The Pumping Lemma

February 1, 2012

Goals

• Understand what the Pumping Lemma says
• See the proof
• Practice working through some examples

The Pigeonhole Principle

Theorem 1. You cannot put \( m > n \) pigeons in \( n \) pigeonholes without putting multiple birds in one of the pigeonholes. In general, given two finite sets \( A \) and \( B \), if \(|A| > |B|\), then there is no one-to-one mapping \( f : A \rightarrow B \).

This is intuitively obvious, but, as an aside, it is interesting to try and write out a proof. In order to make your intuition precise, you will need to be able to state exactly what you mean by “the set \( X \) is bigger than the set \( Y \)” At any rate, it would be a distraction from the issue at hand.

The Pumping Lemma

Recall that a regular language is any set of string that is the language of some DFSA \( A \). Recall that a DFSA is a machine which operates only by moving through a finite set of states, reading a string one symbol at a time. It operates using a transition function \( \delta \), so that we interpret \( \delta(q, a) = p \) as “if the machine is in state \( q \) and reads an \( a \), it will move to state \( p \)”; we extend this to strings using an extended transition function \( \hat{\delta} \), so that we interpret \( \hat{\delta}(q, w) = p \) similarly, but with a sequence of symbols rather than just one. The extended transition function, recall, was defined as:

\[
\hat{\delta}(q, x) \begin{cases} 
q & |x| = 0 \ (\Rightarrow x = \varepsilon) \\
\delta(q, x) & |x| = 1 \\
\delta(\delta(q, a), w) & x = aw, |a| = 1, |w| \geq 1 \\
\text{undefined} & \quad \quad \quad \quad \text{undefined} \\
\hat{\delta}(\delta(q, a), w) & x = aw, |a| = 1
\end{cases}
\]
Theorem 3. Given an automaton with transition function \( \delta \), if \( w = xz \), then \( \hat{\delta}(q,w) = \hat{\delta}(\hat{\delta}(q,x),z) \).

Proof. By induction on the length of \( x \). For \( |x| = 1 \), the claim holds by the definition of \( \hat{\delta} \). Suppose the claim holds for any \( x \) with \( |x| = k \). Consider some \( x \) with \( |x| = k + 1 \). Let \( x = ay \), \( |a| = 1 \), \( |y| = k \). For any \( z \),

\[
\hat{\delta}(q,xz) = \hat{\delta}(q,ayz) = \hat{\delta}(\hat{\delta}(\delta(q,a),y),z) \]

by the inductive hypothesis. But \( \hat{\delta}(\delta(q,a),y,z) = \hat{\delta}(\hat{\delta}(q,a),z) = \hat{\delta}(\hat{\delta}(q,x),z) \) and the claim holds by induction.

Consider the fact that, in order for such a machine to be able to read arbitrarily long strings—that is, in order for the machine not to have a maximum string length, \( p \)—there must be a cycle in the state graph. But the fact that the current state fully defines the action of the machine on reading a given string \( y \) means that, if \( y \) leads us through a cycle, then we can not only read \( y \), but also \( yy \), \( yyy \), and so on, and for the machine all will be equivalent.

Definition 4. Given a regular language \( L \), we say string \( w \in L \) contains a substring \( y \) that can be pumped in \( L \) if we can decompose \( w \) into \( x, y, z \) such that:

(i) \( w = xyz \)
(ii) \( y \neq \varepsilon \)
(iii) For all \( i \geq 1 \), \( x y^i z \in L \)

Theorem 5. (Pumping lemma for regular languages.) For every regular language \( L \), there is some maximum length \( p \), a pumping-lemma constant, beyond which any string \( w \in L \) with \( |w| > p \) will have a substring \( y \) that can be pumped in \( L \). Furthermore, if \( p \) is a pumping-lemma constant such that \( y \) can be pumped, then \( y \) can always be found within the first \( p \) symbols of \( w \): if \( w = xyz \), then \( |xy| \leq p \).

Proof. Let \( L \) be a regular language and let \( A \) be a DFSA with \( L(A) = L \). Let \( k \) be the number of states in \( A \), and consider a string \( w \in L \) of length \( |w| > k \). We will show that \( k \) is a pumping-lemma constant, and, thus, that a pumping-lemma constant exists for any regular language.

First we observe that there must be at least one state that the automaton would pass through more than once while reading \( w \): consider the function which tells us the state of the automaton after reading the first \( m \) characters of \( w \), \( D(m) = \hat{\delta}(q_0,w^m_1) \), where \( w^m_1 \) is the prefix of \( w \) up to the \( m \)th symbol. The Pigeonhole Principle tells us that either \( D \) is not defined for more than \( k \) distinct integers, or \( D \) must map two different integers to the same state; since \( A \) accepts \( w \), the first cannot be true. Thus the automaton must pass through at least one state multiple times on reading \( w \).

Now consider the first two distinct positions \( i \) and \( j \) such that \( D(i) = D(j) \), \( j > i \). Let \( x = w^i_1 \), \( y = w^j_{i+1} \), and let \( z \) be the remainder of \( w \), so that \( w = xyz \). We first show that \( y \) can be pumped. Clearly \( y \neq \varepsilon \), since \( j > i \). To see that \( xy^m z \in L \) for any \( m \), let \( q^i \triangleq \hat{\delta}(q_0,x) \). We know that \( \hat{\delta}(q_0,xy) = q^i \), and thus, by Theorem 3, \( \hat{\delta}(\hat{\delta}(q_0,x),y) = \hat{\delta}(q^i,y) = q^j \). Now suppose as an inductive hypothesis that \( \hat{\delta}(q^i,y^m) = q^j \). Then \( \hat{\delta}(q^i,y^{m+1}) = \hat{\delta}(q^i,y^my) = \hat{\delta}(\hat{\delta}(q^i,y^m),y) = \hat{\delta}(q^j,y) \). Thus by induction we see that \( \hat{\delta}(q^i,y^m) = q^j \) for any \( m \). Thus \( \hat{\delta}(q_0,xy^m) = \hat{\delta}(q^i,y^m) = q^j \), and if \( \hat{\delta}(q_0,xyz) = \hat{\delta}(q^i,z) = q_F \in F \), then \( \hat{\delta}(q_0,xy^mz) = \hat{\delta}(q^i,z) = q_F \), and \( y \) can by pumped.

To complete the proof, we show that \( |xy| \leq k \). Recall that \( i \) and \( j \) are the first positions in \( w \) such that \( D(i) = D(j) \). Thus all \( q^l \triangleq \hat{\delta}(q_0,w^l_1) \) are distinct, for \( 1 \leq l \leq j - 1 \). By the Pigeonhole Principle, either \( j - 1 < k \), or there must be two integers less than \( j \) that \( D \) maps to the same state; but we have just noted that the second cannot be true. Thus \( j - 1 < k \Rightarrow j \leq k \).

Fact. We can extend the notion of “pumping” to: for all \( i \geq 0 \), \( xy^iz \in L \). The pumping lemma will still hold.

Example 6. Prove that \( a^n b^n \) is not a regular language.
Proof. Suppose the language were regular. Then there would be some pumping lemma constant \( p \). Surely \( a^p b^p \in L \). The pumping lemma tells us that there is a prefix of \( a^p b^p \) which is of length \( \leq p \), part of which can be pumped in \( L \). But since any prefix of \( a^p b^p \) of length \( \leq p \) must consist entirely of \( a \)'s, pumping any substring of length \( k \) would imply that \( a^{p+k} b^p \in L \), \( a^{p+2k} b^p \in L \), and so on. These are not in the language by definition, and so \( a^n b^n \) cannot be a regular language.

Example 7. Prove that \( w w^R \) is not a regular language, where \( .^R \) denotes the reversal of a string, and \( w \) is any string over \( \{a,b\} \).

Proof. Suppose the language were regular. Then there would be some pumping lemma constant \( p \). Surely \( a^p bba^p \in L \). The pumping lemma tells us that there is a prefix of \( a^p bba^p \) which is of length \( 0 < k \leq p \), part of which can be pumped in \( L \). But since any prefix of \( a^p bba^p \) of length \( \leq p \) must consist entirely of \( a \)'s, pumping any substring of length \( k \) would imply that \( a^{p+k}bba^p \in L \). Suppose that we could decompose \( a^{p+k}bba^p \) as some string \( w \) followed by its reversal. If \( |w| \leq p+k \), then \( w \) contains no \( b \)'s; but clearly the remainder of the string contains two \( b \)'s, and thus \( w \) cannot be followed by its reversal. Similarly, if \( |w| \geq p+k+2 \), then \( w \) contains two \( b \)'s, while the remainder of the string contains no \( b \)'s, and thus \( w \) cannot be followed by its reversal for the same reason. Thus \( |w| = p+k+1 \) if it exists; but this implies that \( w = a^{p+k}b \) and \( w^R = ba^p \), which is clearly impossible for \( k > 0 \). Thus we have a contradiction and we conclude that the language cannot be regular. \( \square \)