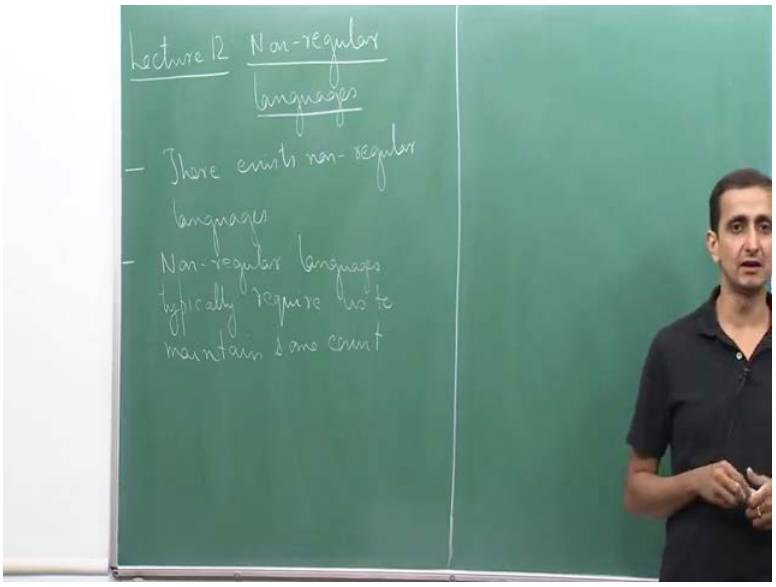
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 12
Non-regular languages and pumping lemma

Welcome everybody. This is the 12th lecture of this course. And today, we will look at some non-regular languages. So far we have been studying regular languages and we saw automaton models and also regular expressions which are capable of accepting these languages.

But the question is that are there other languages for which we cannot construct for example, an automaton, are there languages for which we cannot construct a regular expression. And the answer is yes, and in fact, there are very simple languages, languages which are apparently very simple looking, but we can prove that these languages do not have these languages are not regular which means that these languages do not have any automaton that accepts it.

(Refer Slide Time: 01:15)

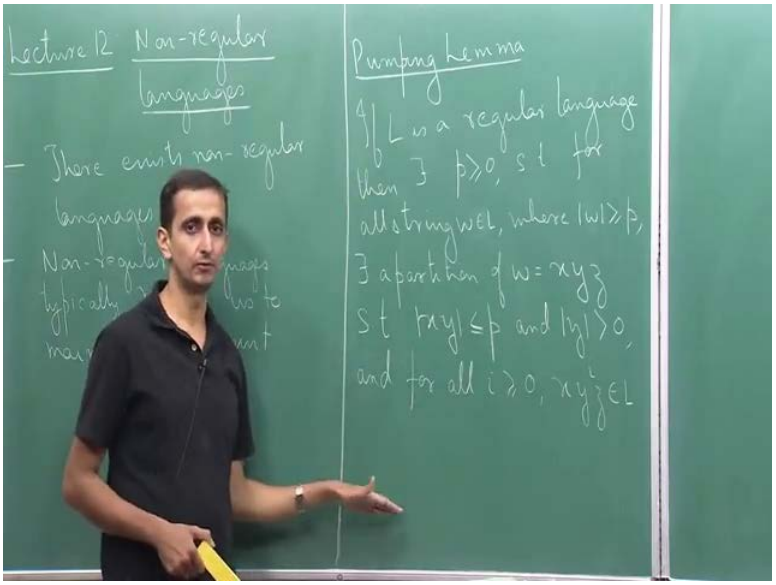


Just to summarize so there exists non-regular languages, and typically what are the I mean what intrinsic properties do these languages have. So, one informal way to think of regular languages is so these are languages, where we do not necessarily require some sort of counting. For languages where to check whether the strings actually belongs to

the language or not, where we actually need to count something. For example, how many 0's are there, how many 1's are there or may be some such thing. These languages typically are non-regular. Non-regular languages are those languages where which requires some sort of counting. So, non-regular languages typically require us to maintain some maintain some count.

So, how do you prove that a language is not regular? To prove this there is a very famous and useful theorem which is called the pumping lemma. The pumping lemma is result which talks of properties of regular languages, and we use this pumping lemma to prove that a language is not regular. So, I am going to state the result of pumping lemma today. And we will see how we can use the pumping lemma to prove that a language is not regular. Next time, we will see proof of correctness of the pumping lemma.

(Refer Slide Time: 03:39)

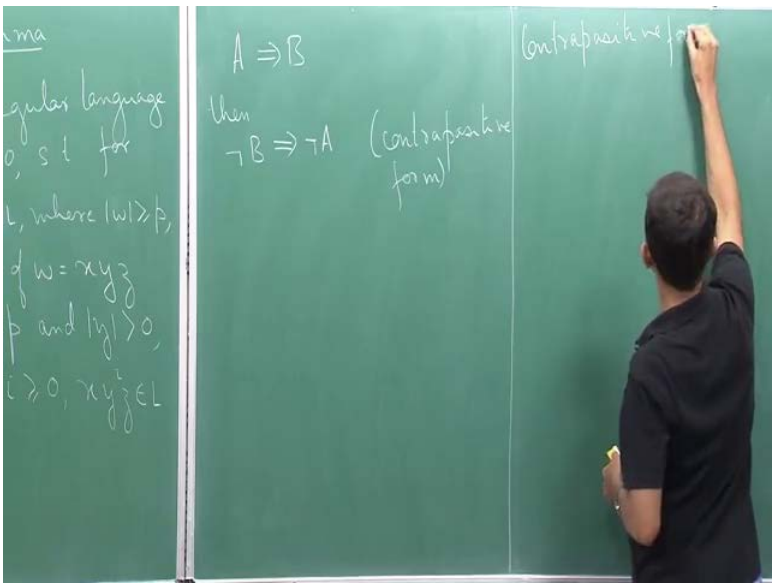


If L is a regular language, then there exists a number p greater than or equal to 0, such that for all strings x, where the length of x is greater than or equal to p, there exists a partition of x equal to. Let me change the notion a little bit, so for all strings instead of x, I will just write w here, this is little more so this is what we use. So, for there exist a partition of w equal to x y z, such that the length of x y is at most p, and the length of y is strictly greater than 0. And for all i greater than or equal to 0, x y to the power i z belongs to the language L. So, for all string we had taken string w which belongs to L. Let me change this also.

Let us try to understand what this lemma is saying. We take a regular language. If we have a regular language, then for this language, there exists some number p such that no matter what string w we take in the language, such that the length of w is at least p. We can partition w into x y z such that the partition satisfies this property that x y is at most p the length of x y is at most p; and the length of y is strictly greater than 0, which means that y is not empty string. And now after we have such a partition, if we consider the string x y to the power i z, for every i greater than or equal to 0, it will always belong to L.

This is basically describing a property of regular language, for every regular language this holds true. So, how can we use this to show that some language is not regular, because this is actually giving a set of properties for or giving a property for a regular language L. The way we use pumping lemma to show that a language is not regular is by using the contra positive form of the statement.

(Refer Slide Time: 08:00)

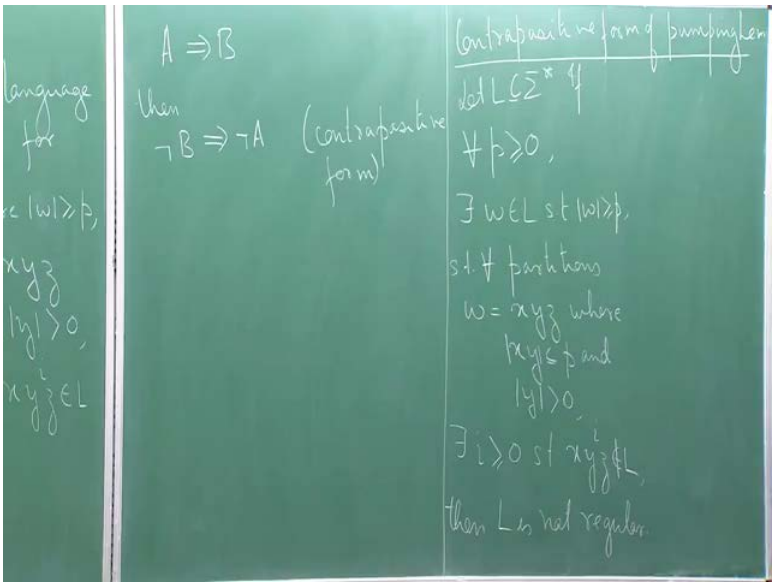


So, just to give a brief recap what we mean by contra positive. If we have a statement that A implies B, so let say we have some statement A implies B. For example, that if a number is let say A is the statement that a number is divisible by 4, implies that the number is even then the contra positive form of the statement is that naught B implies naught A. So, naught of B implies naught of A this is the contra positive form. If I look at the earlier example that A is number is divisible by 4 implies that the number is even, so

which the contra positive form then says is that if a number is not even then it implies that the number cannot be divisible by 4. So, that is essentially what we mean by contra positive form.

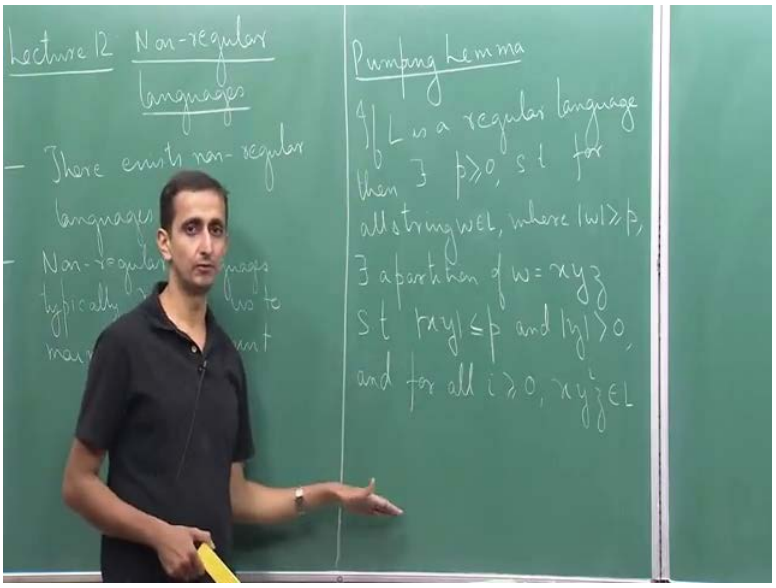
Similarly, in this case we will use the pumping lemma in it is contra positive form or what we will use is that if I have a language where this entire condition is not true, then it implies that the language is not regular. Let us look at the contra positive form of this statement. And again the way we will look at the contra positive form is by thinking of it as a game. We will treat the pumping lemma as a game, where this game is being played by let say you and your opponent. So, whenever you have a there exist move, it means that you are free to choose whatever that there exist quantifies. And if you have a for all move, it is basically your opponent who is doing the choice for you. So, you do not have any option. Let us write this contra positive form of the pumping lemma.

(Refer Slide Time: 10:14)



Let us try to negate whatever we have here in our pumping lemma, let starts with the language L, so let L be some language. So, if for all p greater than or equal to 0, there exists a string w in L, such that length of w is at least p such that for all partitions w equals x y z, where length of x y is at most p. And length of y is strictly greater than 0 there exists sum i greater than or equal to zero, such that x y to the power i z.

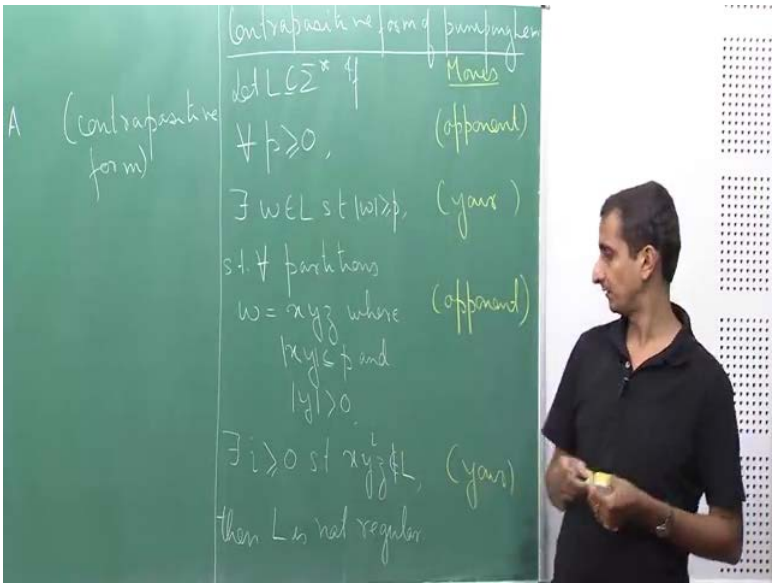
(Refer Slide Time: 12:28)



Let us go back to the pumping lemma. So, in the pumping lemma, we said that if we have such a things if I negate all these things So, instead of there exists p greater than 0, I look at I write for all p greater than 0. Instead of for all strings, I look at I pick a string and instead of a fix partition, I look at all partitions. So, then they it is says is that for all i, x y to the power i z is in L. Here, we will say that such that x y to the power i z is not in L, then L is not regular. This is basically the contra positive form of the statement. And as I said we are going to treat the contra positive form as a game.

This is the game between you and your opponent. This game has four moves, so we have the first place is we are choosing p. The second is we are choosing a string w, whose length is greater than p. In the third step, we are choosing a partition of w, which satisfies these constraints. And in the fourth step, we choose an i such that x y to the power i z is not in L. So these moves have different quantification for all there exists for all and there exists.

(Refer Slide Time: 14:08)



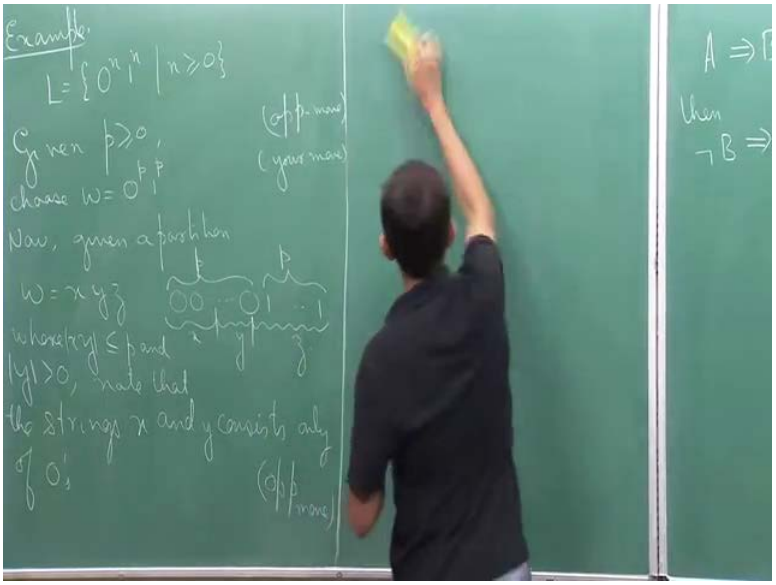
So if it is for all moves, it is your opponents move. This is the move that your opponent will make, so you do not have any control. So, your opponent can pick any number p greater than 0. So, whatever move that you decide to make in the next step should take care of the fact that p can be any number that is greater than 0, but here we have a there exists move. This means that this is your move basically, so these are all the moves. So, this is your move. Now you can pick any w as long as the length of w is more than p, you are free to pick any w in the language L. So, only two things you have take care here; one the string should be in language, and two the length of the string should be more than p.

Now again now that you have made your move; again it is the time for your opponent to make a move. Now the string w that you have picked, your opponent will try to partition it in such a way so basically so that you lose. Basically, the goal of your opponent is to defeat you and your goal is to somehow win. Your opponent will pick some partition of the string w into three sub strings x y and z, such that he has to satisfy the constraint that x y is at most p, and the length of y is 0. Although your opponent can make any partition, but at least he has to satisfy this constraint that x y is at most p and y is greater than 0. So, this is again your opponents move.

And now after your opponent makes a move that is after your opponents gives you a partition of w into x y z, you can choose any i. Again this is a there exists move So, this

is your move. So, you are free to pick any I such that x y to the power i z is not in L. And if you end up doing so, if you win, then you have shown that L is not regular. And if you lose then it does not show anything. Basically if a language is not regular then actually there may be a way which will make you win. Let us try to look at this with an example. Let us try to take a language as an example, and we will try to prove that it is not a regular language using the contra positive form of the pumping lemma.

(Refer Slide Time: 17:11)



We take the language L, which consists of all string of the font 0 to the power n, 1 to the power n, where n is a number let say greater than or equal to 0. This language consists of all strings where you have a sequence of zeroes followed by a sequence of ones such that the number of zeroes is equal to the number of ones. And if you look at this string, you can intuitively realize that in order to check that a string w over 0 1 star whether it belongs to l or not. We need to check two things, whether it is a sequence of zeroes followed by the sequence of ones, and the number of zeroes is equal to the number of ones. So, this requires intuitively some sort of counting.

If you want to construct an automaton for it, the automaton will need to count how many zeroes are there and that is something which is not possible by a finite automaton, an automaton which has finite number of strings because the number n can be anything I mean it can be as large as possible. We will show that L is not regular. So, recall the game that we had designed so of the first move is your opponent move. So, your

opponent gives you a p, so you are given a number p greater than 0. So, you do not have any control over what the number is, so we will just take that as a variable, so it can be anything.

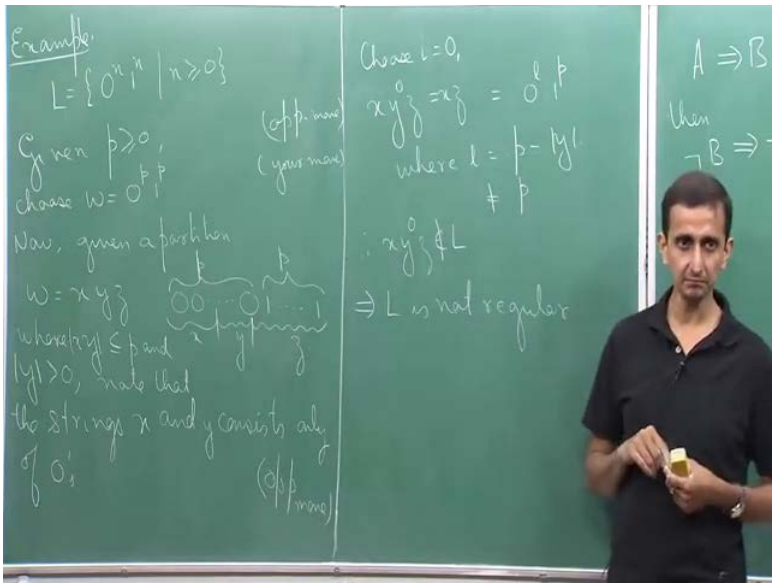
But, now you can choose a string from the language L, which has length more than p. This was your opponents move. So, you choose w equal to in this case we will choose w to be the string, 0 to the power p 1 to the power p. Typically when we are actually designing, when we are trying to prove that a language is not regular using a pumping lemma, our choice of the string w in the second step that is when you are making the move will be a string which is a function of the number p that was given to you by your opponent. So, whatever length it has or whatever symbols it has will be a function of I mean typically it will be a function of p.

In this case, we have p number of zeros and p number ones; you can easily see that the length is more than p, and it belongs to the language L. This was your opponents move and this is your move. Now your opponent takes this string and partitions this into three strings. Now given a partition w equal to x y z, how can the partition look like I mean how can the x y's and z's look like. Let us try to picture this, so how does your string w look like. So, your string w is a sequence of zeroes, so it has some p 0s followed by p 1s. So, if I take the string x y z, what can the string x and y consist of.

The question is can x y consist of one or can the string x or the string y have any one in it. The answer is no, and the reason for that is that when we have this partition, this partition actually has to satisfy the constraint that the length of the x y is at most p. So, since the length of x y is at most p, what can happen is that may be x is some part over here, and y is some part over here, and may be remaining part of it is z. So, may be x is till this point sorry x is till this point, y is till this point, and z is the remainder.

But what is important is that so given a partition x y z, where x y is at most p, and length of y is greater than 0. Note that the strings x and y consists only of 0s. Now it is time to you to make a move. So, this was your opponents move. Now it is time for you to make a move. This was your opponents move. What I will you choose such that x y to the power i z is not in the language. I mean as you can see, no matter what are you choose, I mean if you choose any i that is not one, it will be done. For example, let us take i equal to 0.

(Refer Slide Time: 23:38)



So, choose i equal to zero, so if you choose i equal to 0, what we have. Then the string x y to the power 0 z is nothing but the string x z; and what does the string x z look like. So, the string x z will have some L number of zeroes in it and of course, the number ones is equal to p. And is it possible that L is equal to p. The answer for that is no, because the length of y when your opponent had made the partition, the length of y is greater than 0. If we write it will to 0 to the power L 1 to the power p, where basically L is nothing but p minus the length of y. How many our zeroes were there earlier minus basically what we get by removing y, and this is not equal to p. So, therefore, x y to the power o z is not in L, which implies that L is not regular.

The key point in this proof is that when we choose the string w, we have to choose it carefully. If you note that so now you can realize that why we chose 0 to the power p, 1 to the power p, because we wanted to actually restrict the x y's to contain only 0s. If you wanted the string x y contain only zero and that is why we had chosen it in this particular manner. And which actually constraint your opponent, no matter what partition of x y z your opponent picks, it always must satisfy the property that in particular the string y consists only of 0s. And the length of y is greater than 0, which means that if we take x y to the power 0 z, the number of zeroes is less than the number of 1s.

Here we had taken i equal to 0, what if we take i equal to 2. Again if we take i equal to 2, you will see that the number of 0s in that case will be whatever 0s were there earlier plus

the length of y, because we have y additional the length of y many additional 0s. And since the length of y is greater than 0, which again means that that number is not equal to p. So, i equal to 2 would have also work; hence in the similar manner i could 3, 4 everything the only I that will not work is i equal to 1.

This is as I said one of the standard ways of proving that a language is not regular there are we can also use in certain cases closure properties into show that a language is not regular. And in our next lecture, we will see some more examples that some more examples of non-regular languages and how to prove that they are non-regular.

Thank you.