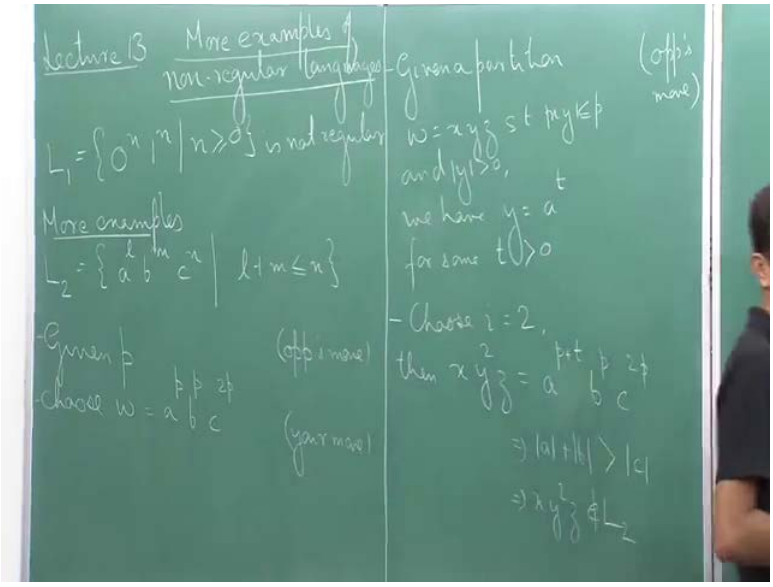
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 13
Examples of non-regular languages

Welcome to the 13th lecture of the course. So, in the last lecture, we showed that the following language 0 to the power n, l to the power n where n is greater than or equal to 0 is not regular using the pumping lemma.

(Refer Slide Time: 00:23)



So, what we planned to do today is look at some more examples of non-regular languages and then give a proof of the pumping lemma, why is the pumping lemma is correct. The next example that we will see is the language L 2, let us call it L 2. Let us take a language over the alphabet a b c. So, we will look at all strings of the form a to the power l, b to the power m and c to the power n such that let say l plus n is less than or equal to n. Let us that l, m, n are non-negative integers and l plus n that is a number of b(s) plus the number of a(s) is at most the number of c(s). So, the claim is that these is also a non-regular language, and let us try to prove this using pumping lemma.

So, once again as we saw last time, so we will treat the pumping lemma. So, we will look at the contra positive form of the pumping lemma and we will treat at as a game between you and an opponent. So, in the first step, your opponent gives you a number, so given p.

So, this is your opponents move. So, you choose a string w that belongs to the language who which has length more than p. So, what string do we choose that belongs to the language. So, in this case, what I will choose is I will choose the string w a to the power p b to the power p and c to the power 2 p.

Now let us verify clearly this string has length more than l. And if you look at the number of a(s) plus the number of b(s), it is equal to the number of c(s). So, hence it belongs to the language L 2. So, this is your move. If it is say your opponent move, it does not given to you; you do not have any control; if it is your move, you can choose any string that you want that satisfies the constraint.

Now, again your opponent will make a move what move does your opponent make. So, your opponent partitions the string into three strings x, y, z such that x, y has length at most p and y has length greater than 0. If x, y has length at most p, what can the string x y contain. So, observe that the way we chose our string w, if x y has length at most p then it consists only of a(s), it cannot have b or c because the number of a(s) is p. So, given a partition w equal to x y z such that x y is at most p and y is greater than 0. We have in particular the string y consists only of a.

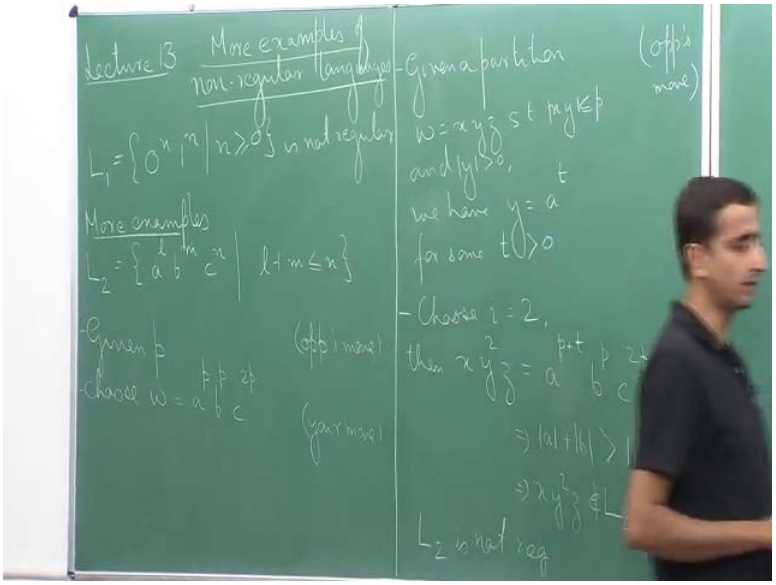
So, y is equal to sum a raise to the power t for some t, strictly greater than 0 because the length of y is greater than 0; and it consists only of a(s) because of what I just said that the length of x and y is at most p. So, this is again your opponents move. Now, what I do you choose now you have to choose i, such that x y to the power i z does not belong to the language. So, in this case, we will choose i equals 2. Let us see what happens if we choose i equal 2. If we consider the string x y square z, how does it look like? So, x y square z is basically nothing but I have x y I mean basically, it has an extra y in between. So, it has an extra number of, it has t extra number of a(s).

So, basically the string will be a to the power p plus t the number of b(s) and c(s) are the same b to the power p c to the power 2 p. So, this implies that number of a is plus the number of b(s) is strictly greater than the number of c(s), which implies that x y square z is not in the language. So, this happened because we have t that is strictly greater than 0. Now in this case we choose i equal to 2, but what would have happen if we chose i equal to 0. So, observe that if we chose i equal to 0 here, then x y to the power 0 z would have

being the string a to the power p minus t, because it does not have that many number of a(s) b to the power p c to the power 2 p.

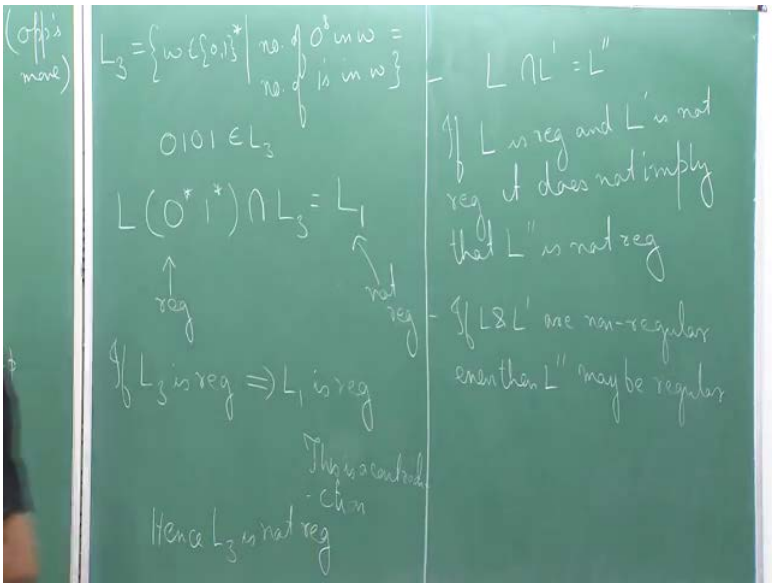
So, number of a is plus the number of b(s) would be p minus t plus p which is less than 2 p which is less than the number of c(s) which would not have put the string x y to the power 0 outside L 2, and that would not have worked. So, basically what I am trying to say is that i equal to 2 works, but i equal to 0 does not work in this case. So, this is one difference between what we choose i this time, and what we had for our language L 1 in last time.

(Refer Slide Time: 08:45)



So and this proves that L 2 is not regular.

(Refer Slide Time: 08:59)



Now let us look at another example, our next example. Let us look at the language L 3, which consists of strings in 0 1 star; such that the number of 0s in w is equal to the number of 1s in w. So, it has equal number of 0s and 1s, but the crucial I mean the difference between this and the L 1 is that in L 1 all the 0s were contiguous and all the 1s were contiguous. So, we had a sequence of 0s followed by a sequence of 1s; here that is not the case. Here, we have they can be distributed anywhere, for example, 0 1 0 1 belongs to the language L 3.

So, how do we prove that L 3 is not regular? So, actually here instead of using the pumping lemma what we will do is, we will use closure properties, we will use closure properties of regular languages, and show that L 3 cannot be regular. So, first consider the language of the regular expression 0 star 1 star. So, 0 star 1 star is a regular expression that corresponds to the language, where we have a sequence of 0s followed by a sequence of 1s; and they can be any number of 0s and any number of 1s, it is not necessary that the number of 0s are equal to the number of 1s. So, and you can very easily construct automaton for this language also if you want.

But anyway, we consider the language of this regular expression L of 0 star 1 star, and we take it is intersection with our language L 3 that is strings which have an equal number of 0s and 1s. What is the answer and what language do we get on the right and side, so if it will take all strings which have a block of 0s followed by a block of 1s. And

it has equal number of 0s and equal number of 1s; it is nothing but our old language 0 to the power n and 1 to the power n equal number of 0 then equal number of ones in a contiguous setting. So, this is nothing but the language L 1. So, we know that this guy is regular, because we have a regular expression for it.

If L 3 is regular what does that imply? If L 3 is regular then by the closure property of intersection, so regular languages are closed under intersection, so this guy is regular this guy is regular, so intersection of these two would also be regular. But that is not the case, if L 3 is regular then it implies that L 1 is regular, but we have already seen earlier that L 1 is not regular.

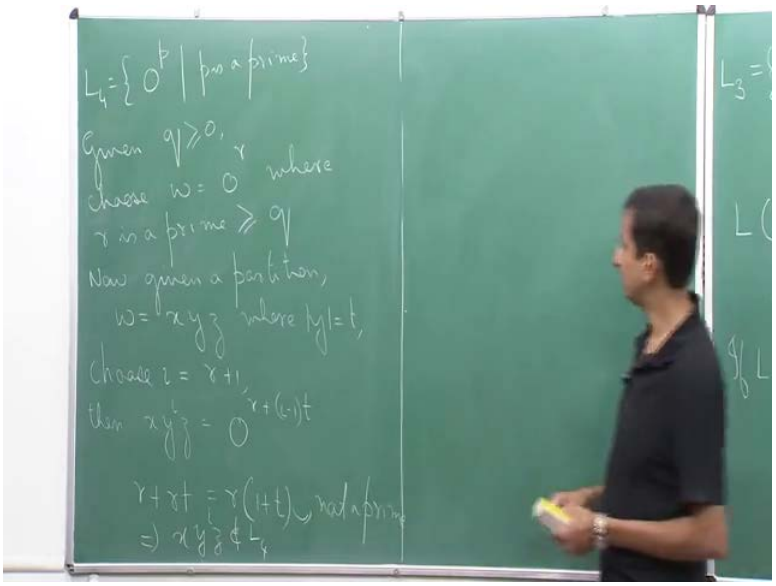
So, this is a contradiction. So, this is a contradiction which means that L 3 is also not regular. So, here I want to make point. So, here to prove that a language is not regular or if we are using is that we are taking another regular language. We are looking at the operation between our given language, our candidate language, and this regular language. And we are showing that there is some other non-regular language. So, this guy is he know is not regular, and this allows us to prove that L 3 is not regular.

But here a common mistake that beginners tend to make is that is the following that. So, suppose instead of this if we had something like let say some let us call it L - some language L intersection with some language L prime gives let say n L language L double prime. If L is regular and L prime is not regular, it does not imply that L double prime is not regular. So, this is the fallacious argument, so this is not correct. If we want to show if we have a language which is regular, and we take another language which is not regular, their intersection might very well be regular also. Even more if L and L prime are not regular, even then L double prime maybe regular. Even if you have two non-regular languages, you can take their intersection and get a language, which is very well regular.

I mean a simple example here is, for example, if you take the language if you take any non-regular language as L and you take the compliment of that language as L prime. If language is non-regular, it is compliment is also non-regular because of this same, because of this a reason. And the intersection between a language and it is compliment is always an empty set which is a regular language, hence we get regular language by

taking intersection of two non-regular languages. So, this is key point which should be kept in mind.

(Refer Slide Time: 16:51)



Now let us look at our last example of a non-regular language today. So, we have a unary language this time, language which is over a single alphabet. So, it consists of all the strings of the form 0 to the power p, where p is a prime. So, what is a prime number a prime number is a number which is divisible only by itself and 1. So, how do we prove that this is not regular? So, once again given in this case let say given well I cannot use p. Let me use another symbol. So, given q greater than or equal to 0, choose w equal to 0 to the power r where r is a prime greater than or equal to q. So, pick any prime number that is more than greater than or equal to q as this. So, by definition w belongs to the language and it has length at least q.

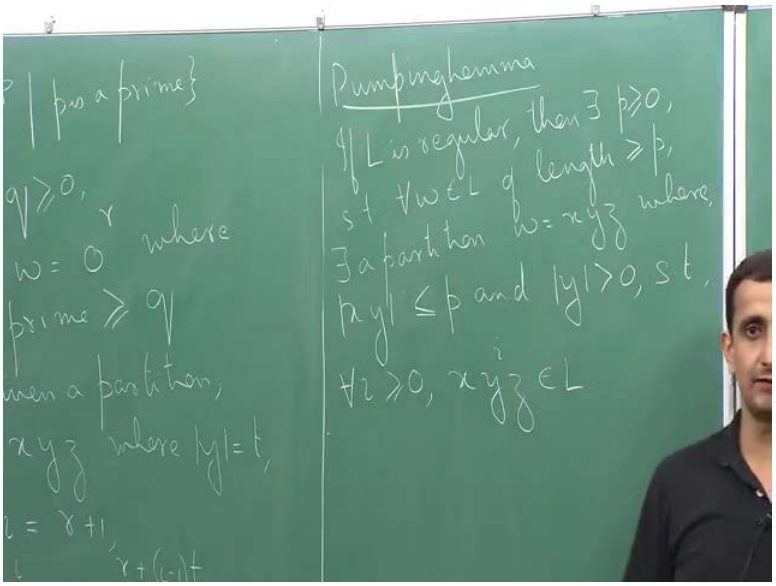
Now, given a partition of w as x y z, where let say that the length of y is sum t. We will choose, so what do we choose i as, so we want to choose i in such a manner, so that x y to the power i z, the length of x y to the power i z will not be a prime, so that is what we want to choose i as. Let us take in this case i as r plus 1. So, we choose i equal to r plus 1, where r was the length of w was the initial length of the string.

So, what do we get? So, then x y to the power i z is nothing but so what is x y to the power i z, so I can write this as, so I have 0 to the power r over here, so x y z is my string w which is 0 to the power r. And then I have another i minus 1 many y(s); this will be 0

to the power r plus i minus 1 which is r plus 1 minus 1. Let me write this as i minus 1 times that many y(s), so i minus 1 times t.

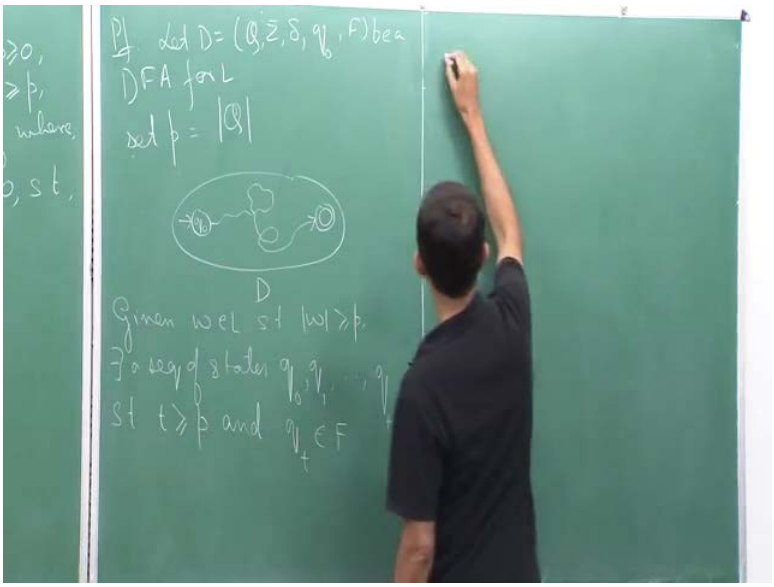
So, what is this number? If i just write this number. So, this is r plus. So, i minus 1 is nothing, but r plus 1 minus 1 which is r t. So, this is equal to r times 1 plus t. And now what are r and 1 plus t. So r is a prime number that is that has length more than q, which means that r has length at least 2, 2 is the first prime. And t has length; t has value that is at least equal to 1 because t is the length of y, which is strictly greater than 0 which means it at least 1. It means that 1 plus t is at least 2. So, what we get is a composite number. So, this is not a prime which implies that x y to the power i z is not in L 4, so that completes the example.

(Refer Slide Time: 22:31)



Now as I said we will prove give a proof of the pumping lemma and that is going to be the last thing that we will discuss today. So, recall the statement of the pumping lemma let me write it down. So, the pumping lemma stated that if L is regular, then there exist a number p greater than or equal to 0, such that for all w belonging to L of length at least p, there exists a partition w equals x y z, where x y is less than or equal to p. And y is greater than 0, such that for all i greater than or equal to 0, x y to the power i z belongs to L. Let us prove this lemma. If L is a regular language, what can we say about L? If L is a regular language, then basically we by definition there exist DFA for L.

(Refer Slide Time: 24:13)

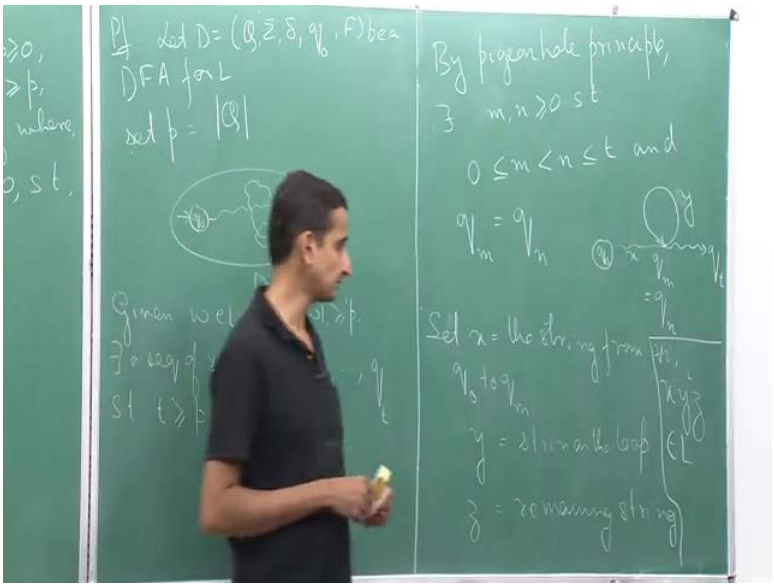


Let this is our proof. Let D equal to Q, sigma, delta, q naught F be a DFA for L. Now what can we say about the DFA. Let us try to understand. So, first we need to pick, we need to show that there exists p. If L is regular, there exists p. So, we set p as the number of states in the DFA. So, in this DFA, we will set p to be the number of states. Let us look at it in a pictorial manner first.

Let say that this is my DFA, so here is my start state q naught. If I take a string w, that has that belongs to the language L which means that it is accepted by the DFA, and it has length greater than or equal to p what does it mean. So, what it means is that there is a path from or not path there is a walk from the state q 0 to some accept state of the DFA on given the language on the given the string w. So, this is my DFA D. There exists a path or there exists a walk it starts at q 0 and it ends at an accept state of the DFA.

So, given w in L, such that length of w is greater than or equal to p there exists a sequence of states q 0, q 1 up to q let say t, such that t is greater than or equal to p, and q t is an accept state. So, there exists a sequence of states that the DFA takes and such that the last state is an accept state; and the number of states is 0 to t, which means t plus 1. Now by pigeon hole principle what can we say. So, how many states does the DFA has, the DFA has p states; and now we have a number t that is greater than or equal to p. So, therefore, in this sequence, actually in this sequence, there is t plus 1 state because i start at 0 and t is greater than or equal to p.

(Refer Slide Time: 27:56)



Which means that by pigeonhole principle there exists m comma n greater than or equal to 0, such that 0 less than or equal to m strictly less than n less than or equal to t. So, basically there exists and q of m equals q of n. So, in this sequence, there exist two states which are the same, two distinct states, so two states which are not in the same position in the sequence. So, they are at different positions, but that states are the same, so there is a reputation. So, basically this is what we have. So, we are going from q 0, there is some point where there is loop; and from this loop, we go to q t. And this is the state q m or q n, whatever you want to call it.

Now what do we have. Now, we set basically x to be this portion of the string this portion from q 0 to q, so set x equal to the string from q 0 to q m, and let m be the first time there is such a reputation which would mean that. So, this is x set y to be this loop, so the string on this loop. So, y is the string on the loop and z is the remaining string. So this means that, so if I do this setting it means that the length of x plus the length of y is the at most p, because the number of states is at most p, and that is the first time I am getting a reputation.

The length of y is strictly greater than 0, because this loop is non-trivial there is some edge on this loop because of this pigeon hole principle that m and n are not equal. And z is the remaining part of the string.

Now basically the remaining part is easy that for all i, if we look at the string x y to the power i z, it just means that after I take x, I repeat y, I number of times. So, every time I repeat y, I will end up at this state; and then if I follow z, I will end up at q t, which means that this belongs to the language L which completes our proof, so that will be all for today.

Thank you.