CS 4510 Automata and Complexity	January 31
Lecture 6	
Lecturer: Frederic Faulkner	*Scribe(s):*

1 Non-Regular Languages

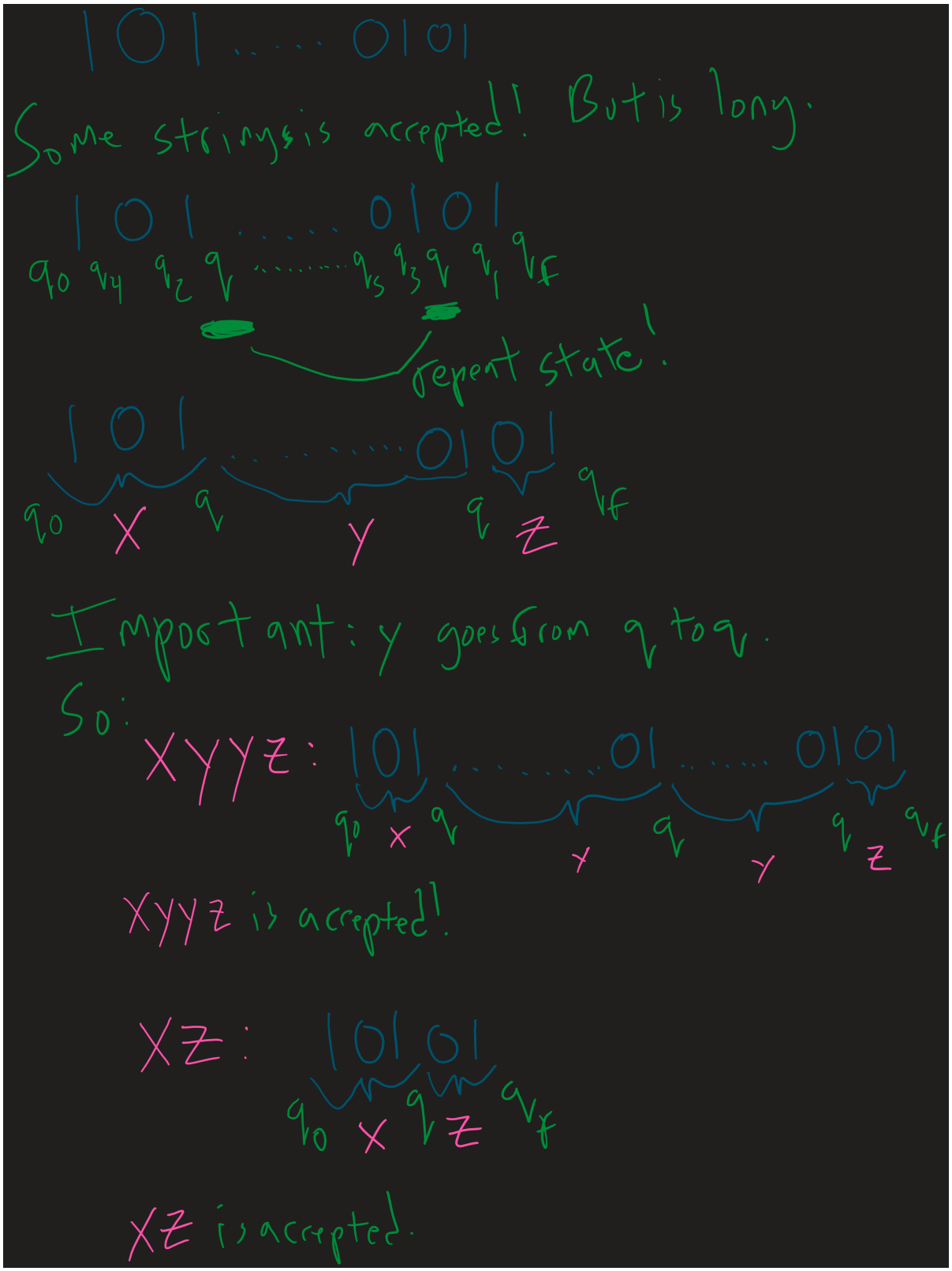
So far in class we have been discussing different methods for recognizing and handling regular languages. You would be forgiven for thinking that perhaps all languages are regular, since we have never worked with any other kind of language. However, there do exist non-regular languages, or languages that somehow cannot be captured by *any* DFA.

Consider the language $\{0^n1^n | n \in \mathbb{N}\}$, i.e. strings like 01, 0011, 000111, etc. If you were to write a program for this language, could you make one that uses only finite memory? Intuitively it feels like you cannot, since you need to keep track of a potentially very large number of 0's so you can compare them to the 1's. This is just an intuition, not a proof, but 0^n1^n is indeed not regular! We shall now discuss how we might prove that fact.

1.1 A Useful Property

The key idea here is that, since DFA's have a finite number states, sufficiently long input strings cause the DFA to repeat states. For example, if a string of length 7 is given to a DFA with, say, 4 states, it is not possible for the DFA to visit a new state for each character. Eventually, as the DFA processes the input, it must return to a state that it has already visited so far this computation. In other words, for some state q, there is some sequence of characters that takes the machine from q to q, i.e. there is a loop in the machine.

Suppose some *accepted* string causes the machine to loop by repeating some state q. Then we could break that string into three parts: the part before the loop, the part read during the loop, and the part after the loop that ends in an accepting state. Call those parts x, y and z. x takes the machine from the start to q, y takes the machine from q to q, and z takes the machine from q to an accepting state. xyz is just the original string, which we know is accepted. HOWEVER! We also know that xz is accepted! Because x ends in state q, and z goes from q to an accepting state. Similarly $xyyz$ is accepted, because the machine goes from the start state to q (by reading x), then from q to q (by reading y), then from q to q (by reading y), and finally from q to some accepting state (then by reading z).


101....0101
Some string is accepted! But is long.
101....0101
q_0 q_4 q_2 q ..... q_5 q_3 q q_1 q_f
repeat state!
101.....0101
q_0 X q Y q Z q_f
Important: y goes from q to q.
So:
XYYZ: 101.....01.....0101
q_0 X q Y q Y q Z q_f
XYYZ is accepted!
XZ: 1001
q_0 X q Z q_f
XZ is accepted.

In other words, for strings that are long enough, we can generate new strings in the language by repeating some section of them over and over again (or removing it). This turns out to be a fundamental property of regular languages, which we can use to identify non-regular languages.

For example, consider 0^n1^n again. If we pick some very long string in this language, say $0^{1000}1^{1000}$, not only does this string probably cause a repeated state in the machine, but there are so many 0's that the machine repeats a state while still reading 0's before it even sees the 1's. So what? Well, this means that there is some number of zeroes that we can remove or add to create new strings in the language.

But that doesn't make sense! If we remove some of the zeroes, we will no longer have the same number of 0's and 1's. Similarly for adding 0's. So it would seem that 0^n1^n is not a regular language. We now formalize this train of reasoning.

1.2 The Pumping Lemma

Theorem 1 *Let L be a regular language. Then all sufficiently long strings in the language can be "pumped", i.e. there is some section of the string that can be repeated or removed to generate new strings of L.*

Formally, if L is a regular language, then there is some positive integer p called the "pumping length" such that all long strings $s \in L$ (i.e. s is longer than p), we have that $s = xyz$, i.e. s can be split into three parts such that

- $|y| > 0$ *i.e. y is not empty*
- $|xy| \leq p$
- $xy^iz \in L$ *for $i \in \mathbb{Z}^{\geq 0}$.*

Proof. Let L be an arbitrary regular language with DFA D. We would like to find some number p to be the pumping length. Well, which p should we choose? We want to find a number such that all strings that are that long are "long enough" to be pumped, i.e. long enough to create a loop in the machine D. So let's set p equal to the number of states in D.

Clearly, if a string $s \in L$ is longer than p, it will cause D to repeat a state. Importantly, there will be a repeated state q somewhere in the first p characters, because there are only p states. So, like before, we can split s into three parts x, y, and z, where x is before the first occurrence of q, y takes D from q to q, and z takes D from q to a final state. Then it is clear that x, y, and z meet the conditions of the theorem:

- y is not the empty string, because in order to repeat a state, we must have read at least one character to transition
- $|xy| \leq p$ because p is the number of states in the machine, so once we see p characters we must have repeated a state
- $xy^iz \in L$ for all positive integers i. This is just a fancy way of saying that xz, xyz, $xyyz$, $xyyyz$, etc. are all strings of L.

1.3 Using the Pumping Lemma

Now that we have found this property that all regular languages possess, we can show that languages are not regular by showing that they do not possess this property. (CAUTION! If a language does not satisfy the pumping lemma, it is not regular. However, the converse does not hold: if a language does satisfy the pumping lemma, it is not necessarily the case that it is regular.)

The pumping lemma says that for each regular language there is some length p such that all long strings (length $\geq p$) can be pumped. Therefore to show that a language is not regular, we must show that there is no pumping length for that language, i.e. that for any p we pick, there is a string longer than p that CANNOT be pumped.

Consider $L_1 = \{0^n 1^n \mid n \in \mathbb{Z}^{\geq 0}\}$. As a counterexample, suppose that some p is in fact the pumping length of L_1. Then let $s = 0^p 1^p$. Since this string is longer than p, according to the pumping lemma we can break $0^p 1^p$ into three parts x,y and z. But note that $|xy| \leq p$ (condition 2) i.e. x and y must both come from the first p characters of s. But the first p characters of s are all $0'$. So in particular, y is all $0'$. So we can write $y = 0^k$. By condition 1, $k \neq 0$.

But according to the pumping lemma, all the strings xz, xyz, $xyyz$, $xyyyz$ should be elements of L_1. So if *any* single one of them is not, we will have found our counterexample.

Consider $xyyz$. This adds k 0's to the original string. So $xyyz = 0^{p+k}1^p$. But this is not in L_1 because it has an unequal number of 0's and 1's! So we have found a contradiction.

Let's recap what just happened here:

- As a contradiciton, we assumed that L_1 was regular. Then, according to the pumping lemma, it has some pumping length p.
- We picked a single, specific string s from L_1 that was longer than p.
- Since s was longer than p, we know from the pumping lemma that we can split it into x, y, and z (where x and y come from the beginning of the string, and y is not empty).
- It should have been the case that all of xz, $xyyz$, $xyyz$, etc. were all members of L_1. But we found one that was not, namely $xyyz$. This is a contradiction! So our original assumption is wrong. But we assumed that L was regular. So L is non-regular.

Notice that this proof does not mention loops or states or DFA's. We used those to prove the pumping lemma; but now that we have the pummping lemma, we can leave those things behind and just use the pumping lemma directly.

We can do another. Let $L_2 = \{1^{x^2} | x \in \mathbb{Z}^+\}$, containing 1, 1111, etc. Our proof proceeds as follows:

- As a contradiction, assume L_2 is regular. Thus it must have some pumping length p.
- We need a string $s \in L$ that is longer than (or equal to) p. Let's choose 1^{p^2}.
- Since s was longer than p, we know from the pumping lemma that we can split it into x, y, and z (where x and y come from the beginning of the string, and y is not empty). Note that because s is all 1's. y is all 1's as well. In particular, $y = 1^k$ for

some number k between 1 and p. If we split s properly, we should get that xz, $xyyz$, $xyyyz$, etc. are all strings of L_2.

- HOWEVER! We consider here $xyyz = 1^{p^2+k}$. Is this string in L_2, i.e. is $p^2 + k$ a perfect square? It is not. The next biggest integer square after p^2 is $(p+1)^2$. There are no squares in between. But $p^2 + k$ is greater than p^2 but smaller than $(p+1)^2$. (Proof: $(p+1)^2 = p^2 + 2p + 1 > p^2 + 2p > p^2 + p \geq p^2 + k$ because we said in the previous step that $k \leq p$.)

- So $p^2 + k$ is not a perfect square, and $xyyz = 1^{p^2+k}$ is not in L. But this contradicts the fact that xz, $xyyz$, $xyyyz$ should all be members of L. So our assumption that L_2 was regular fails. It must be the case that L_2 is not regular.

One last one. Let $L_3 = \{0^m 1^n \mid m > n\}$. We wish to show that L_3 is not regular.

- As a contradiction, we assume that L_3 is regular. Then it must have some pumping length p.

- So we need a string $s \in L$ which is at least as long as p. Let's pick $s = 0^{p+1}1^p$.

- According to the pumping lemma, we can split s into xyz such that $|xy| \leq p$ and $|y| > 0$. Note that this means that y is all 0's, i.e. can write $y = 0^k$ for some $1 \leq k \leq p$. If we split properly, we should get that xz, $xyyz$, $xyyyz$, etc. are all members of L_3.

- However, we consider xz. $xz = 0^{p+1-k}1^p$. But this string is not in L_3, since $p+1-k \leq p$.

- So our assumption that L_3 was regular was flawed. L_3 is therefore not regular.

Caution 1 *When doing a proof using the pumping lemma, you must show that all positive integers p are not the pumping length. This means that, in order for a string to be "longer than p", you must choose a "general" string s that involves p in some way. In other words, a word like* 0^p1^p *is clearly longer than p, regardless of the value of p. However, a string like* 000111 *is not a valid choice because it is not longer than every possible p.*

Caution 2 *One of the things that students struggle with the most is the idea of splitting up the string into x, y, and z. In particular, you must show that it is not possible IN ANY WAY to split up the string s so that it can be "pumped". This means that just picking a single example for x, y, and z is NOT a valid proof. For example, take* L_3*. All we could say about y was that it consisted of all* 0*'s. (We know this because the Pumping Lemma requires that* $|xy| \leq p$*.) However, we cannot say that y consists of exactly 000, for example, because then we will be disproving one particular case of x, y, and z, but not all of them.*