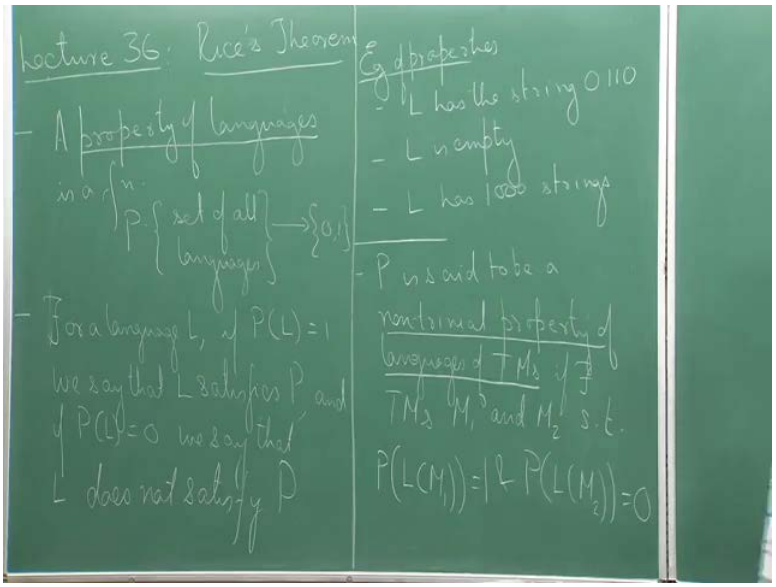
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute Of Technology, Kanpur

Lecture 36
Rice's theorem

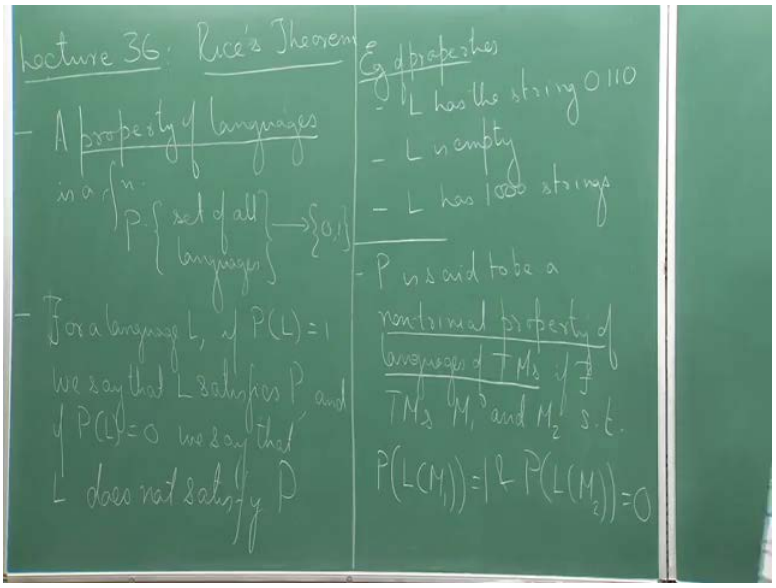
Welcome to the 36 lecture of this course. Today, we will talk about this important theorem known as Rice's theorem in the context of undecidability. So, Rice's theorem gives us a mechanism to show the undecidability of an infinite set of languages. So, it shows that if a language as a certain form then we can prove that is undecidable. So, in that sense it is a powerful theorem and what we are going to show today is we will first see the statement of Rice's theorem, and then we will look at a proof of the correctness of the statement.

(Refer Slide Time: 01:06)



So, before I state Rice's theorem I need to introduce a few definitions. So, the first thing that we will see is a property of languages. A property of languages is a function P that goes from the set of all languages to 0 1. So, basically we associate with every language either 0 or 1, so what that means is that so if we associate 1, we say that the language satisfy the property P; and we associated 0, we say the language does not satisfy P. So, for a language L, if P of L is 1, we say that L satisfies P; and if P of L is 0, we say that L does not satisfy the property P, so this is what we mean by property of languages.

(Refer Slide Time: 03:16)

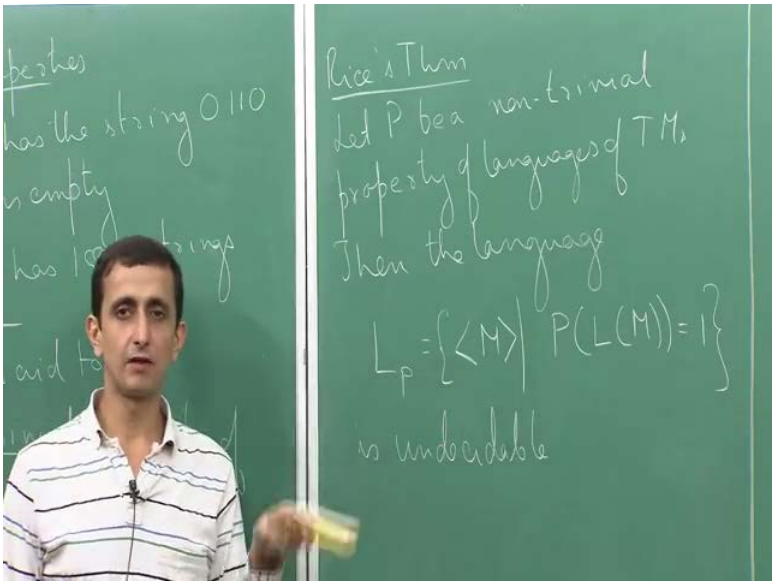


So, some examples, so you can construct any examples let us say that so examples of properties. So, the language I will just use L for language. So, L has the string 0 1 1 0, so there are some languages which will have that string, there are some languages which will not have that string, so this is the property. So such that languages which have this string we will say that those languages satisfied this property; others, do not satisfied this property. The L is empty, so basically this means that the languages which are empty for them we say that in fact, this is only one empty language so that is the only language which satisfies this property all other languages will not satisfy this property. L has 1000 strings. Once again there are a finite number of languages which have so many strings other languages have either less or more number of strings and so on, you can have lots and lots of properties.

So, now we defined what is called a non-trivial property of languages of Turing machines. So, P is said to be a non-trivial property of languages of Turing machines, if there exists Turing machines M 1 and M 2 such that P of L of M 1 is 1 and P of L of M 2 is 0. So, we say that a property is a non-trivial property of languages of Turing machine, if there is some Turing machine whose language satisfies that property, and there is some Turing machine whose language does not satisfy that property. For example, if I say that if I look at all of these properties, so there are Turing machines whose languages empty there are Turing machine whose languages not empty. There are Turing machines whose languages has 1000 strings, we can construct; there are Turing machines whose language

has more than thousands or less than thousand strings, similarly for the first property also. So, all these three properties are non-trivial properties of languages of Turing machines.

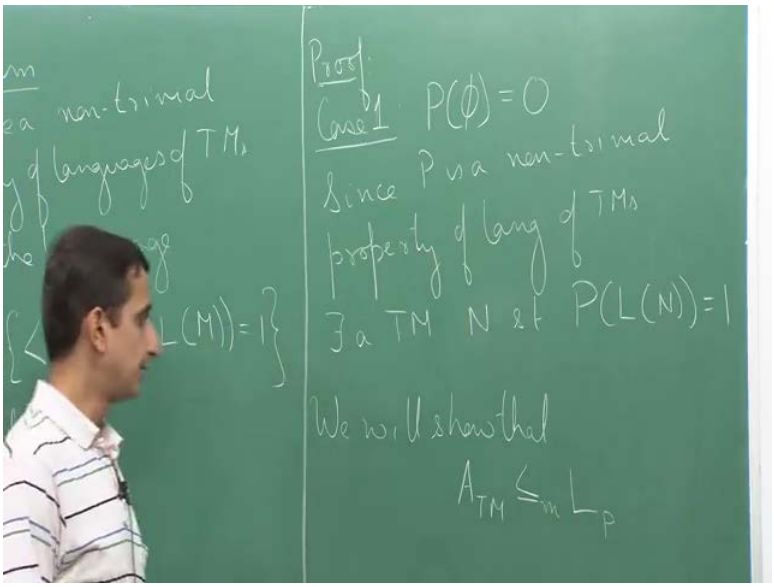
(Refer Slide Time: 07:33)



So, now, we can state Rice's theorem as follows. Let P be a non-trivial property of languages of Turing machines, then the language consisting of those Turing machines. So, I will call it L subscript P encodings of machines M such that L of M satisfies P or I can write it as P of L of M is equal to 1 is undecidable.

So, if we have a non-trivial property of languages of the Turing machines then the language of encodings of Turing machines whose language satisfies the property is undecidable, which also means that the language of encoding of machines whose language does not satisfy the property is also an undecidable, because decidable problems are closed under compliment. So, what it means is that so we had actually already proven for this. So, remember that the language E TM that we had shown consisted of encoding of all those machines whose language was empty so that is basically one way to show that. So, this if you prove Rice's theorem it gives us an alternate proof of the fact that E TM is undecidable.

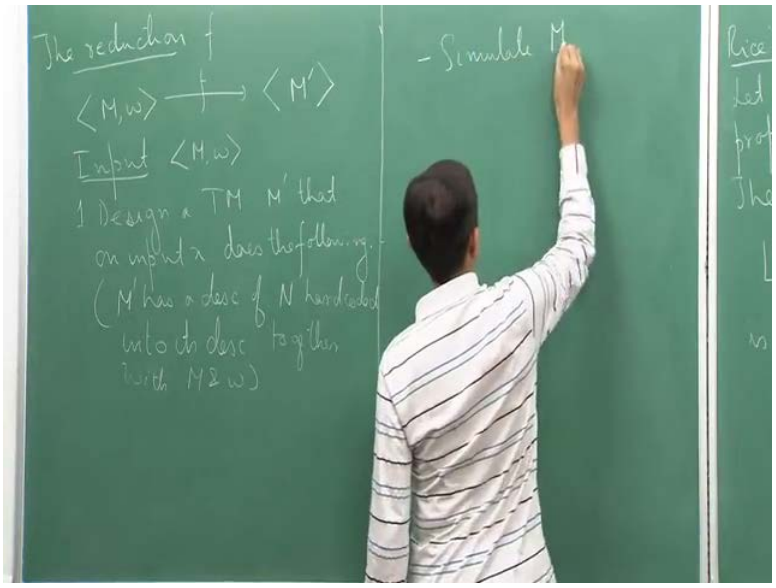
(Refer Slide Time: 09:58)



So, how do you prove this property, so how do you prove the theorem? So, we will prove the theorem in two parts in two cases. So case 1, so consider the language phi that is the empty language. So case 1 is assume that so we have this property P, assume that P of phi is 0. So if we assume that P of phi is 0 then what we can say is that since P is a non-trivial property of languages of Turing machines there exists a Turing machine such that the language of so there exists the Turing machine, let us call it N. Such that the language of N satisfies the property in or in other words P o f L of N equals 1.

So, here we have a language which does not satisfy the property, so that means that there must be some other Turing machine whose language satisfies. And of course, we can construct a Turing machine, whose language is empty set so we just have a Turing machine which rejects all inputs so of course have a Turing machine for phi. So, now using this Turing machine, we will prove that so using this, we will show that A TM reduces to L P, so we call that a TM is undecidable so this will prove that L P is undecidable.

(Refer Slide Time: 13:05)

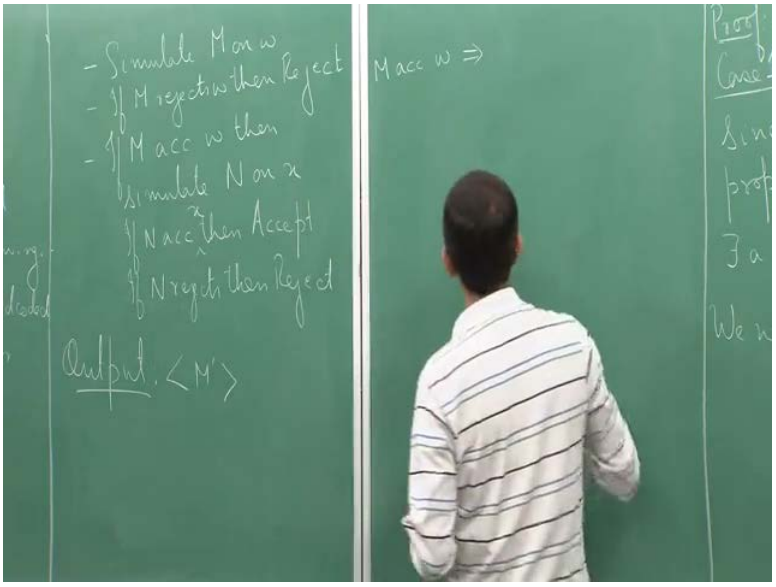


So what is the reduction? The reduction let us call it f. So, what will f do, so f is a function which will take an instance of A TM and it will produce an instance of L P. So, what f does is that it takes let say an instance of A TM M coma w. And it will produce and instance let say M prime. So, we have to design M prime given M coma w; this is what we have to do. So, we take as input M coma w and then we give a construction of M. So, design a Turing machine M prime that on any input x does the following. So, before I give the description of M prime, let me say something. So, we just saw that there exists a Turing machine M, whose language satisfies the property. So, we saw this existence and we said that this Turing machine we called it as N.

So, what we will assume is that the Turing machine M prime that we are going to design as a copy of N hard coded to its description. Of course, it as M w also, but it also as this Turing machine hard coded. So, let we write it. So, M prime has a description of N hard coded into its description together with of course M and w. So, once again we saw how should we picture this, I mean picture this as so you want to write a program M prime or you want to write a design an algorithm M prime. So, this algorithm what it does is that so inside the description of the algorithm or inside the description of the program there is already a description of N, M and w that is given in the description. It is not part of input; it is the part of description of this program. Now what it does is that now given an any input x to M prime, now we have to decide how will the Turing machine M prime behave on x.

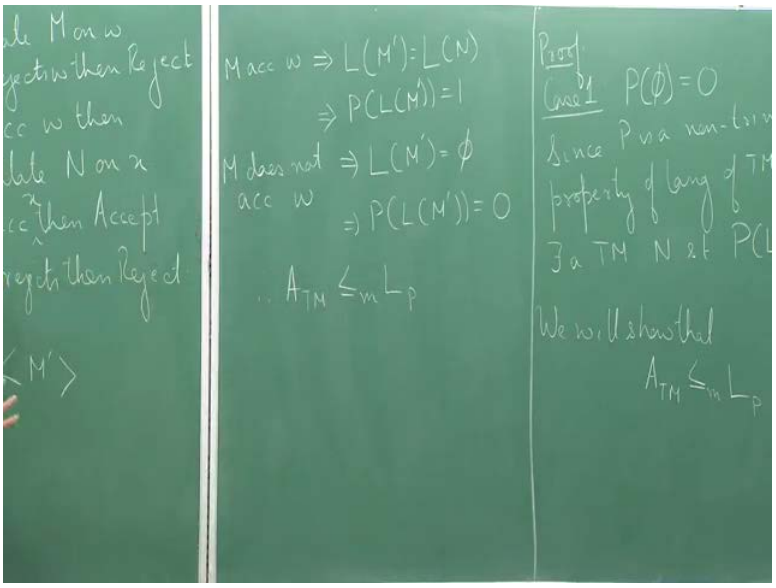
So, let us I want to emphasize this fact because, this is very important that all this three things N, M and w they are not the part of the input; in other words, they do not change with different input. So x is what will change every time you run the program or it might change, but these three things they remain constant for every time you run the machine M prime.

(Refer Slide Time: 17:00)



So, the first M prime it will simulate M on w; if M rejects w, then it rejects, it does not do anything. If M accepts w then simulate N on x, so if the machine M accepts w, so here there is two things that can happen, in fact, three things. If M rejects w then we reject if M run forever on w then of course it runs forever, it never goes to the second step; but if M accepts w, then what we do is that we take the description of the machine M and we simulate the input to M prime on the machine N. Now if N accepts then accept; and if N rejects then of course reject. So, our machine M prime is rejecting here once N rejects it will reject and if M rejects then also it rejects. The only time it accepts a string if M accepts w and N accepts x. So let me write it here explicitly; N accepts x and then we accept. So, now so that is also this is the description of the machine. And now once we have this description we just output the description of M prime.

(Refer Slide Time: 19:24)

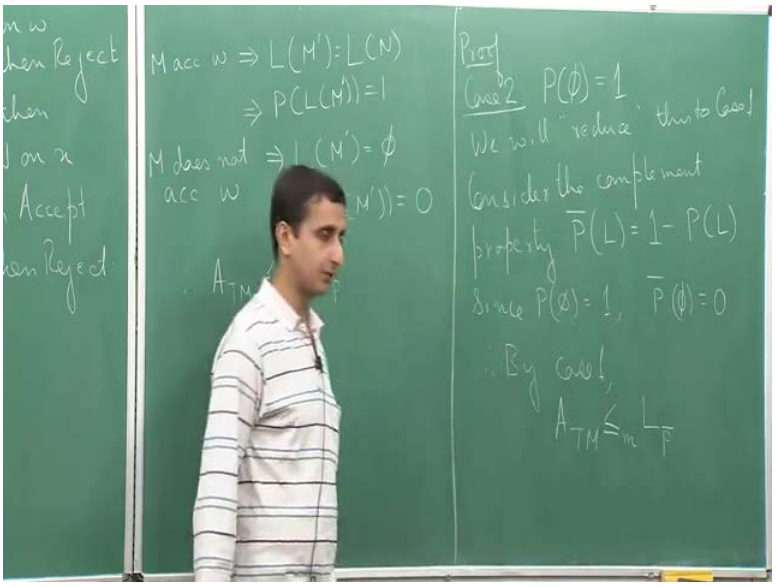


So, now, let us see why this reduction is correct. So, suppose if M accepts w, so what happens in this case; if M accepts w then what we have is that M coma w is a s instance of A TM. So, it means M w is accepted by a machine that accepts A TM. So, if M accepts w then what we have here is that simulate N on ox, if N accepts x then we go ahead accept x. So, then the language of M prime is going to be all those strings that are accepted by the machine M is nothing but the language of N. So, whenever N accepts an x, M prime will accept the same x which implies that so the language of M prime is the language of N and we know that so again from our assumption, we know that the language of N satisfies the property it satisfies the property P. So, therefore, P of L of N is equal to 1. On the other hand, if M does not accept w, then we do not even go to the next step so we do not even go to the third step here, so we directly reject. So, in that case the language of M prime is equal to phi.

And now our assumption was that P of phi is 0, which implies that P of L of M prime is 0. So, here actually what I should say is so that since the language of M prime and the language of N are the same. So, therefore, P of L of instead of N, I can write this as M prime also is equal to 1. So, what we have here is that if M accepts w, then P of L of M prime is one or the language of M prime satisfies P; and if M does not accept w then the language of M prime does not satisfy P. So, therefore, A TM reduces to the language of P. So, this is the first part of the proof. So, remember that we had only considered the case when P of phi was equal to 0. So, now, we need to consider the case, when P of phi is

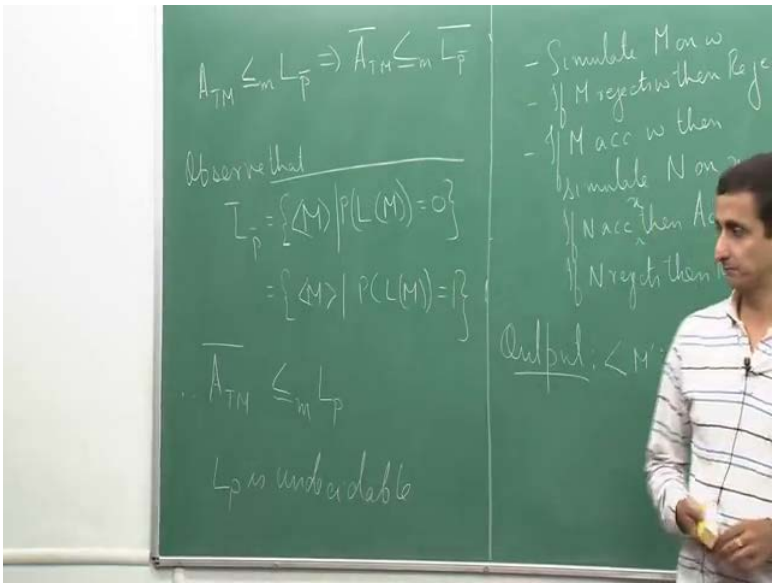
equal to 1, but actually this case is not very different; in fact, what we will do is we will reduce this case to the case that we just saw.

(Refer Slide Time: 22:44)



So, suppose P of phi is equal to 1, so this is case 2. So, we will reduce this to case 1. So, if P of phi is equal to 1, consider the compliment property P bar. So, what is P bar so whenever P of a languages equal to 1, P bar of that language is will be 0; and whenever P of a language is equal to 0, P bar of that language is equal to 1. So, it basically just assigns the other bit – the compliment bit. I mean if you want you can define it also, so P bar of a language L is equal to 1 minus P of L, so if P of L is 0 then P bar of L is 1; and P of L is 1, then P bar of L is 0. So, now, since P phi is equal to 1, therefore P bar of phi is equal to 0. Now whatever we said for in case 1, for P of phi we can say it for P bar of phi. So, therefore, by case 1 what we have is that A TM reduces to L of P bar, so this is by case 1. Now let us understand here what is happening here.

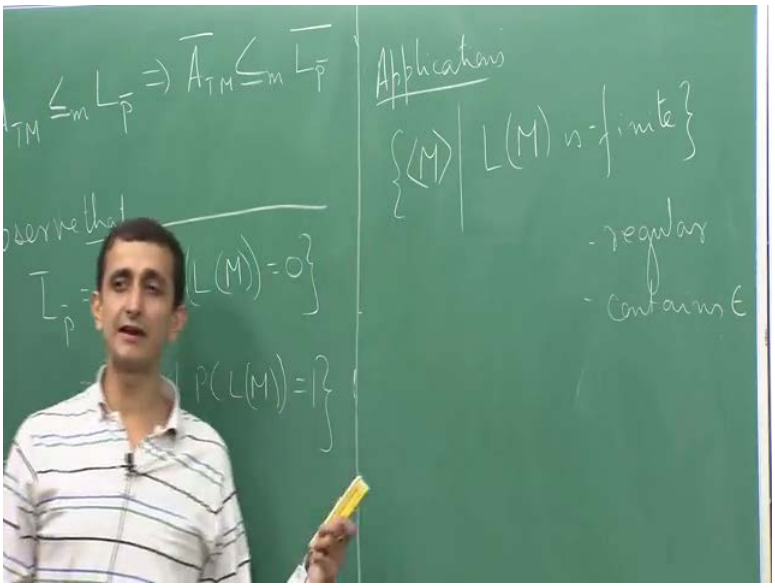
(Refer Slide Time: 25:08)



So, if A TM reduces to L of P bar then what I can say is that A TM bar reduces to L of P bar whole bar. So, this actually again follows from definition of reduction; so remember so recall the definition of reductions, so how did we define? So, we said that a language L reduces to a language L prime, if for all x that belongs to L is mapped to L prime, and every x that does not belong to L that is which belongs to L compliment is mapped to L prime compliment, so that is all that we are using here. So, if A TM reduces to L P bar then A TM bar reduces to L bar of P bar, so I mean the general statement is that if a reduces to b then a bar reduces to b bar.

Now what is this language L bar P bar, so before I state, so observe that L bar of P bar, how do we do define. So, it is set of all those machines M, so L of P bar is all those machines M which does not satisfy P, so L bar of P bar will be all those will be the compliment of that language. So, therefore, it will be all those machines M such that L of M or just to add one more step. So, it is compliment of the language such that L of M does not satisfy P with a bar on top which is nothing but all those machines M which satisfy P, therefore, A TM bar reduces to L P. So, in case 1, we had shown that if P of phi is equal to 0, then A TM reduces to L P and in case 2 what we showed is that a TM bar reduces to L P. So, both A TM and A TM bar are undecidable problems, therefore L P is undecidable, so that completes the proof of Rice's theorem.

(Refer Slide Time: 28:17)



So, few quick applications, so for example, if we consider language of machines M such that lets say L of M is finite. So, there are machines whose languages are finite, there are machines whose languages is not finite, so this is undecidable. Instead of finite, now we can have let say regular which we have already shown language of machine which contains the empty string, so which contains epsilon, so all this so you can construct an infinite set of languages which are undecidable. So, all these problems will be undecidable by just applying Rice's theorem. So, I will stop here today.

Thank you.