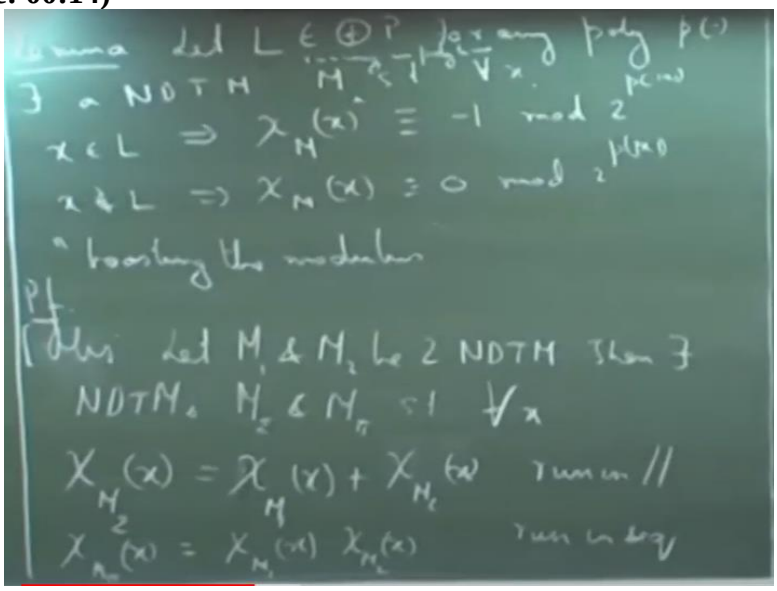
Computation Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Module No # 06
Lecture No # 27
Toda's Theorem - II

(Refer Slide Time: 00:14)



Deterministic turning machine M such that for all x, if x belongs to L then the number of parts of this machine M on x is congruent to -1 mod 2 to the power P x. And otherwise it is congruent to 0 mod 2 to the power P x so what are we saying were essentially? So suppose if you take any language in parity P what that means by definition is that there exists a non-deterministic so no not only a non-deterministic but a running in a polynomial time.

So what we have is by definition if we have if you take a language in parity P by definition there is a NP machine such that if x belongs to L. The number of parts of M on x is congruent to 1 mod 2. And otherwise it is congruent to 0 mod 2 that is what we have by definition. So what we are doing here is we are essentially boosting the modulus. For any polynomial P we are boosting it to 2 to the power p. So that is what we are doing in this Lemma.

So let us see why this can be done, so that is what we want to ensure. So it is not just any machine, so it is not the parity P machine so there is some machine. Let us say M tilde which acts as an evidence of the fact that L belongs to parity P. Which has; the property that Chi M

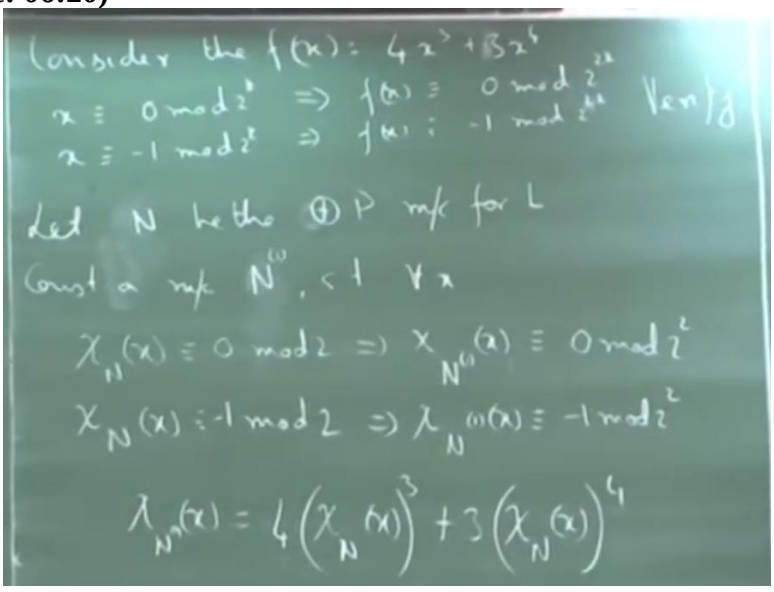
tilde X will be congruent, to 1 mod 2 and this will be congruent to 0 mod 2. But what will show is that we can also construct another machine M which has this nice property.

I mean why this is nice? We will see later but let us see why this is true? So before I give the proof let us make a small observation. So let M 1 and M 2, be 2. So whenever we talk about the non-deterministic machines for the rest of the lecture we always mean polynomially whose run time is polynomially bound. So if we have 2 non- deterministic machines then there exist NDTM'S.

Let me call them M sigma and M pi such that for all x the number of accepting parts of M sigma on any given X is equal to the number of accepting parts of M 1+ the number of accepting parts of M 2. And similarly we can say the same thing about or similar thing about M pi where the number of accepting; parts are product of 2 machine M 1 and M 2. So why is this true? Actually it not only gives existence actually we can construct these machines M pi and M sigma very easily given M 1 and M 2 how is that?

So basically you non-deterministically decide whether to run M 1 or M 2 for M sigma. So which or in other words you just run these 2 machines in parallel. So then the total number of accepting parts of M sigma will be the sum and for M pi you just run them in sequence. So here you run in parallel and for this you run them in sequence so for every first you run M 1 whenever M 1 accepts then you decide to run M 2. And whenever M 2 accepts then your M pi will accept.

(Refer Slide Time: 06:20)



So now consider the following function this is a sort of a magic function, well not magic but. So this function has the nice property that, if x is congruent to 0 mod 2 to the power k for any k. Then f of x is congruent to 0 mod 2 to the power 2 k. And if x is congruent to -1 mod 2 to the power k then f of x will be congruent to again -1 mod 2 to the power 2 k, so I will just leave this as an exercise you can verify this.

So I mean the first part is very easy because any way I will have a x square term that comes outside. So the number of parts will be 2 to the power k times 2 to the power k. And for this -1 the point to note here is that f x cube+3 x to the power 4+1 has x+1 whole square as a factor. So I mean that is the main idea rest is just filling out the details. You mean 3 K that is all that with the, I mean that is trivially we have because we anyway have. I mean if we take function f x to be the identity function.

We any way have 0 mod 2 to the power k and -1 mod 2 to the power k what we want is more stronger. We so our goal is to boost the modulus. So now the rest is easy because what we do is. So let N be the parity P machine for L so construct a machine N 1 such that for all x if Chi N of x is congruent to 0 mod 2. Then Chi let me call this not N 1 but N superscript 1 chi N 1 of x is congruent to 0 mod 2 square.

And if chi N of x is congruent to 1 mod 2 then Chi N 1 of x is congruent to -1 mod 2 square. So here so again so we have a machine M because our language belongs to parity P which means that X belongs to the language. Or in other words so x does not belong to the language if the number of accepting parts is even. And if x belongs to the language then the number of accepting parts is odd. And by our construction so construction is basically so what is Chi N 1 of x?

So Chi N 1 of x here is nothing but so what is it? So in terms of the function notation if I just want to write it is nothing but 4 Chi N of x whole cube + 3 Chi N of x whole to the power of 4. So the thing to be noted here is that it does not matter since we are working in the field of size 2, 1 is seen as -1. So if x belongs to L I can say I mean instead of writing it this way I can also write that Chi N of x is congruent to -1 mod 2.

So, 1 mod 2 and -1 mod 2 are the same things in set 2. So what this function f tells us is if I apply the function f to this number it can be easily boosted to the 2 to the power of 2**.** And the reason

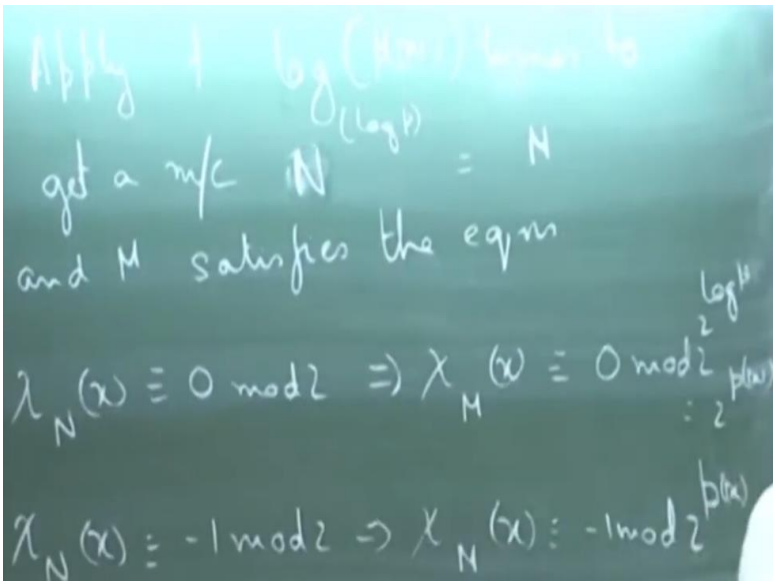
why we can apply this is because of this observation because what do we do to say to get this part so we multiply 4 with so basically we construct a machine. Let us say some N prime which is the product of N with itself.

And then again we consider it is product with N again. So that we will give us x is q and then we consider let say a trivial machine which has only 4 accepting parts and we consider the product of that machine with whatever machine we got here. So that will give us this part, similarly we get the other part and we combine them. So this combination can be, 3 if you run 3, if you run 4 of them in parallel then**.**

You want to run 4 of them in (()) (14:09) so that is basically taking this summing these 4 times. Of course so because our ultimate goal is to come up with a machine M which has this property what we are here actually shows 2 things. First it shows existence of a machine which is what you are telling but actually what we are giving is something more, stronger. We are constructing a machine M which will have this property.

So a (()) (14:44) you do not know so, when you are given this machine N the party P machine for machine L you do not know how many parts N has. And that is something which is which you even cannot get in polynomial time or even in parity P. The only thing that you can get in parity P is the parity of the number of parts. But what we want to show is that we can construct another machine very efficiently which will have the property that the number of parts will satisfy this function. Think about it may be we can talk after class if you are not still convinced.

(Refer Slide Time: 15:40)



So what we do we want now so apply f, how many times do we need to apply f? To get a machine N log P so I will just call this, so this will be our machine M actually and M satisfies the equations that if Chi N of x is congruent to 0 mod 2. Then Chi M of x is congruent to 0 mod 2 to the power log P which is nothing but 2 to the power P. And similarly if chi N of x is congruent to -1 mod 2 then Chi M of x will be congruent to -1 mod 2 to the power P of x.

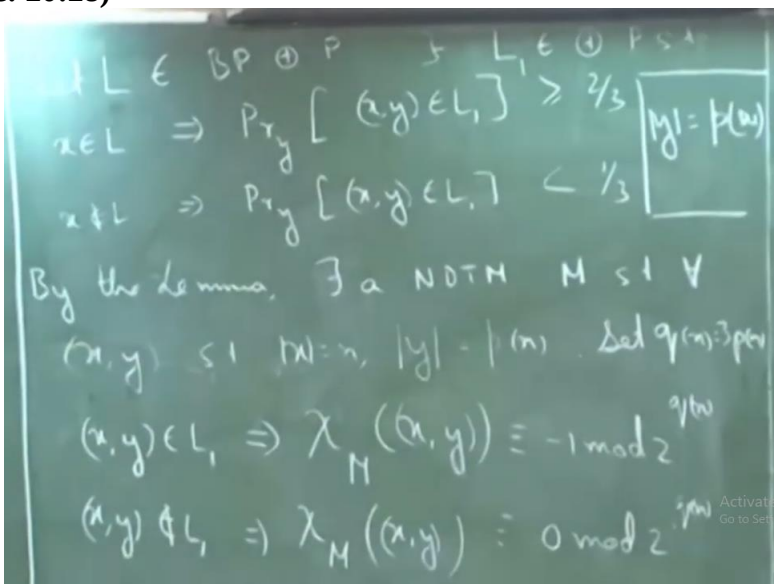
So that proves the Lemma what you can do I mean so that can be taken care of so what you can do is you go to the. So you take the floor function, you go to the integer which is less than log (()))) (18:13) whatever I mean 2 to the power floor of log P. And then what is left is basically P of x -2 to the power log to the power 2 to the power log of. So whatever is the difference P of x-2, to the power floor of log P of x.

So now you again I mean use the same function too. Well the way I stated here it will not work. So what we can assume is that for any polynomial P you look at the next highest power of 2 you look at a power of 2 which is greater than P of x. And you go up to that polynomial that will actually suffice. So you are right so if P x is not a polynomial then this will actually not work. But you can actually go to the next power that will actually suffice.

So do I mean you can look at another construction? So that instead of going from mod 2 to mod 2 square you or may be here from 2 to the power k to 2 to the power 2 k, you go from 2 to the power k to 2 to the power k +1. So that is why I said that may be this function will not work but I

am sure that some other function will definitely work. So you can always take it to some exact polynomial P of x but that is also not necessary so the way we will use this Lemma even P of x which is power of 2 will work for us.

(Refer Slide Time: 20:28)

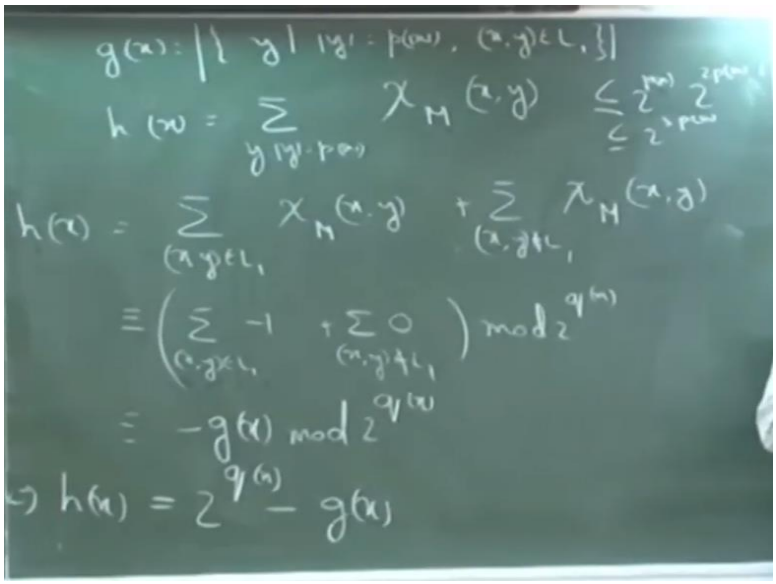


So let us come back to remaining part of the theorem So let L be a language in BP dot parity dot P so this implies that there exist a parity P language. L 1 such that x is in L implies that the probability that x, y belongs to L 1 is greater than 2 third, so here we are using the first definition of BPP that we gave. We do not need the stronger version and if x does not belong to L then the probably that x y belongs to L 1 is less than 1 third.

So this is the definition of BP dot parity dot P. So now we do some counting so let us use the Lemma so by the Lemma for all. x, y belongs to L 1 so note that L 1 is a parity P machine. So there exist a non-deterministic turing machine M such that for all pairs x, y. Such that let say n x has some length N and length of y is some p of n. x, y belongs to L 1 implies that Chi M on x, y is congruent to -1 mod 2 to the power p of n.

And if x, y does not belong to L 1 and this is congruent to 0 mod 2 to the power p n so because L 1 is a parity P language. So now let us define so we are almost there so let us define 2 functions.

(Refer Slide Time: 24:13)



g of x is the number of good certificates for a given x. So given an x, g of x is the number y such that cardinality of y is p of x and x, y belongs to L 1. So here what so actually **so** the thing is how do we do get this polynomial p. So p is basically the length of this random string so should have mentioned it here so y has length p of x. So that is why we chose all pairs which have this particular property that x has some length n and y has length P of n.

So g of x is the all set of all good certificates for a string x in the sense that if x belongs to so if x belongs to L what is g of x? All, the y is such that the x, y belongs to L 1 and since the probability of picking a y is greater than a 2 third. So the size of the set is greater than 2 third times 2 to the power p of x, because 2 to the power p of x is the total number of strings of length p of x.

And if x does not belong to L 1 this set has size less than 1 third of 2 to the power P of X. So g of x actually has this gap between these 2, type of x's. So now let us define another function h of x which is summed up over all y's such that cardinality of y is p of x over Chi M of x, y. So you look at all the y's that have length p of x for each such y you count the number of accepting parts that M has so M is our parity.

Well it is not the parity P machine per se but it is this machine that we have and you sum them up. So what can you say about h of x in term of g of x. We just want to make a small correction so this will be sum q of x so I will come back and fix what this q of x is later on. We will take it

up to sum q of x so h of x can be summed up over all pairs x, y that belong to L 1 of Chi M x, y + x, y which do not belong to L 1.

So I am just dividing into 2 sets y's which make it lie within the language and otherwise. So what about this number? So this is sigma so if x y belongs to L 1 we know that this is a -1 mod 2 to the power q of n. So we have -1+sigma 0 and I just take the mod 2 to the power q n outside so this is congruent. So now this part is basically 0 because I am just summing up 0 and what about this part? What is this number equal to? But more exactly what is this equal to.

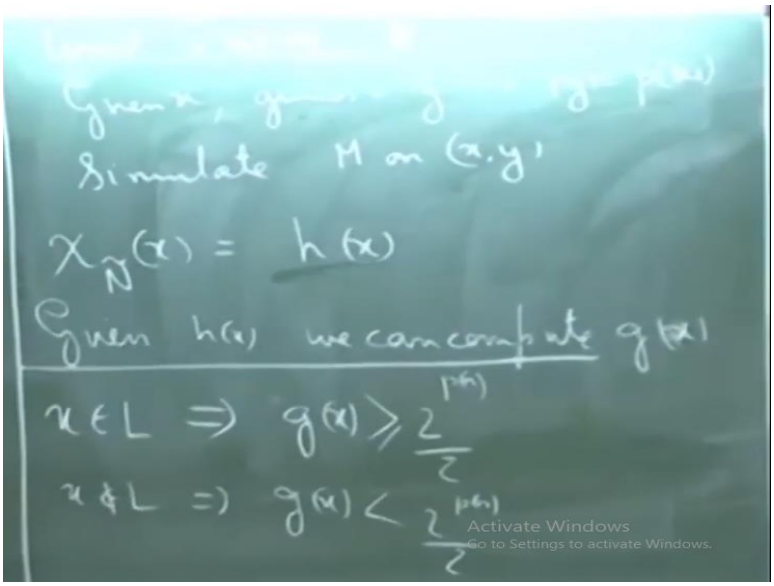
So this is basically all, those y is which make x belong to our L or in other words all those pairs x, y which belong to L 1. So this is exactly the number g of x by definition. So this is – g of x mod 2 to the power q of x so now let us go back to the function h of x. So what will be an upper bound on h of x so what is so that is what we have not set q x yet, so I want to set q x which large enough.

So let us look at h of x first so what is an upper bound in this number. Maybe a little bit more because this is a machine which takes a string which takes 2 strings as input 1 has length n and other has length p of n so basically takes as input a string of length n + p of n. So it is total the number of parts can be 2 to the power n + p of n. So I will just roughly denote that to be 2 to the power 2 p of n.

And I am summing it up over all y's of size again p of n so that is another 2 to the power p of n. So therefore this is less than 2 to the power 3 p of n, h of x because the total number of y's which satisfy this property is 2 to the power p of n and Chi M x, y can at most be 2 to the power n + p of n which is at most 2 to the power 2 p of n. So now we can set our q's so we will set our q of n to be 3 p of n so that is why I had said so.

And basically what this allows us to do is if we set our q large enough this immediately implies that h of x = 2 to the power q of n-g of x. So basically; if I divide h of x by 2 to the power q of n the quotient is 0 and the remainder is 2 to the power q of n - g. Because I have chosen my q of n to be large enough that it exceeds h of x. So it will never give a positive quotient, the quotient will always be 0. So that actually completes the analysis so now a little bit is left.

(Refer Slide Time: 33:03)



So I try to finish it so now look at the following machines so construct a NDTM. Let us call it N tilde as follows given x what it does is guess a y of length p of x and then simulate, simulate what? Simulate M on x, y. So basically I want to look at an upper bound on h of x so are you convinced that this is an upper bound on h of x. So h of x is greater than this number. So this number is nothing but 2 to the power q of n by the way I defined q of n.

So if h of x is smaller than this number what is the quotient that I get if I divide h of x by 2 to the power q of n what is the quotient ? Quotient is 0, and by the property that we have here the remainder is 2 to the power q of n - g of x. So just so again the best way is always to work it out yourself so work out the equation yourself it will be clear. So how many parts does N tilde have? So N tilde is a non-deterministic machine how many paths does it have?

Some x on some x so it gets as a y of length p of and the; it stimulates M on x, y. So what are we going here how are we defining h of x? For all Y I am just summing up the number of accepting parts that M has on x, y this basically I mean the first step basically corresponds to this summation. So I am guessing a Y so for all different wise I have different computation parts and then on each of those, computation path I am basically simulating M on this input.

So that; will give me so many so the total number of accepting paths is h of x. So and now by the way we defined our h of x and so given h of x we can compute g of x easily and so now how do we decide if something is in the language. So as I said earlier so x will belong to the language so

if x belongs to a language then we know that g of x is greater than or equal to the 2 to the power P of n by 2.

So just keeping a very crude lower bound and if x does not belong to L we know that g of x is less than 2 to the power P n by 2. So actually know we actually know something more, stronger but that this is crude bound on it. So now the thing is that given any input x we just call the sharp p function corresponding to h of x because h is a function which corresponds to a non-deterministic machine.

So we just make a sharp p called which gives us the value h of x given h of x we compute what g of x is if g of x is greater than this number. We accept the input and if g of x is less than that number we reject that input so that is all. Anyway I will stop here now but so the proof is complete but if you have any more questions or anything else we can discuss a little bit more about it on Wednesday.