CS 4510 Automata and Complexity January 22nd 2024

Lecture 4: The Pumping Lemma

Lecturer: Abrahim Ladha *Scribe(s): Michael Wechsler*

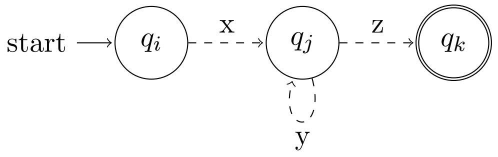
1 The Limitation of DFAs

We previously mentioned that we have some intuition on the limitations of DFAs. Although they seem quite powerful, there are languages which have no DFA to decide them. The goal of today is to prove that.

Consider the language $\{a^n b^n \mid n \in \mathbb{N}\}$. A DFA has a finite amount of states, and is only able to read the string left to right. It cannot do any pre or post processing. It cannot read symbols it has previously. As it reads left to right, it somehow is tasked with memorizing an arbitrarily large amount of information, the number of a's, in order to match them to the number of b's. Note how different this is than $(ab)^*$, or a^*b^*, which can be computed using only a finite number of states. A DFA of say, 20 states may correctly decide if a string has the form $a^{20}b^{20}$, but this DFA must fail on a string of a large enough size, say $a^{10000}b^{10000}$.

2 The Pumping Lemma

Suppose we have a DFA, D, made up of p states. Consider a word, w, such that $|w| = p$. When we simulate D on input w, consider the sequence of states visited when deciding if D accepts or rejects w.



By the Pigeonhole Principle, some state (q_j above) appears twice in this sequence, and our computation path through the DFA must contain a cycle. Note each letter of our word takes not one state, but one transition. If a DFA has three states, then all states could be visited by two transitions, a word of length two, including the start state. If we have p states, and compute on a word of length $\geq p$, then some state is visited twice in our computation path.

- ⋆ Pigeonhole Principle[1]: if you have $p+1$ pigeons and p pigeonholes, there must be at least 1 pigeonhole with greater than 1 pigeons in it

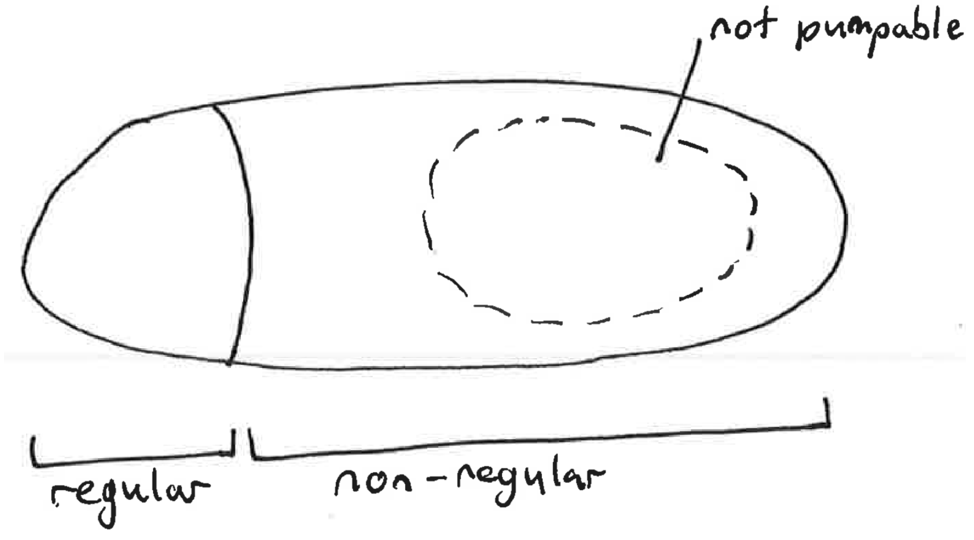
[1]The PP is often mis-stated, even in papers. It says nothing about randomness or distribution. Simply that if you assign m pigeons to n pigeonholes with $m > n$, then there must exist a pigeonhole with more than two pigeons. It doesn't say where the pigeonhole is or how many pigeonholes have more than one pigeon or how many pigeons are in the hole. All m pigeons could be crowded into the one hole even.

We may not know where this loop is or how long it is, but we know that it must exist by the Pigeonhole Principle. Given the fact that a DFA has only finitely many states, but is tasked with computing on arbitrarily long strings, we may choose a string long enough such that the computation path has a repeated state by the pigeonhole principle, and thus a cycle. Then the following claim is true.

$$\text{If } xyz \in L, \text{ then } \forall i \in \mathbb{N}, xy^iz \in L$$

If a string is long enough, we say that it can be pumped. If an infinite language is regular, than it can be pumped. You should think of i here as the number of times you may traverse the loop. Traversing it one time is the original string xyz. You may also traverse it zero times, so $xyz \in L \implies xz \in L$. You may traverse it twice, so $xyz \in L \implies xyyz \in L$, and so on. If a language is regular, than it can be pumped.

If a language is regular, then it can be pumped. Note that we have not found a perfect characterization when a language is regular or not. But we have found a technique to prove that a language is not regular. If a language cannot be pumped, then it is not regular. If a language can be pumped, that does not imply it is regular.



There do exist some (rare) languages which are pumpable but not regular.

Let us derive the exact conditions to apply the pumping lemma. Suppose we want to prove that a language L is not regular. First, suppose that our language is regular, with a DFA to decide it. Then choose some string $s \in L$ with $|s| \geq p$. This is so we may have our pigeonhole principle criterion, and the sequence of states visited during the computation of s contains a repeated state, and thus a repeated cycle, a pumpable substring. We don't know where this substring of y in $s = xyz$ is, so consider all possible cases to break up s into $s = xyz$. We may condition each case of breaking the string up with the following two conditions. First, is that $|xy| \leq p$. This guarantees that our pigeonhole collision occurs at the last state visited during the computation of the prefix xy. Basically, this enforces that y is what we say it is. Second, we may require that $|y| > 0$. We would like to force ourselves to pump something non empty. You may always trivially pump the empty string, but this will not help us to reach a contradiction. Then for each case, choose an $i \neq 1$ and show that $xy^iz \notin L$. Since there is no way to pump this string $s \in L$, then we can conclude that L was not pumpable, and thus, must not be regular.

3 Formula

The pumping lemma has many moving pieces and can be tricky to apply. There are existential and universal quantifications through out. Of the proof techniques available to you, it certainly is the most cumbersome. I suggest you use this formula exactly. Suppose that L is the language we want to prove is not regular.

1. Assume to the contrary, L is regular with pumping length p
2. Choose some $s \in L$ such that $|s| \geq p$
3. For all cases $s = xyz$ such that $|xy| \leq p$ and $|y| > 0$
4. Choose $i \neq 1$ and demonstrate that $xy^iz \notin L$
5. Conclude that L cannot be pumped, which means L is not regular

Lets go through the importance of each step.

- First, by assuming to the contrary that L is regular with pumping length p, we are supposing that there exists a DFA of p states. When we reach a contradiction, then no such DFA of p states can exist. Since p is general, this means that no such DFA can exist at all. We cannot fix p. If we did a pumping lemma proof with $p = 3$, this would conclude that there is no DFA of three states. It does not imply there is no DFA at all, as there may exist a DFA with more than three states to decide the language.
- We choose to pump some string in the language. By choosing $s \in L$, we know s brings our assumed DFA to an accept state, like a path in a graph. By requiring $|s| \geq p$, we get to apply the pigeonhole principle, and we know that this computation contains a repeated state. It is not uncommon for us to choose strings with length much much larger than p. The only requirement is that its length is greater than or equal to.
- In the computation of s on our assumed DFA, we are guaranteed that there exists a loop somewhere by chosing $|s| \geq p$, but we don't know where. So we have to consider all possible cases of where this loop could be. We model this as considering all ways to break up s into the three parts $s = xyz$ subject to our two conditions on each case. Firstly that $|xy| \leq p$. This ensures that the occurance of a repeated state occurs somewhere before the end of what we denote as y. The second condition $|y| > 0$ ensures that this cycle is actually occuring. Note that $|y| = 0$ implies $y = \varepsilon$ and $\forall i \varepsilon^i = \varepsilon$, so $xy^iz = xyz \in L$, ensuring we could never reach a contradiction.
- For each case, you only need to choose an i so that $xy^iz \in L$.
- Since we took a long enough string in the language, showed it was impossible to pump, then there cannot exist a DFA to decide L, and we must conclude that L must not be regular.

4 Examples

We apply the five step formula as previously described.

4.1 $L_1 = \{0^n1^n \mid n \in \mathbb{N}\}$

1. Assume to the contrary, L_1 is regular with pumping length p
2. Let $s = 0^p1^p$ and notice that $s \in L_1$ and $|s| = 2p \geq p$
3. There is only 1 case since the first p characters in the string are all 0s
   $x = 0^a,\ y = 0^b,\ z = 0^{p-a-b}1^p$ subject to $|xy| = a + b \leq p$ and $|y| = b > 0$
4. Choose $i = 2$
   $xy^iz = xy^2z = xyyz = 0^a0^b0^b0^{p-a-b}1^p = 0^{p+b}1^p$
5. We know that $b > 0$, so the number of 0s does not equal the number of 1s since $p + b > p$. Thus, L_1 cannot be pumped, and as a result, is not regular.

Lets annotate this proof. The language 0^n1^n is the canonical example of a non-regular language. We choose s as a function of p so that $|s| \geq p$ is obvious. By choosing a good s, we can ensure that we reduce the number of cases required. The number of cases is technically a function of p, the number of ways $a + b \leq p$ subject to those conditions. We group these all into one case as the contradiction is identical. Note that like s, the substrings x, y, z also end up being a function of p. We only need to to show one $i \neq 1$ gives a contradiction, so we choose a smallest and simplest one, that $i = 2$.

4.2 $L_2 = \{ww^R \mid w \in \Sigma^*\}$

Note that by w^R, we denote the reveral of the string w. This language, ww^R then consists of the even length palindromes.

1. Assume to the contrary, L_2 is regular with pumping length p
2. Let $s = 0^{p-1}110^{p-1}$ (We are choosing a poor s on purpose)
   Confirm that $s \in L_2$ and $|s| = 2p \geq p$
3. The first p characters in the string are different, meaning there are several cases

   (a) $x = 0^a,\ y = 0^b,\ z = 0^{p-1-a-b}110^{p-1}$
   Subject to $|xy| = a + b \leq p$ and $|y| = b > 0$

   (b) $x = 0^a,\ y = 0^{p-1-a}1,\ z = 10^{p-1}$
   Subject to $|xy| = a + p - 1 - a + 1 = p \geq p$ and $|y| = p - 1 - a + 1 > 0$
4. Choose i for each case

(a) Choose $i = 2$
$xy^2z = xyyz = 0^a0^b0^b0^{p-1-a-b}110^{p-1} = 0^{p-1+b}110^{p-1}$
Since $b > 0$, we know that $p-1+b \neq p-1$. Therefore, the two sections of 0s are unequal. If we were to split the string in half, The pair of 1s has moved right, past the previous midpoint so that the first half contains no 1s, and the second half contains two 1s, implying that this is not a palindrome.

(b) Choose $i = 0$
$xy^0z = xz = 0^a10^{p-1}$
Since there is only a single 1, this is not an even-length palindrome.

5. For both cases, the language could not be pumped. Therefore, L_2 is not regular.

Lets annotate this proof as well. We chose a poor s on purpose, resulting in more cases. There were two cases, whether or not xy contained a 1 or not. Had we increased the string length so that the initial block of 0's exceeded p, we would only have one case. For case b, we chose $i = 0$. We call this "pumping down". We could have chosen a worse s as $s = 0^p0^p$. Note that this is a simple even length palindrome, but it is too simple. It can be easily pumped. You want a string so that it is barely in the language, at the extremal conditions. Any small peturbation results in it no longer being in the language. Lets do another example with a better chosen s.

4.3 $L_3 = \{\ ww \mid w \in \Sigma^*\}$

This language consists of words which are themselves concatenated twice. It is not $\Sigma^*\Sigma^*$, but it contains strings like $abab, abaaba, aabbaa$ and so on.

1. Assume to the contrary, L_3 is regular with pumping length p

2. Let $s = 0^p10^p1$ and notice that $s \in L_3$ and $|s| = 2p + 2 \geq p$

3. There is only 1 case since the first p characters in the string are all 0s
$x = 0^a,\ y = 0^b,\ z = 0^{p-a-b}10^p1$
Subject to $|xy| = a + b \leq p$ and $|y| = b > 0$

4. Choose $i = 2$
$xy^2z = xyyz = 0^a0^b0^b0^{p-a-b}10^p1 = 0^{p+b}10^p1$
Take the right-most $p + 2$ characters in xy^2z. This string, which we'll call $w_2 = 10^p1$. Now, there are two cases for the leftmost string, $w_1 = 0^{p+b}$.

   (a) If $b = 1$, xy^2z is not even length, and therefore not in L_3

   (b) If $b > 1$, the midpoint of $xy^2z = w_1w_2$ is in the first block of 0s. We can tell that $w_1 \neq w_2$, and therefore, xy^2z is not in L_3

5. Both cases end with the pumped string not being in L_3. Thus, L_3 cannot be pumped and is not regular.

Lets do a unary example.

4.4 $L_5 = \{1^{n^2} \mid n \in \mathbb{N}\}$

1. Assume to the contrary, L_5 is regular with pumping length p
2. Let $s = 1^{p^2}$ and observe that $s \in L_5$ and $|s| = p^2 \geq p$
3. There is only 1 case since the first p characters in the string are all 1s
   $x = 1^a,\ y = 1^b,\ z = 1^{p^2-a-b}$
   Subject to $|xy| = a + b \leq p$ and $|y| = b > 0$
4. Look at $i = 2$
   $xy^2z = xyyz = 1^a1^b1^b1^{p^2-a-b} = 1^{p^2+b}$

   Since $b > 0$, $p^2 + b > p^2$
   Since $a + b \leq p$, $b \leq p$
   Thus $p^2 + b \leq p^2 + p < p^2 + p + (p+1) = p^2 + 2p + 1 = (p+1)^2$

   By the first and third lines, we know $|1^{p^2}| < |1^{p^2+b}| < |1^{(p+1)^2}|$
5. By the last line, we can see that xy^2z falls between two adjacent strings in L_5. Therefore, its length is not some perfect square and is not in L_5. Thus, L_5 cannot be pumped and is not regular.

5 Advanced Examples

Here are some interesting problems which require a slightly more difficult application of the pumping lemma

5.1 $L_6 = \{1^q \mid q$ **is prime**$\}$

1. Assume to the contrary, L_6 is regular with pumping length p
2. Let $s = 1^q$ where q is the next largest prime greater than p. By this definition, $s \in L$ and $|s| = q > p$.
3. There is only 1 case since the first p characters in the string are all 1s
   $x = 1^a,\ y = 1^b,\ z = 1^{q-a-b}$
   Subject to $|xy| = a + b \leq p$ and $|y| = b > 0$
4. Consider at $i = q + 1$
   $xy^{q+1}z = 1^a1^{b(q+1)}1^{q-a-b} = 1^{q+qb}$

   Since $b > 0$, $q + qb = q(1 + b)$
5. Since $q(1 + b)$ is a product of two numbers, it is composite, and not a prime, so we see that $xy^{q+1}z \notin L_6$ and thus, it cannot be regular.

For this example, how did I know to choose $i = q + 1$? I worked it out before hand, solving for i which would lead to a contradiction. Each pumping lemma proof should be done twice. Once to know the structure of the proof, and the second time formally.

5.2 $L = \{0^{a_1}10^{a_2}1...10^{a_k} \mid i \neq j \implies a_i \neq a_j\}$

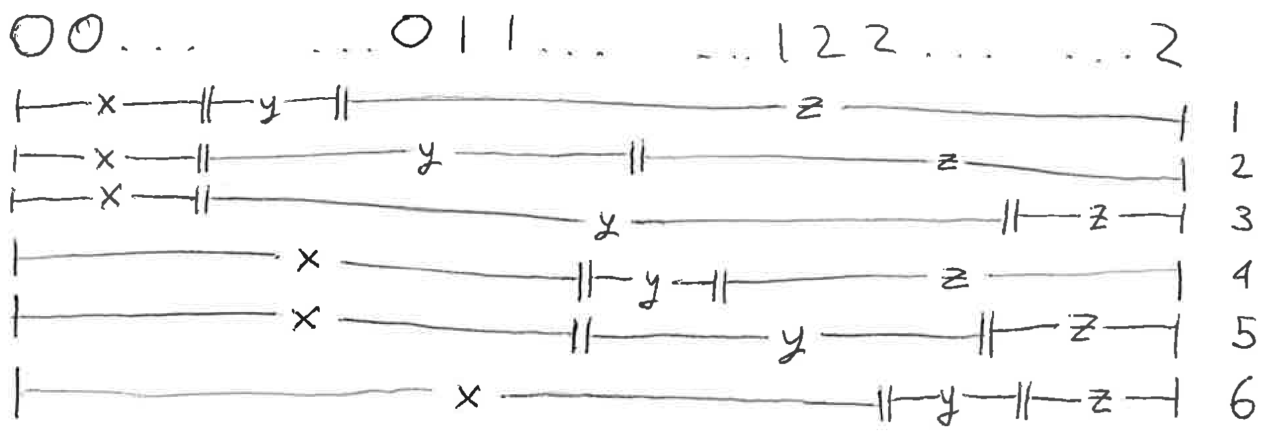
Essentially, this is a language made up of k blocks of zeroes, each delimited by a 1. No two blocks have the same number of zeroes. There can be any number of blocks.

1. Assume to the contrary, L_6 is regular with pumping length p
2. Let $s = 0^p10^{p-1}10^{p-2}1...1011$. Each block has sequentially decreasing length, and there is one block per value from $p, p-1, ...2, 1, 0$. Note that $s \in L$ and $|s| > p$
3. There is only 1 case since the first p characters in the string are all 0s
   $x = 0^a,\ y = 0^b,\ z = 0^{p-a-b}1...11$
   Subject to $|xy| = a + b \leq p$ and $|y| = b > 0$
4. Consider $i = 0$, we will pump down.
   $xy^0z = xz = 0^{p-b}10^{p-1}1....11$

   Since $a + b \leq p$ then $b \leq p$. And since $b > 0$ we know that $0 < p - b < p$
5. So $xz = 0^{p-b}10^{p-1}1....11$ has two duplicate blocks by the pigeonhole principle, and $xz \notin L_6$ so we know L_6 cannot be regular.

6 A note on choosing a bad s

Consider the language $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$. It is not regular for similar reasons to $a^n b^n$. When choosing an s, you want to eliminate the number of possible cases, not just so you have a shorter cleaner proof, but so you will be less likely to make a mistake. A good choice of s is $s = 0^p 1^p 2^p$, there is only one case. What happens if we chose a bad s like $s = 0^{\lfloor p/3 \rfloor + 1} 1^{\lfloor p/3 \rfloor + 1} 2^{\lfloor p/3 \rfloor + 1}$? We would actually end up with six possible messy cases. Note that your proof would be incorrect if you miss enumeration of a case.



By choosing a larger s so that the first block of 0s is length p instead of length $\lfloor p/3 \rfloor + 1$, we can use the condition $|xy| \le p$ to eliminate the cases $2-6$ where y may contain symbols other than 0.