Automata Theory - Part III

Richard Cole
September 25, 2006

We turn now to a method for demonstrating that some languages are not regular. For example, as we will see, \( L = \{ a^n b^n | n \geq 1 \} \) is not a regular language. We use a proof by contradiction. So suppose for a contradiction that \( L \) were regular. Then there must be a DFA \( M \) that accepts \( L \). It has some number \( k \) of states. Let’s consider feeding \( M \) the input \( a^h b^j \). Look at the sequence of \( k+1 \) states (vertices) \( M \) goes through on reading \( a^h : q_0 = q_{i_1}, q_{i_1}, \ldots, q_k \).

As there are only \( k \) distinct states, there is at least one state (vertex) that is visited twice in this sequence, \( q_h = q_i \), say, for some pair \( h, j, 0 \leq h < j \leq k \). This can be drawn as follows.

In fact we see that \( q_{i+j} = q_{i+h} \), if \( j + 1 \leq k \), \( q_{i+k} = q_{i+h} \), if \( j + 1 \leq h \), and so forth. But all we will need for our result is the presence of one loop in the path, so we will stick with the representation in the figure above.

It is helpful to partition the input into four pieces: \( a^h, a^{i-h}, a^{k-j}, b^j \). The first \( a^h \) takes \( M \) from state \( q_0 \) to \( q_k \), the next \( a^{i-h} \) takes \( M \) from \( q_k \) to \( q_i \), the final \( a^{k-j} \) takes \( M \) from \( q_i \) to \( q_k \), and \( b^j \) takes \( M \) from state \( q_k \) to an accepting state. What happens on input \( a^h a^{i-h} a^{k-j} b^j = a^{i+j-k-h} b^j \)? The initial \( a^h \) takes \( M \) from \( q_0 \) to \( q_k \), the first \( a^{i-h} \) takes \( M \) from \( q_k \) to \( q_i \), the second \( a^{k-j} \) takes \( M \) from \( q_i \) to \( q_k \), the \( a^{i+j-k-h}b^j \) takes \( M \) from \( q_k \) to \( q_k \), and the \( b^j \) takes \( M \) from \( q_k \) to an accepting state. So \( M \) accepts \( a^{i+j-k-h}b^j \), which is not in \( L \) as \( j - h > 0 \). This is a contradiction, and thus the initial assumption, that \( L \) was regular, must be incorrect.

We now formalize the above approach in the following lemma.
Lemma 1 (Pumping Lemma) Let \( L \) be a regular language. Then there is a number \( p = p_L \), the pumping length, for \( L \), with the property that for each string \( s \) in \( L \) of length at least \( p \), \( s \) can be written as the concatenation of 3 substrings \( s = xyz \), which satisfy the following conditions.

1. \( |y| > 0 \)
2. \( |xy| \leq p \)
3. For each \( i \geq 0 \), \( xy^iz \in L \).

Proof. Again we use a proof by contradiction. So suppose that \( L \) were regular and let \( M \) be a DFA accepting \( L \). Let \( M \) have \( p \) states, \( q \), the number of states in \( M \), will be the pumping length for \( L \). Write \( s = s_1s_2 \cdots s_n \), where each \( s_i \) is a character in \( \Sigma \), the alphabet for \( L \). Consider the substring \( s' = s_1s_2 \cdots s_j \). We look at the path \( M \) follows on input \( s' \). It must go through \( p+1 \) vertices (states), and as \( M \) has only \( p \) vertices (states), and as \( M \) has only \( p \) vertices (states), at least one vertex (state) must be repeated. Let \( q_0 = q_{0h}q_{0h} \cdots q_0 \) be this sequence of states and suppose that \( q_0 = q_h, 0 \leq h < j \leq p \) is a repeated state.

Let \( x \) denote \( s_1 \cdots s_h \), let \( y \) denote \( s_{h+1} \cdots s_j \), \( z \) denote \( s_{j+1} \cdots s_{p+1} \cdots s_n \). As \( j > h \), \( |y| = j - h > 0 \). Also \( |xy| = |s_1 \cdots s_j| = j \leq p \). So (1) and (2) are true.

Clearly, \( xz \) is accepted by \( M \), for \( x \) takes \( M \) from \( q_0 \) to \( q_h \) and \( z \) takes \( M \) from \( q_h \) to a final state.

Similarly \( xyz \) is accepted by \( M \) for any \( i \geq 0 \), for \( x \) takes \( M \) from \( q_i \), to \( q_h \), each repetition of \( y \) takes \( M \) from \( q_h \), and then the \( z \) takes \( M \) from \( q_h \) to a final state. Thus (3) is also true, proving the result. \( \square \)

Now to show a language \( L \) is non-regular we use the pumping Lemma in the following way. We begin by assuming \( L \) is regular so as to obtain a contradiction. Next, we assert that there is a pumping length \( p \) such that for each string \( s \) in \( L \) of length at least \( p \) the three conditions of the Pumping Lemma hold. This (and more substantial) task is to choose a particular string \( s \) to which we will apply the conditions of the Pumping Lemma, and condition (3) in particular, so as to obtain a contradiction.

Let us look at the language \( L = \{a^n b^n | n \geq 1 \} \) again. Recall that we chose the string \( s = a^p b^p \). The reason that this was a good choice is that the pumping enabled by the Pumping Lemma occurs in the first \( p \) symbols of \( s \); if in Rule (3) we choose \( i > 1 \), \( i = 2 \) to be specific, this creates a string supposedly in \( L \) with more than \( p \) \( a \)'s but with exactly \( p \) \( b \)'s, which is clearly impossible.

Let me stress the sequence in which the argument goes. First the existence of pumping length \( p \) for \( L \) is asserted (different regular languages may have different pumping lengths, but each regular language with long enough strings will have a pumping length). Then a suitable string \( s \) is chosen. The length of \( s \) will be a function of \( p \). \( s \) is chosen so that when it is pumped a string outside of \( L \) is obtained. An important point about the pumped
substring $y$ is that while we know it occurs among the first $p$ characters of $s$, we do not know exactly which ones form the substring $y$. Consequently, a contradiction must arise for every possible substring $y$ of the first $p$ characters in order to show that it is not regular. Further examples will be given in class.

**Question** Does the proof of the Pumping Lemma work if we consider a $p$-state NFA that accepts $L$, rather than a $p$-state DFA?