Chapter 10

Pumping Lemma for Regular Languages

So far, we have seen a few different abstractions, from DFAs to regular expressions, that accept/describe exactly the same class of languages, the regular languages. This implies that regular languages represent quite an important class of languages. One might ask if there are any languages that are *not* regular. The intuitive answer is that there have to be, as there are many capabilities that DFAs, as the simplest model we considered, are missing. For one, they don't have any memory, so they cannot remember information in any other way than through a limited number of states. Which means that the information they can remember has to be limited, in terms of, for example, how many different values it can have. Consider, for example, the language L over the alphabet $\{0, 1\}$, where $L = \{w | w$ has the same number of symbols 0 and $1\}$. To accept this language, any DFA would have to accept every string that has the same number of 0's and 1's, regardless of what this number is. To do that, the DFA would need to remember the difference between the number of 0's and the number of 1's seen so far. But, since a DFA does not have memory, it cannot store this difference as, say, a variable that is updated each time 0 or 1 are read. It could only remember this difference by having different states that correspond to the difference between the number of 0's and 1's seen so far. But the number of states has to be finite! Therefore, if the number of states in the DFA is n, it would not be able to remember more than n different values. Therefore, it wouldn't be able to tell the difference between a string where this difference is, for example, n and $n + 1$. For example, it wouldn't be able to tell the difference between the string 0^n1 and $0^{n+1}1$. So, no matter how many states a particular DFA has, there will always be strings of 0's and 1's for which it wouldn't be able to count the difference between the number of 0's and of 1's seen, and will not be able to tell if there are as many 0's as 1's.

So, it seems that we have found a language L that is not regular. But how would we *prove* that L is not regular? Proving that a language *is* regular is relatively easy - we "just" need to find a DFA (or an NFA or a NFA with ϵ-transitions or a regular expression) that accepts that language. But proving that a language is not regular is notably harder. We would somehow need to prove that there does not exist a DFA (or some other model we considered that accepts regular languages) that accepts that language. This is very hard, because we would somehow need to consider all the DFAs and conclude that none of them can accept our language. Intuition, such as in the case above, might not work - for example, the language $L_2 = \{w | w \in \{0, 1\}^*, w$ has the same number of substrings 01 and $10\}$ is regular, even though intuitively we might think that we need to count the number of occurences of substrings 01 and 10, and that therefore, using the same intuition as for our language L, there is no DFA that accepts L_2. Fortunately, there are some relatively easy to use techniques that can help us with proving formally that a language is not regular. We give one of them now, in the form of the famous Pumping Lemma.

Theorem 10.1: Pumping Lemma

Let L be a regular language. Then there exists a positive integer n, such that for any string $w \in L$ such that $|w| \geq n$, there exist strings xyz such that $w = xyz$ and

(i) $|xy| \leq n$;

(ii) $|y| > 0$;

(iii) For any $i > 0$, $xy^iz \in L$.

Before trying to comprehend what the Pumping Lemma states, let's start with some basics. Firstly, we

said that the Pumping Lemma helps us prove that a language is *not* regular. But the Pumping Lemma states some property of regular languages! How can then it be used to prove that a language is not regular? Very easily. If we have a language L and we manage to prove that the property *does not* hold, that has to mean that L is not regular. Otherwise, if it was regular, the property would have held. This is precisely how we use the Pumping Lemma - we show that the property in it does not hold for our language L, and from that conclude that L is not regular. In fact, it is worth noting that we cannot use the Pumping Lemma to prove that some language L is regular. We can show that the language L has the property of the Pumping Lemma, but that still does not mean the language L is regular, as non-regular languages can also have the same property. It is not hard (and we will do this at the end of this section) to find an example of a non-regular language L that still has the property from the Pumping Lemma.

Remark 10.1

If this reasoning still sounds strange to you, consider a simple real-world example. We know that if it rained around our house, our street will be wet. So, we know that if it rained around our house, the property "our street is wet" will hold. But we cannot use the truthness of the property that our street is wet to prove that it rained around our house. If our street is wet, there might have been some other reason for it being wet other than rain, e.g. that it has been washed. However, if our street is not wet, we can definitely conclude that it did not rain. Because if it rained, our street would have been wet, which it isn't. Therefore, it cannot be true that it rained. Exactly the same reasoning is used in applications of the Pumping Lemma.

Now, let us analyse what the Pumping Lemma actually states. The Pumping Lemma states that the property that every regular language L has is that there exists a positive integer n, such that for any string $w \in L$ of length at least n, there exist strings x, y, z such that the following three properties hold: (i) $|xy| \leq n$; (ii) $|y| > 0$; and, (iii) For any $i > 0$, it has to be true that $xy^iz \in L$. Let us call this property the *pumping property*. The pumping property looks quite complicated! Furthermore, we said that we use the Pumping Lemma to prove that some language is not regular, which we do by proving that for that language the pumping property does not hold. What does it mean that the pumping property does not hold? We, essentially, need a negation of this property. To figure out what this negation is, let us break the pumping property into components:

- Firstly, at the top level, the property says that "there exists a positive integer n, such that X holds", where X is the rest of the property. What is the negation of this property? It is "there does not exist a positive integer n, such that X holds". Or, in other words, "for every positive integer n, X does not hold".
- Going further, the property X is "for any string w of length greater or equal to n, Y holds", where Y is, again, some other property. "X does not hold" means "it is not true that for any string w of length greater or equal to n, Y holds". This can be written as "there exists a string w of length greater or equal to n for which Y does not hold".
- The property Y is "there exist strings x, y, z such that $w = xyz$ and (i), (ii) and (iii) hold.". "Y does not hold" is then translated to "for any strings x, y, z such that $w = xyz$, it cannot be true that all the properties (i), (ii) and (iii) also hold". So, to show that Y does not hold, we need to show that for any three strings x, y, z such that $w = xyz$, at least one of the properties (i), (ii) or (iii) does not hold.

To sum our discussion so far up, the negation of the pumping property for some language L is "**for every** positive integer number n, **there exists** a string w of length at least n such that **for any** strings x, y, z such that $w = xyz$, **at least one** of the properties (i) $|xy| \leq n$; (ii) $|y| > 0$; and, (iii) for any $i \geq 0$, $xy^iz \in L$, **does not hold**." If we manage to show this for some language L, that will mean that the pumping property for L does not hold, and therefore L cannot be regular. In practice, we show this by finding, for any n, a string w of length at least n, such that no matter how we break the string w into concatenation xyz, if the properties (i) and (ii) hold, then the property (iii) cannot hold.

Before we present a formal proof of the Pumping Lemma, let us see an example of its application.

Example 10.1: 0^n1^n is not regular

Use the Pumping Lemma to prove that the language $L = \{0^n1^n | n \geq 0\}$ over alphabet $\{0, 1\}$ is not regular

Solution. Let us assume that L is regular. Then the pumping property has to hold for L. We are going to prove that it cannot hold.

We have to prove that there cannot exist a positive integer n such that the property X hold. Let us take an arbitrary positive integer n. Now we need to find a string w of length at least n such that the property Y does not hold. Let us take $w = 0^n1^n$. The length of this string is $2n$, which is certainly at least n. Now we need to prove that however we break this string into a concatenation xyz, if the properties (i) and (ii) hold, then the property (iii) cannot hold. Let us then take arbitrary strings x, y, z such that $w = xyz$. Let us, further, assume that the properties (i) and (ii) hold for x, y, z. The property (i) says that $|xy| \leq n$. Since the first n symbols of w are 0 (because $w = 0^n1^n$), xy is some initial portion of w, and the length of xy is at most n, we certainly know that xy consists only of 0's. This also means that y consists only of 0's. The property (ii) says that $|y| > 0$. So, y is not an empty string, therefore it has to contain at least one 0. Now we need to prove that the property (iii) cannot hold. For this, we need to prove that for some $i \geq 0$, xy^iz cannot belong to the language L. Recall that L is the language of all strings that comprise a sequence of 0's, followed by a sequence of 1's, and where the number of 0's is the same as the number of 1's. Let us, then, take $i = 2$. What is xy^2z? Firstly, we know that xyz is in L, because we have choosen $w = xyz$ to be 0^n1^n. xy^2z is $xyyz$. So, xy^2z has two repetitions of the string y. We have concluded that the string y consists only of 0's, and that it has a length of at least 1. But this means that in the string $xyyz$, we have added some more 0's to the string xyz. Therefore, the number of 0's in $xyyz$ is greater than the number of 0's in xyz. But the number of 1's in $xyyz$ is the same as the number of 1's in xyz, since all the 1's are concentrated in z (remember that $|xy| < n$, and the first n symbols of xyz are 0's)! Therefore, $xyyz$ has more 0's than xyz, but the same number of 1's. This means that the number of 0's and 1's in $xyyz$ cannot be the same (because we know that the number of 0's and 1's in xyz is the same). This further means that xy^2z cannot be in L. We have shown that the property (iii) cannot hold, therefore the pumping property does not hold for L. Therefore, L is not regular. □

Let us recap again the steps we have taken in the above solution, as they are typical of pretty much all of the examples of the use of the Pumping Lemma to prove that some language L is not regular. To prove that the language L is not regular, we

1. Assume that L is regular, and therefore that the pumping property holds.
2. We take an *arbitrary* positive integer n.
3. For such n, we choose *one* suitable string w of length at least n.
4. We take an *arbitrary* decomposition of w into strings x, y, z such that $w = xyz$.
5. We show that if the properties (i) and (ii) hold for strings x, y, z, then the property (iii) cannot hold.
6. This means that for any positive integer n, we can find a string w such that no matter how we write w as a concatenation xyz, the properties (i), (ii) and (iii) cannot hold for x, y and z.
7. From this, we conclude that the pumping property does not hold for L.
8. If L was regular, the pumping property must hold. Therefore, the only possibility is that L is not regular

A very common mistake in the application of the Pumping Lemma is to choose one particular decomposition of w into xyz, and to show that for this particular decompositon, the properties (i), (ii) and (iii) do not all hold. This is obviously wrong, as we need to show that this is true for any decomposition of w, not just one particular decomposition.

10.1 Proof of the Pumping Lemma

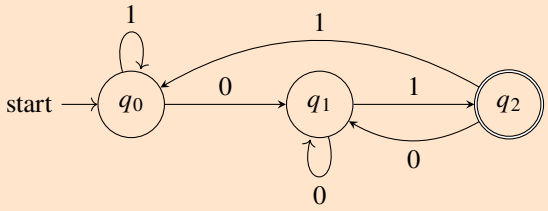
Let us now prove the Pumping Lemma. To prove it, we need to take a regular language L, and then for this language find a positive integer n, such that for every string of length at least n, there exist strings xyz such that the properties (i), (ii) and (iii) hold for x, y, z. Therefore, let's start with a regular language L. If L is regular, then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L. Let n be *the number of states* in L. We will show that this is the n from the pumping property!

Let us now take an *arbitrary* string w from L of length at least n. For the moment, we will assume that such string exists and we will later consider what happens if this is not the case. Let us write w as $a_1a_2a_3 \ldots a_m$, where a_i is a symbol from Σ and m is the number of symbols in m. What happens when M

reads the string w? Let us denote by $p_0, p_1, p_2, \ldots, p_m$ the states M goes through when reading w, starting in the state p_0 (which is, obviously, the start state of M, q_0). p_1 is the state M goes to from p_0 upon reading the symbol a_1, p_2 is the state M goes to from p_1 upon reading the symbol a_2 and so on. p_m is the state to which M goes after reading the last symbol, a_m, of w. Since we assumed that $w \in L$, we know that p_m is an accepting state. Furthermore, let us focus only on the states $p_0, p_1, \ldots, p_n$, which are the states M goes through when it reads the first n symbols of w. In the sequence $(p_0, p_1 \ldots, p_n)$, there are $n+1$ elements. But since the elements of this sequence are states, and there are only n states in M, it has to be the case that some element in this sequence appears twice. So, it has to be the case that $p_i = p_j$, for some p_i and p_j from this sequence.

Remark 10.2: Example

To give an example of this consideration, let us look at the following example of a DFA



and let us see what states does it go through when it reads the input string 00101. This sequence of states is

$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_1 \xrightarrow{1} q_2$$

Looking at the first 3 (which is the number of the states in our DFA) states from this sequence, we get (q_0, q_1, q_1, q_2). We can see that the state q_1 appears at two different positions in the sequence.

So far, we have that $(p_0, p_1, \ldots, p_m)$ is the sequence of states M goes through when reading w, p_0 is q_0, p_m is one of the accepting states, and in the first n elements of this sequence, some state p_i appears twice. So, it is true that $p_i = p_j$ for some indexes i, j of our sequence, and we can assume $i < j$ and $i, j \leq n$. Let us now choose strings x, y, z in the following way:

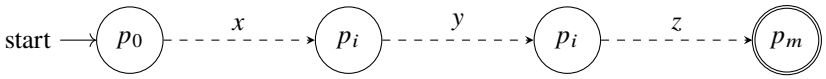
- x is the prefix of w that corresponds to the symbols read from the state $p_0 (= q_0)$ to the *first* appearance of the state p_i in our sequence.
- y is the substring of w that corresponds to the symbols read between the first appearance of the state p_i in our sequence to its second appearance $p_j = p_i$.
- z is the substring of w that corresponds to the symbols read from the second appearance of p_i (that is, from the place j in the sequence) to the end of the string.

Let us now show that the strings x, y, z chosen in this way satisfy the properties (i) $|xy| \leq n$; (ii) $|y| > 0$; and, (iii) $xy^i z \in L$ for every integer $i \geq 0$.

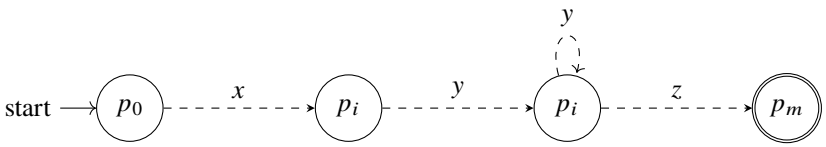
(i) Since x corresponds to the prefix of w read until the state p_i is reached the first time, and y corresponds to the string of subsequent symbols of w read until the second occurence of p_i (the j-th element of ours equence of states). Therefore, xy corresponds to the first j symbols of w and we know that $j \leq n$. Therefore, $|xy| \leq n$.

(ii) y corresponds to the string read between the i-th and $j-$th state in our sequence of states, and we know that $i \neq j$, therefore we know that at least one symbol is read between these two elements in our sequence. Therefore, $|y| > 0$.

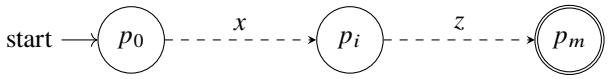
(iii) Finally, let us note that for string $w = xyz$, reading x takes us from the state $p_0 (= q_0)$ to the state p_i, reading y takes us from p_i back to the state $p_i (= p_j)$ and reading z takes us from p_i to p_m, which is an accepting state



Let us now consider the string $w_w = xyyz = xy^2z$. Reading the initial portion x takes us from $p_0 (= q_0)$ to p_i, reading the first substring y takes us from p_i back to p_i, reading the second instance of y again takes us from p_i back to p_i and finally reading z takes us from p_i to some accepting state p_m. Therefore, the string w_2 is also accepted by M!



It is easy to see that the same holds no matter how many times we repeat y in the middle. So, xy^iz is accepted for any $i > 0$. Let us now see what happens when the string $w_3 = xz = xy^0z$ is read. Reading x takes M from p_0 to p_i and reading z takes it from p_i to an accepting state p_m.

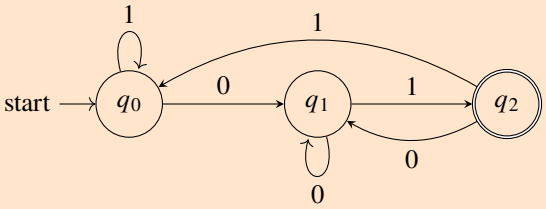


So, xz is also accepted. So it is, indeed, true that xy^iz is accepted by M for every integer $i \geq 0$.

It is worth noting that the string x in our above discussion can be empty, as can z. The only certain fact is that y is non-empty.

Remark 10.3: Example Continued

Coming back to our example DFA



we can see that our n here is 3 (the number of states in the DFA). As an example string w, let us again take 00101.

$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_1 \xrightarrow{1} q_2$$

Then our x is 0, y is 0 and z is 101.

- $w = xyz = 00101$
- $xy^4z = 000000101$ is also accepted by M.
- $xz = 0101$ is also accepted by M.

Let us take a step back and see what we have actually done. For a regular language L, we first considered some DFA M that accepts it. Then we have choosen n to be the number of states of M. Then we have shown that for every string w of length at least n, we can find strings x, y, z, such that $w = xyz$ and (i) $|xy| < n$; (ii) $|y| > 0$; and, (iii) $xy^iz \in L$ for every integer $i \geq 0$. Therefore, we are almost done, as this is exactly the pumping property! The only thing remaining is to consider what happens if there are no strings of length at least n that are accepted by M, because our choice of x, y, z depended on the string w being of length

at least n. But if there are no strings of length at least n in L, then the pumping property trivially holds. This also means that the language L is finite, because over any alphabet there is only finitely many words of each length (because an alphabet has to be finite). Every finite language is regular, as a finite union of one-string languages. Therefore, the pumping property indeed holds for all regular languages L and the Pumping Lemma is proven.

Concluding Remarks

Let us finish the discussion about the Pumping Lemma with two important remarks. We said that the pumping property of the Pumping Lemma holds for all regular languages. But we also said that the fact that some language has the pumping property doesn't necessarily mean that it is regular. There exist languages that are not regular, but which have the pumping propety. Let us consider a following example.

Example 10.2: Non-Regular Language With the Pumping Property

Let

$$L = \{01^n2^n | n \geq 1\} \cup \{0^n1^m2^l | n \neq 2, m \geq 0, l \geq 0\}.$$

Show that the pumping property holds for L.

Solution. Let us take, for example, $n = 5$ and let us consider a string w of length at least 5. There are two possibilities for such string: (a) it starts with one 0 and then contains a block of 1's followed by a block of 2's, where these two blocks are of the same length; or, (b) it starts with no 0's or with two or more 0's, which is followed by a block of 1's, followed by a block of 2's, and these blocks do not need to be of the same length. In the case (a), we can write w as xyz, where $x = \epsilon$, $y = 0$ and $w = 1^n2^n$ for some n. It is obviously true that $|xy| < 5$, $|y| > 0$. $xy^0z = xz = \epsilon 1^n2^n$ belongs to L, as does $xy^iz = \epsilon 0^i1^n2^n$ for any $i > 0$. Therefore, in this case $xy^iz \in L$ for every $i \geq 0$. In the case that w is of the form (b), if it starts with two 0's, it is easy to see that writing w as xyz, where $x = \epsilon$, $y = 00$ and z being the rest of w works. Finally, if w starts with no 0's, then writing it as xyz, where $x = \epsilon$, y is the first symbol of w and z is the rest of w also works. Therefore, whatever form w has, it can be written as xyz such that x, y, z satisfy the conditions (i), (ii) and (iii) of the pumping property.

However, L is not regular. If L was regular, then the language $\{01^n2^n | n \geq 1\}$ would also be regular, as it is an intersection between languages L and $\{0^n1^m2^l | n \neq 2, m \geq 0, l \geq 0\}$, both of which are regular, and we have seen that the intersection of two regular languages is also a regular language. But 01^n2^n is not regular (this can also be easily proven using the Pumping Lemma). Therefore L cannot be regular either. □

The second observation is that, as we have noted before, to prove that the pumping property holds, we need just to find *one* n, such that for *any* string w of length at least n, there exist *one* set of strings x, y, z, such that $w = xyz$ and the conditions (i) – (iii) of the pumping property hold. However, to prove that the pumping property does not hold, when considering some string w, we need to show that *no* strings x, y, z such that $w = xyz$ can satisfy the conditions (i) – (iii). It is not enough to show this for just one set of strings x, y, z such that $w = xyz$. Consider the following wrong application of the Pumping Lemma:

Example 10.3: Wrong Application of the Pumping Lemma

"Prove" that $L = \{w | w$ contains substring $101\}$ is not regular.

Non-Solution. "Prove" in the example is in quotation marks because we cannot prove this, since L is a regular language. But consider the following "proof". Let us assume that L is regular and let us take some arbitrary n. Let us then take $w = 101^n$. $|w| = n + 2$, so w is definitely of length at least n. Let us write w as xyz, where $x = 1, y = 0$ and $z = 1^n$. $|xy| \leq n$ and $|y| > 0$. But $xy^2z = 1001^n$ does not belong to L. Therefore, L does not have the pumping property, and is not regular. □

The error here is, obviously, that we have considered just one particular way of writing w as xyz, and not all of them. There are many others that satisfy the conditions (i) – (iii). For example, $x = 10$, $y = 1$ and z being the rest of w satisfy the conditions (i)–(iii) for any $n \geq 3$.