CSE 322 - Introduction to Formal Methods in Computer Science
The Pumping Lemma for Regular Languages

Dave Bacon
Department of Computer Science & Engineering, University of Washington

So far we have talked about regular languages and showed that they are captured in a variety of different models: they are accepted by NFAs and they correspond to the languages of regular expressions. In other words, we have lived and breathed regular languages deep down to our very basic sole. But are all languages regular?

Consider a quintessential example, the language $L_b = \{w | w = 0^k 1^k, k \geq 1\}$. Is this language regular? Well consider a deterministic finite automata which is designed to accept L_b. Let that machine have q states. Now suppose that we consider the machine on inputs in the set $\{0^i\}$ with $1 \leq i \leq q$. By the pigeon hole principle, there must exists two inputs to the machine, call them 0^i and 0^j with $1 \leq i \neq j \leq q$ such that the machine is in the same state for these inputs. Now think about what happens when the machine enters such a state. I could have entered it with 0^i or 0^j. Suppose it enters on 0^i. Then the machine should accept if it next reads 1^i but reject if it next reads 1^j. But this contradicts the fact that it could have gotten to the state via reading 0^j. Thus it is impossible for the language L_b to be accepted by a DFA and hence the language is not regular. The above intuition is quantified by a lemma called the pumping lemma for regular languages.

I. THE PUMPING LEMMA

Lets begin by stating the pumping lemma and then given some idea about what it means and then proving it.

Pumping Lemma for Regular Languages If L is a regular language, then there exists a constant n (which depends on L) such that for every string w in the language L, such that the length of w is greater than or equal to n, we can divide w into three strings, $w = xyz$ where

1. y is not the empty string, $y \neq \{\varepsilon\}$.
2. The size of xy is less than or equal to n: $|xy| \leq n$.
3. For all $k \geq 0$, the string xy^kz is also in L.

Intuitively what this says is that we can always find a nonempty string y not far from the beginning of the string $w \in L$ that can be "pumped": by repeating y any number of times (including zero times, i.e. deleting it) you obtain a string which is still in the language.

The pumping lemma tells us that all regular languages must possess the pumping property. Thus we can show that a language is not regular by showing that it violates the pumping lemma. However, just because a language satisfies the pumping lemma, this does not mean the language is regular. In other words there are non-regular languages which obey the pumping lemma. Thus we should be careful and not use the pumping lemma to prove that a language is regular! We should only use it to prove that some language is not regular. Later we will give a condition which is both necessary and sufficient for a language to be regular.

Now lets prove the pumping lemma. This proof will hopefully also give us intuition about why the pumping lemma holds.

Proof: (Pumping Lemma for regular languages) Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA which recognizes the language L. Let n be number of states in M: i.e. $n = |Q|$. Let $w = w_1 w_2 \ldots w_p$ be a string in L where $p \geq n$. The machine, on reading this string, will enter the states $r_1, r_2, \ldots, r_{p+1}$. Among the first $n+1$ states in this sequence, there must be two states which are the same, by the pigeon hole principle. Call the first of these r_j and the second of these r_k. j and l must be less than or equal to $n+1$. Let $x = w_1 \cdots w_{j-1}$, $y = w_j \cdots w_{l-1}$ and $z = w_l \ldots w_p$. The string x takes r_1 to r_j, the string y takes r_j to r_l and the string z takesr_l to r_p. But r_j is equal to r_l! Thus the system must loop between those two states. Therefore xy^iz must be accepted by the machine for $i \geq 0$. Further we have that $y \neq \epsilon$ since $j \neq l$. Finally we have that $|xy| \leq n$. Thus we have proven the pumping lemma.

II. EXAMPLES

A. An Example, Redux

Let's use the pumping lemma first to show that the language above $L_b = \{w|w = 0^k1^k\}$ is not regular. Assume that L_b is regular (for contradiction.) Let n be the pumping length in the pumping lemma and let $w = 0^n1^n$. w is in L_b. The pumping lemma tells us that w can be split up into three pieces $s = xyz$ satisfying the lemmas conditions. Consider the following three cases

1. y is made up entirely of 0s. Then xy^2z will have more 0s than 1s and hence is not in L_b. But the pumping lemma says that xy^kz should be in L, so this is a contradiction.
2. Similarly if y is made up entirely of 1s then this leads to a contradiction.
3. Thus the only case left to check is if y contains both 0s and 1s. But then $xy^2z = xyyz$ will have strings which are out of order, i.e. there will be some 1s before some 0s. In other words xy^2z is not in L_b. This is a contradiction of the pumping lemma.

Thus we cannot avoid a contradiction in the three possible cases for y. Notice that, having picked n, any old string of length n or greater will work to provide a contradiction. Further note that we made no use of the restriction that $|xy| \leq n$.

B. Example: Balanced Strings

Suppose that we consider the language $B = \{w|w$ has an equal number of 0s and 1s$\}$ and want to know whether it is regular at not. Suppose we choose the string 0^n1^n and n is the pumping length in the lemma. Then it would seem that we could apply the pumping lemma because if we let x and z be the empty string, then $y^k = (0^n1^n)^k$ is still in B. But wait, this would then violate $|xy| \leq n$. If $|xy| \leq n$, then y must consist only of 0s, so $xyyz$ is not in B. Thus 0^n1^n cannot be pumped.

Another observation is that the choice of the string really matters. If we had choosen, for example $(01)^n$ instead, we would have been in trouble. Why? Because we could set $x = \varepsilon$, $y = (01)$ and $z = (01)^{n-1}$. Then $xy^iz \in B$ for all i, $y \neq \varepsilon$, and $|xy| \leq n$.

C. Example: Factorials

Consider the language $C = \{w|w = a^{k!}, k = 1, 2, \ldots\}$. Lets prove that this is not regular. Assume that this language is regular and that the pumping length is n. Then choose the string $w = a^{n!}$. Then the pumping lemma tells us that we can write this as xyz with the three conditions satisfied. The condition that y is not the empty string and $|xy| \leq n$, implies that y must be a^j where $j \in \{1, 2, \ldots, n\}$. The pumping lemmas says that then xy^2z should be in C. But $xy^2z = a^{n!+j}$. But if we add 1 to n to $n!$ we do not get another number which is a factorial.

D. Example: Pumping Down

Consider the language $D = \{0^i1^j|i > j\}$. Again we will prove that D is not regular by contradiction. Begin by assuming D is regular. Let n be the pumping length for this language. Consider strings of the form $s = 0^{n+1}1^n$ which are in D. Express this in the form xyz satisfying the pumping lemma. Now the condition that $|xy| \leq n$ implies that xy will be made up entirely of zeros. Therefore if we examine something like xy^2z it will have more zeros than the original xyz. Thus xy^2z is still in D. Uhoh. And further powering of y will given even more zeros. But wait! The pumping lemma also works for pumping to the zeroth power: i.e. xz should be in D. But, for $|y| > 0$ it is easy to see that the number of 0s in xz is less than or equal to the number of 1s. Thus we have a contradiction with the pumping lemma. Therefore D is not regular.