language cannot be "pumped," then the language is not regular.

Let \( m \) be a sufficiently large integer for the pumping lemma,
and let \( w \) be a word in \( L \), with \(|w| \geq m\); this can always hold
if we choose \( m \) large enough.

\[ w = x y z \]

There are three cases:

(a) \( y = a^k \) for some \( k \in \mathbb{N} \);
    in this case \( x y^2 z \notin L \) because \( x y^2 z \) has more \( a \)'s than \( b \)'s.

(b) \( y = b^k \) for some \( k \in \mathbb{N} \);
    analogous to the previous case.

(c) \( y = a^k b^k \) for some \( k \);
    in this case \( x y^2 z \notin L \) because symmetry is lost.

(d) \( y = a^k b^k \) for some \( k, k \in \mathbb{N} \); \( k \neq k \)
    else in this case symmetry is lost in \( x y^2 z \).

We have seen that there is no way to "pump" words of \( L \),
therefore \( L \) is not regular.
Exercise

Prove that the language
\[ L = \{ wu^k v w \mid u, v \in \{a, b\}^* \} \]
is not regular.

Note that \( L \) is the language of palindromes on \( \{a, b\} \).

Solution

It is immediate to notice that we can pump strings of \( L \)
only in the middle of the string; this for symmetry reasons.
Let \( z = u^k v w u^k \in L \), we denote \( u = \alpha \), \( v = \sigma \), \( w = \upsilon \),
and \( u^k v w u^k \in L \). We can pump \( \alpha \sigma \upsilon \) at will, still obtaining palindromic strings (which obviously are in \( L \)):

\[
\begin{align*}
\alpha^k \in L \\
\upsilon \sigma \upsilon \sigma \in L \\
\upsilon \sigma \upsilon \sigma \upsilon \sigma \upsilon \in L \\
\alpha^k \sigma \upsilon \sigma \in L \\
\end{align*}
\]

Notice that this is the only way we can pump; in general, to prove
that a language is not regular by showing that strings cannot be
pumped, one has to consider all possible ways of pumping.

In this case, however, the condition \( \alpha^k \sigma \upsilon \sigma \in L \) for the first part
of the given string \( u^k \sigma \upsilon \sigma \upsilon \sigma \) does not hold; in fact, the string
that is pumped is arbitrarily far from the beginning of the
string. We can finally conclude that \( L \) is not regular.
Exercise

Prove that the language $L = \{ a^k \mid k \text{ is prime} \}$ is not regular.

solution

We apply the pumping lemma and show that there is no way to pump. Without loss of generality, we choose $|w| = m \cdot n$, where $m \in \mathbb{N}$, $n$ is the constant of the pumping lemma, and $m \in \mathbb{N}$.

Now let $w = x y z$, with $|y| = m$. From the pumping lemma, $x y^i z \in L$ for all $i \in \mathbb{N}$; we choose $i = m + 1$, and we have $w' = x y^m z$.

$$|w'| = |x| + |y^m| = |x| + m |y| = m(1 + |y|)$$

which is not prime. Therefore, there is no way to pump and $L$ is not regular.
Exercise (4.3.4 from textbook)

Give an algorithm to tell whether two regular languages have at least one string in common.

Solution

It suffices to check whether the intersection \( L \) of the two languages, that we denote with \( L_1 \) and \( L_2 \), is empty.

We have \( L = L_1 \cap L_2 = \overline{L_1} \cup \overline{L_2} \).

An automaton accepting \( L \) can be easily constructed from automata accepting \( L_1 \) and \( L_2 \); emptiness of the language accepted by an automaton can be checked by checking reachability of a final state.

Notice that when constructing, for example, the automaton accepting \( \overline{L_1} \), the automaton accepting \( L_2 \) of which we invert final and non-final states has to be deterministic; otherwise, we do not obtain an automaton accepting \( \overline{L_2} \).

Another solution consists in the direct construction of the automaton accepting \( L_1 \cap L_2 \).
Exercise

Construct a DFA accepting the language denoted by the regular expression

$$L = (00+0)^*(10)^* + (101)^*$$

Solution

We start with the expression $$(00+0)^*$$; we can make initial state and final states collapse.

We go on with other sub-expressions:

- $$0^*$$ accepts 0

- $$0^* 101^*$$ accepts 011

- $$0^* 101^*$$ accepts 0011
A DFA accepting the desired language is the following:

Now we transform this NFA into a DFA.

\[
\begin{align*}
\delta(A, a) &= B \\
\delta(B, a) &= C \\
\delta(B, b) &= DEF \\
\delta(C, a) &= B \\
\delta(DEF, a) &= CDG \\
\delta(DEF, b) &= B \\
\delta(CDG, a) &= B \\
\delta(CDG, b) &= DEF \\
\delta(DEF, c) &= DEF \\
\delta(DEF, d) &= DEF \\
\delta(DEF, e) &= BG \\
\delta(CDG, f) &= CDG \\
\end{align*}
\]
The final DFA is:

[Diagram of DFA with states and transitions]
Exercise

Consider a m-state DFA accepting a language L. Prove that if L ≠ ∅ then exists x ∈ L such that |x| < m.

Solution

Let w be the shortest string accepted by A. By contradiction, let |w| ≥ m; then, for the pumping lemma w = xyz, and x ∈ L, with |xz| < |w|.

Contradiction.

Exercise

Let A be a DFA with n states, accepting the language L. Prove that

L is infinite iff ∃ w ∈ L | n ≤ |w| < 2m

Solution

If by the pumping lemma, L contains infinite strings.

only-if

If L is infinite, then exists w ∈ L with |w| ≥ 2m. If |w| < 2m we are done; otherwise, we apply the pumping lemma: w = xyz, x ∈ L. Note that |xz| < m and |yz| ≥ m. We can repeat this step iteratively until we obtain a string of L of the desired length. Note that the fact that |yz| ≥ m implies |y| ≤ m, so that we are guaranteed that we cannot obtain a string of length < m from a string of length ≥ 2m.
To check whether the language $L$ accepted by a DFA is empty, finite or infinite, we feed the automaton with all strings of length $\leq 2m$, where $m$ is the number of states. We have the following cases:

(a) If no string is accepted, $L$ is empty;
(b) If all accepted strings have length $<m$, the language is finite;
(c) If there is an accepted string of length $\geq m$, the language is infinite.