Pumping Lemma for Regular Languages

A language $\mathcal{L}$ is *regular* if it is it exactly the set of words accepted by some automaton M. We shall consider a property of all regular languages, which also provides us with a technique to prove that some languages are not regular.

Pumping Lemma. Let $\mathcal{L}_M$ be the language accepted by the automaton $M = \{A, S, \nu, s_0, F\}$[1] where the number of the states (the cardinality of S) is n. For every string w in $\mathcal{L}_M$ of length $|w| \geq n$, we can break w into three string $w = xyz$ such that

1. $|y| > 0$;
2. $|xy| \leq n$;
3. for all $k \geq 0$, the string $xy^k z$ is also in $\mathcal{L}_M$.

Proof. Let $w \in \mathcal{L}_M$ be a string of length $m \geq n$, i.e., $w = a_1 a_2 \ldots a_m$, and let $s_0 s_1 s_2 \ldots s_m$ a list of the states the automaton such that

(i) s_0 is the initial state and

(ii) $\nu(s_i, a_{i+1}) = s_{i+1}$, for all i such that $0 \leq i < m$.

Namely, s_{i+1} is the state which M is in after reading $a_1 \ldots a_{i+1}$. Since $w \in \mathcal{L}_M$ is accepted by the automaton M, the state w_m is final ($w \in F$). Since M has only n states, it is not possible for the states $s_0, \ldots, s_m$ to be all distinct (*pigeon-hole principle*). Therefore we can find natural numbers i and j with $0 \leq i < j \leq n$ such that $s_i = s_j$. Now we can break $w = xyz$ as follows:

1. $x = a_1 \ldots a_i$, the subword M reads before reaching state s_i starting from state s_0;
2. $y = a_{i+1} \ldots a_j$, the string M reads before passing from state s_i to state $s_j = s_i$;
3. $z = a_{j+1} \ldots a_m$, the rest of the string, which is read before reaching the accepting state s_m from state s_j.

---

[1] Here A is the alphabet, S is the set of internal states, ν is the transition function, s_0 is the initial state and F is the set of final, or accepting, states.

Notice that x may be empty (in this case, $i = 0$) and z may be empty (in this case $j = n = m$) but the string y is nonempty, by the assumption that $i < j$.

Consider the behaviour of M with the inputs $xy^k z$ for $k \geq 0$. If $k = 0$, then after reading x the automaton M is in state s_i and reads the string z; since $s_i = s_j$, after reading the string z the automaton M reaches s_m from s_i. But s_m is an accepting state, hence M accepts the word xz, so xz must be in $\mathcal{L}_M$. If $k > 0$ then after reading x the automaton loops from s_i to s_i k times while reading y^k and then, after reading z, it reaches the accepting state s_m. Hence M accepts $xy^k z$ and $xy^k z$ is in $\mathcal{L}_M$. The proof is finished.

Example: We may use the Pumping Lemma to show that some languages are not regular. Let A be the alphabet $\{0, 1\}$ and let

$$\mathcal{L}_{eq} = \{w \in A^* \mid w \text{ contains the same number of 0's and 1's } \}.$$

Let M be any automaton which accepts all the words in $\mathcal{L}_{eq}$; we show that M accepts also words which are not in $\mathcal{L}_{eq}$. Let n be the number of states of M and consider the word $w = 0^n 1^n$: certainly w belongs to $\mathcal{L}_{eq}$. If $s_0 s_1 \dots s_n \dots s_{2n}$ is the list of the states M is in while reading w, from the initial state s_0 to the accepting state s_{2n}, then we must have $s_i = s_j$ with $i < j \leq n$ (as M has only n states). Therefore $w = xyz$ with $x = a_1 \dots a_i$, $y = a_{i+1} \dots a_j$ where x and y consist only of 0's. By the Pumping Lemma, any word of the form $xy^k z$ for $k \geq 0$ is also accepted by M. Hence xz is also accepted by M; but xz has less 0's than 1's (as y contained only 0's and has been removed from w); thus xz does not belong to $\mathcal{L}_{eq}$. Therefore the set $\mathcal{L}_M$ of words accepted by M is *strictly larger* than the language $\mathcal{L}_{eq}$. Since this fact holds for any automaton M, the language $\mathcal{L}_{eq}$ is not regular.

Exercise: Show that the set $\mathcal{L}_{(,)}$ of strings of balanced parentheses is not regular. (*Hint:* Let M be any automaton accepting $\mathcal{L}_{(,)}$, let n be the number of states in M, let 0 be "(" and 1 be ")" in the word w of the above example.)

Reference: J. H. Hopcroft, R. Motwani and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley, Second Edition, 2001, pp. 126-130.