Formal Languages and Automata Theory

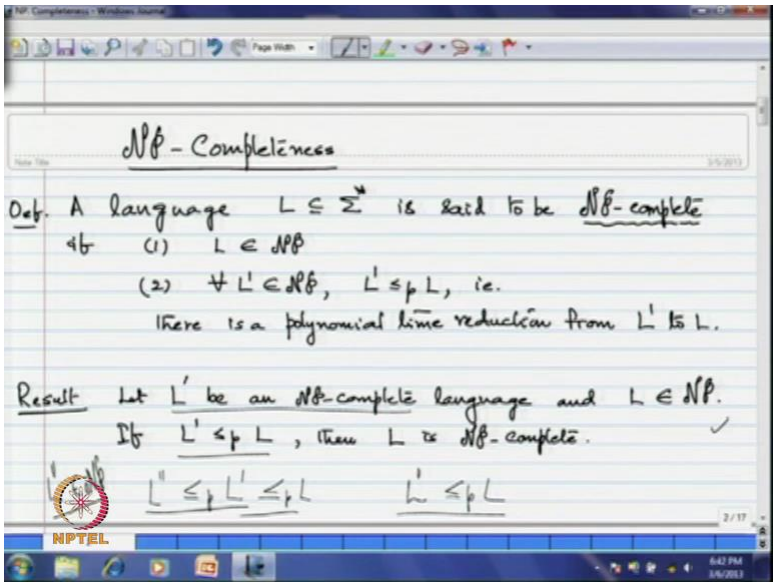
Prof. Dr. K.V. Krishna.
Department of Mathematics
Indian Institute of Technology, Guwahati

Module – 14
Introduction to Complexity Theory
Lecture – 03
NP- Completeness

In the previous lectures, we have already introduce the concept of NP completeness. And I have talked about the importance of NP complete problems or NP complete languages. In this lecture, I will now formally establish some of the NP complete languages systematically, and as I had mentioned that the first problem in the history, which was observed that it is NP complete was satisfiability problem is so called Cook's theorem, Stephen Hook he has established this theorem. This result, so I will be proving this Cook's theorem in the sequel, what we do, although satisfiability problem was the first one to establish that it NP complete.

I will take a class room approach, we will systematically develop the proof of that may not be the original. The proof, but we will systematically develop using the techniques, and the approach that we had adopted. In this course, and give an elegant proof I mean a proof that, it is very quickly understandable based on the previous lectures, we give in this lectures on NP completeness.

(Refer Slide Time: 02:01)



Let me just recall, what is NP complete language, what we have talked, A language L in sigma star is said to be an NP complete language. If it is an NP language, and you take any language in NP the class NP, there is a polynomial time reduction from that language to L. And also, I have mentioned about you know, establishing NP complete languages. In this, you know you look at, if you are knowing an NP complete language say L dash, and to establish a language which is in NP as NP complete.

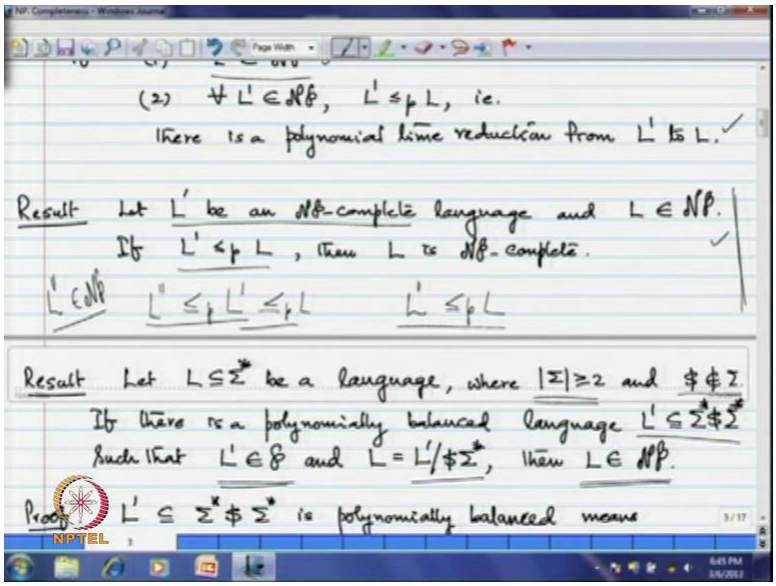
If you can polynomial time reduction from that language L dash to L that is sufficient, why because to L dash thus if you take any L double dash, an NP language from L double dash to L dash you will have a polynomial time reduction, because it is NP complete, because L dash is NP complete. And now, from L dash to L if you have a polynomial time reduction, you know the transitive property of this polynomial time reduction. This reduction in polynomial time, this less than equal to p what we are writing this relation, this is a transitive thing.

So that, you will have L double dash is reduce to L in polynomial time this is you take an arbitrary L double dash in NP. So, take an arbitrary L double dash in NP, since L dash is NP complete, you have this and since this is given, you get the transitivity L double dash is less than equal to p L. So that, every NP language can be reduced to L in polynomial time. So, we have already discussed this theorem, but anyway I wanted to just recall this. So, this one that will be useful, if you are already knowing an NP complete language,

because you see to establish a language is NP complete, you have to observe that this is NP.

And second thing what we have to observe is, for every language in NP we have to give a polynomial time reduction to a, so this is a hard problem. So, and the complex and you see, we have to give such a polynomial time reduction from every language to the consider language L. So instead, what can be done, if you are already knowing a language is NP complete, then you can establish a new language. If you NP complete by give a polynomial time reduction from that language to this, but observing one language is NP complete anyway requires all these, you know thing to be look into.

(Refer Slide Time: 05:28)

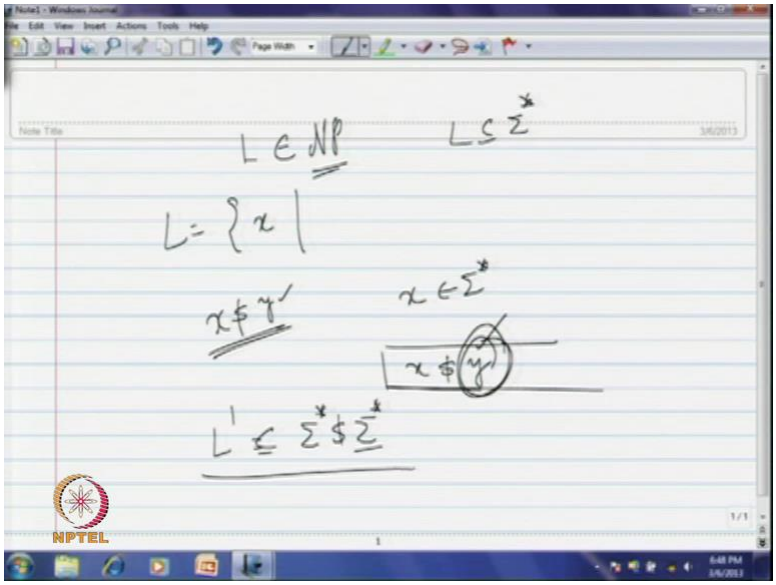


Now, when we are talking about languages in NP to prove this thing, let me first give a criteria here, a language L over sigma an alphabet of size at least 2, and resume dollar is not a symbol in sigma. Now, if there is a polynomially balanced language L dash in sigma star dollar sigma star, so L dash how does an element in L dash look like, that is every element is of the form x dollar y for x and y in sigma star.

So, if you can have a polynomially balanced language L dash, such that this L dash is in p, this is there is a polynomial time deciding algorithm for L dash. And this L is just you know a coefficient of L dash with dollar sigma star, that means in the language L dash the polynomially balanced language L dash. If you just remove all the things from on and after dollar, whatever is there, if that is L then we can observe that this L is in NP,

what exactly is this L dash. So, L dash what we are trying to give here after dollar, because you know what is language in NP a language is in NP.

(Refer Slide Time: 07:23)



If you have a polynomial time verifying procedure, polynomial time verification procedure for the language, and given a certificate. So, for the language L if you want to look at a language L, if you want to see this is in NP. So, suppose these are the strings in L, etcetera. Now, any string you take, if I give a certificate y and if there is a polynomial time procedure to verify this, but how to give this certificate this certificate we give non deterministically.

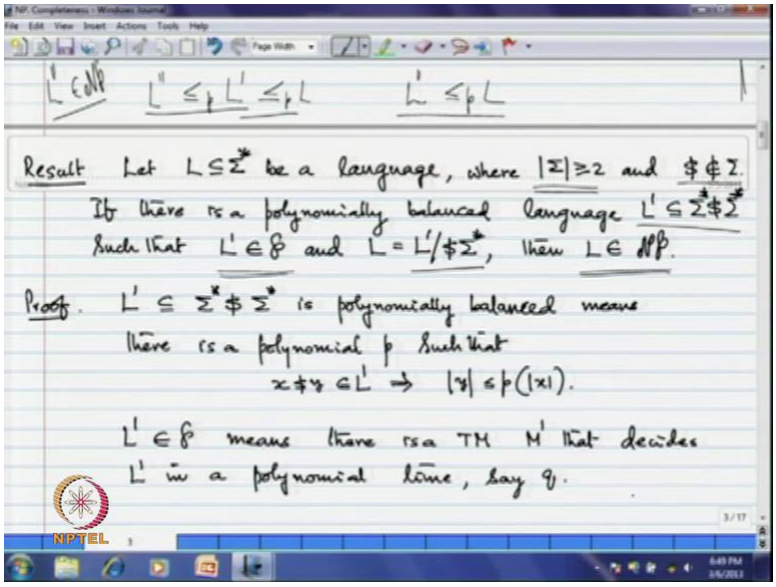
So, that is why this NP you know this non deterministic turing machine, which verifies that x is in the language L or not in polynomial, I mean x is in the language in polynomial time. So, given a string x in sigma star, say L is a language in sigma star, now you given a string x in sigma star, what you require a non deterministic turing machine. You take x as the input, and this non deterministic machine you know has to check whether x is in L.

So, if x is in L, this non deterministic machine halts, and the time it takes should be a polynomial time, that is what is the definition. So alternatively, you know I can say like this, if checking non deterministically means, given a certificate say I will give that on the input itself, I give non deterministically a certificate, a solution you know this, and then this x y. If I have a polynomial time decision for this, polynomial time decision for

this, then I can say the language L is in NP, that is what is exactly we are trying to look at in this criteria.

So once again, now here the y what you are we are taking the language L dash, what we are creating that is subset of sigma star dollar sigma star. So, what are the certificates we are going to give, that we should be able to give in polynomial time with respect to the a string x. So, that is what is the essentially the language is polynomially balanced, the meaning of polynomially balanced language is exactly this.

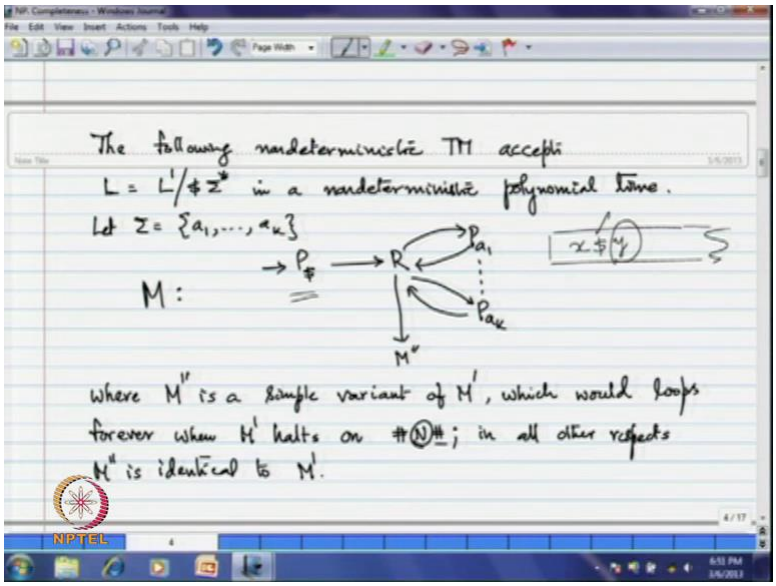
(Refer Slide Time: 09:36)



So, let me look at that, L dash in sigma star dollar sigma star is polynomially balanced, what is the meaning of that means there is a polynomial p, such that x dollar y is in L dash, then the length of y is less than equal to p of mod x, that means with respect to x, you know the length of y is in the polynomial bound. So, that is the polynomially balanced language.

Now, what is another point in the hypothesis, this L dash the polynomially balanced language L dash is in p that is also given to us. The meaning of this is there is a turing machine M dash that decides L in a polynomial time. Let me say that is q time, now what I want to look at that this L is in NP. So, to show L is in NP, I have to give a non deterministic machine, which accepts L in a polynomial time.

(Refer Slide Time: 10:37)



Now, the following non deterministic turing machine that accepts L, what is L, L dash by dollar sigma star in non deterministic polynomial time, we are constructing it here, how do we construct. You just take the tape, and the x is given as input to you from sigma star to accept L what I am going to do, first you print dollar here, so that is what is this p dollar, this component of the turing machine. And then here, non deterministically generate the possibilities of certificates.

So, this y it is from sigma star, so if I assume sigma is a 1 a 2 a k, I have the possibilities of you know printing a 1 or a 2 or and so on I have here k loops, printing the connecting to a printing machine, printing a 1 or a 2. So, whatever is the certificate, I mean what are is the string from sigma star, I should be able to generate following this loop, this particular portion of the this thing turing machine. And since this is non deterministic machine, you know whatever is you want, whatever is that you want you can print using this loops, and then once you print a certificate y you know you connect it to M double dash.

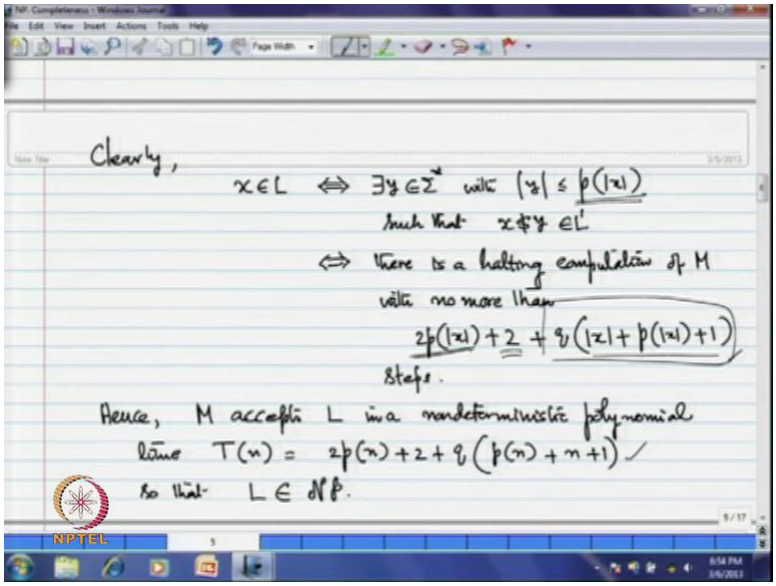
Now, you ask me what is M double dash, the M double dash here this is a simple variant of M dash, because this M dash you know, because L dash is in p, we have a turing machine M dash, which decides L dash and polynomial time. Now, what I am doing this M double dash is just a simply variant of M dash, which would loop forever, when M dash halts with no, because M dash is an algorithm you know one when you give an

input to that, it will say yes or no. If it whenever it is saying no, this variant M double dash will simply loops forever may it is not going to halt.

In all other respect, you know M double dash is just identical to M dash means, whenever it is halting by saying yes with all the aspects. So, we will just leave it that way, so that is what is M double dash. Now, you look at once again the turing machine M, so this M th turing machine what we are doing in the beginning, what are is input that you are taking x. We first print the dollar non deterministically, you will be able to print any string y from sigma star.

And then this input will be given to M double dash, aware M double dash is simply M dash, except that whenever this is saying no, it is not going to halt, it will go simply you know it will loop forever. So that, you know the no case will not be accepted, because we are constructing a non deterministic machine here.

(Refer Slide Time: 13:26)



So now, you look at this x is in L, if and only if, there exists y in sigma star with you know length of y is less than equal to p mod x, because it is such that this x dollar y is in L dash. Because, you see the language L is equal to L dash by dollar sigma star, since L is equal to L dash by dollar sigma star, whatever is x is in L, you know you will be, you will have some y with this kind of property. Now, you look at, because this x dollar y is in for some y this is in L dash, there is an halting computation of M with no more than this many steps.

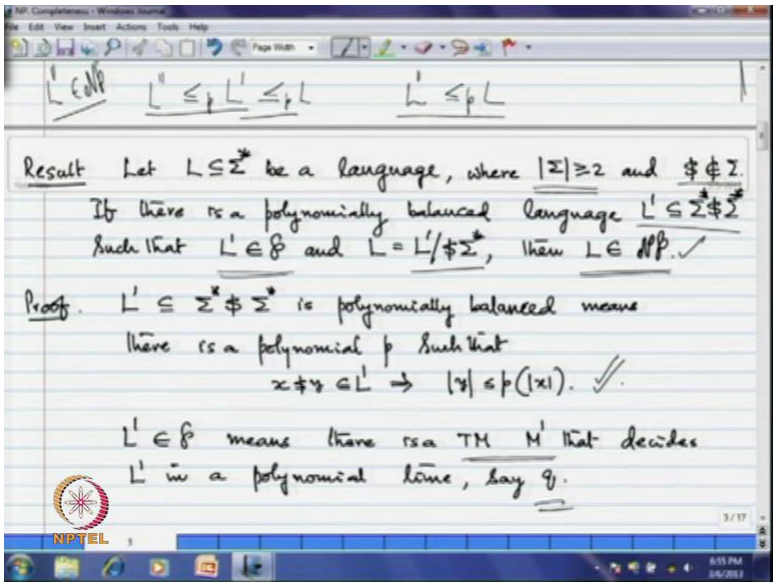
Now, you look at how many steps you would require, because to print dollar as the given position one step, and then take a right move, and whatever is y that you want to print, every time you will be taking a right move and print it, All right. And thus, see whatever the length of y you know it is bounded by p mod x. So, printing that many 2 p mod x this many printing steps for that, and then printing dollar is one step. And finally, you will be taking one more right move to go to this position, so printing dollar is one step, going to taking a right move one step here.

So that, now there are two steps here, printing y which is of length maximum p mod x, so that is 2 p mod x number of steps to print y, two steps for this purpose what I have mentioned dollar and taking a right move there. And once you have this input, since the machine M, M dash is taking the polynomial time q, and now the what is the input size here, this is mod y mod x plus 1, and mod y is less than equal to p mod x. So, this is the total, now input length for M dash, so now it is to M double dash, so that is the polynomial time with respect to q here.

So, this is the total number of steps that this machine takes, and you see this is the polynomial. Because p is a polynomial; q is a polynomial with respect to the length of x, what you are having this is you know in a polynomial time, hence M accepts L in a non deterministic polynomial time T n, where the function T is given by the expression 2 times p n Plus 2 plus q times p n Plus n Plus 1 this is the polynomial, you compare with this, and thus you can understand that L is in NP.

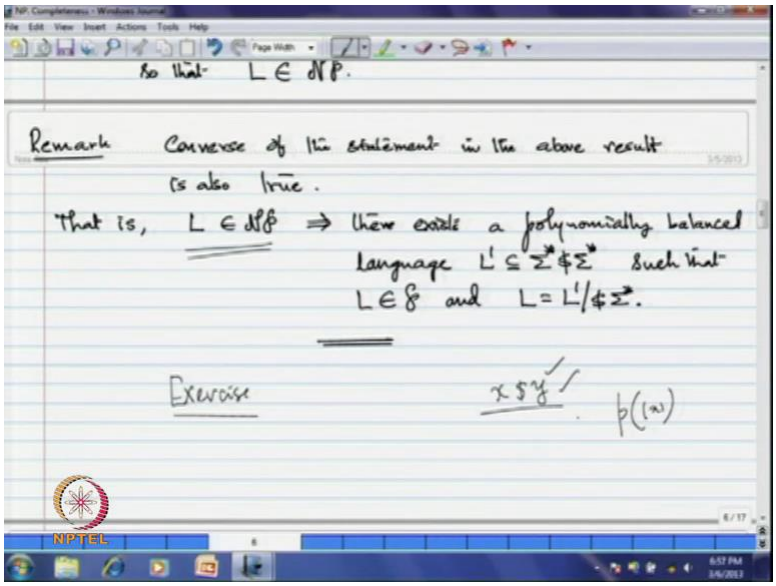
Once again we look at the statement, what I what we are trying to see here, to observe a language is in NP, if you can identify a polynomially balanced language is nothing else, but you know given a certificate given a certificate, which is which within polynomial time, you should be able to decide whether that you know is in p. So this, with this criterian, we can cross check the given language is in NP.

(Refer Slide Time: 16:45)



And where is this small remark, but of course, very interesting that converse of these statements in the above result is also true, what is the meaning of that, if you take a language L in NP, you will be able to identify a polynomially balanced language L dash, in which is a subset of sigma star dollar sigma star, such that L dash is in p. And the given language L which is in NP is equal to L dash by dollar sigma star. So, we will be able to identify a polynomially balanced language L dash always, so polynomially balanced language means the one which satisfies this criteria.

(Refer Slide Time: 17:13)



So, this is also true, that is this there exist a polynomially balanced language L dash, contained in sigma star dollar sigma star such that, L is in p and L is equal to L dash by dollar sigma star. You can try this as an exercise, because what is the language L dash that you wanted to identify. So, simple hint can be you know, what are the certificate corresponding to the given input, you have to append those certificates. Then you are creating this L dash, when it is in NP you know, you are non deterministically accepting it non deterministically accepting in a polynomial time.

Now, when you want to have a polynomially balanced language this L dash, how do we construct, next to the input x you have to give a certificate y, and this y length should not be more than you know for some polynomial p mod x, the length of x. So, that kind of language we have to identify. So, this is in fact, you can use the what is called the computation history of this machine the machine corresponding to L, you can make use of that and you can create the certificate, this is a hint here we can try this is an exercise.

Now, to prove to establish a language NP complete, you see we have two conditions. One is to observe that the language is in NP, and the other is you take any languages in NP you have to give a polynomial time reduction to the language, what you are targeting to show that is NP complete. For both the things I have mentioned like you know, what is the approach that we will be following, this is a necessary and sufficient condition

corresponding to NP to observe that language in NP. And thus more or less you know you can consider this technique, and understand that a language is in NP.

So, you have to essentially give certificate in a polynomial time, and verify in polynomial time, that the string is in language L dash or not. And second thing is, if you have already established some language is in a language is NP complete, then we know that you just give from that language to the targeted language, a polynomial time reduction these two things I have just mentioned. Now, what do we do any way first we have to establish some language is NP complete, then that kind of reduction just single reduction will be sufficient, how do we do that what is the approach.

Now, again you look back, when we are talking about undecidablity, we have first established that halting problem is undecidable. So, once we have established the halting problem is undecidable using that halting problem, we have reduced many other languages to show that they are undecidable language. So, in this course particularly, if you just look at, whenever we wanted to show some language is undecidable, either directly from halting problem or some variants of the halting problem or you know whatever that we have already established there undecidable from there you know, we started reducing the targeted problem.

And establish that they are undecidable, here also we will go in a similar approach what do we consider, and a variant of halting problem in the present context, what is the halting problem, halting problem ask you to verify I mean to decide whether given a turing machine M and an input string w, whether M halts on w, where there is an algorithm for that. So, that decision problem, you know we have observed that it is undecidable, All right that is an unsolvable problem.

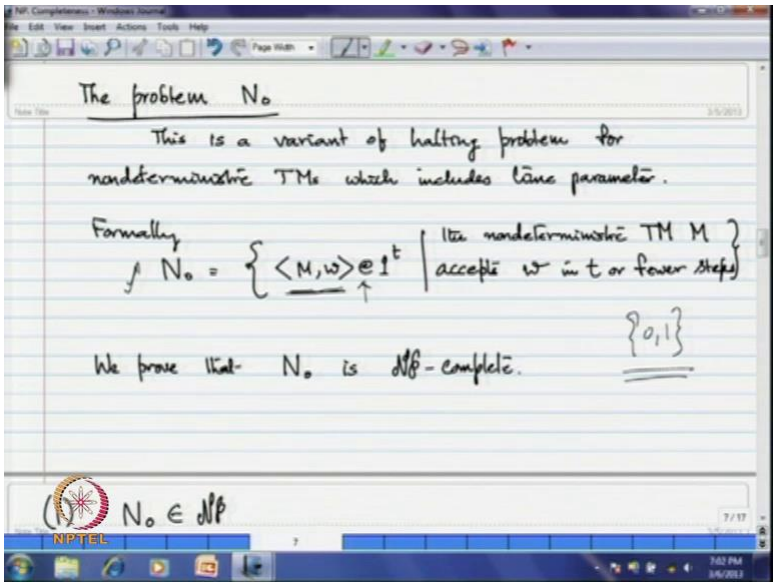
Now, a variant of that in the present context means, what do we consider in case of a turing machine, I consider a non deterministic turing machine. So, now you see, the system is going even more complex, and so the question is not you know given an non deterministic turing machine input string w, whether M halts on w it is not the question, because even if considering a standard turing machine you know the problem is undecidable.

So, when I am looking at this, of course I ask you, I give you some other parameter, there what is the time, the what is that time parameter that means I ask you to see,

whether this M accepts w the non deterministic machine M accepts w in so many steps, I give you a fixed number of steps. And the corresponding to give a turing machine and a string, and I ask you this question.

Now, this is the first language this is I can call it as a non deterministic quantitative analogous problem to the halting problem. So, such a variant we consider, and I will establish first that this problem is NP complete, to when what are the definition we give. So, we will verify both the conditions and establish that this particular problem is NP complete, for halting problem we would have written h h naught or whatever, halting problems similarly, we are since it is non deterministic machine.

(Refer Slide Time: 22:15)



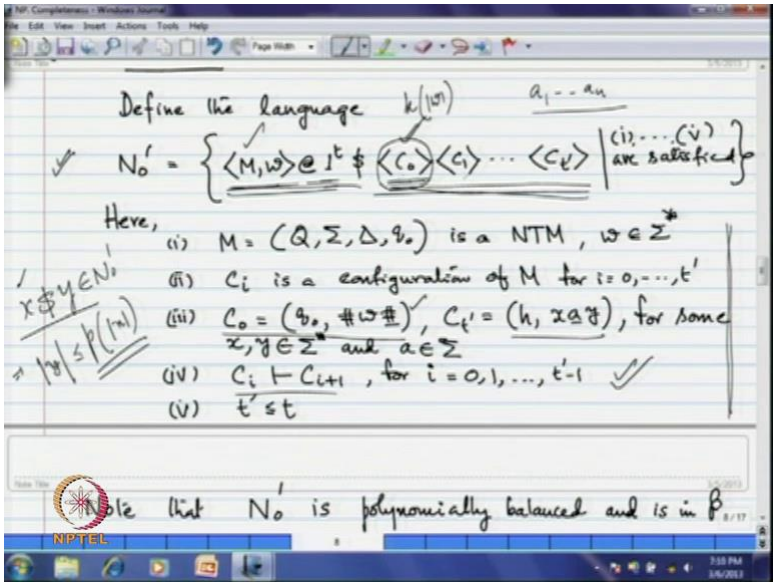
Let me start writing N 0 the problem, let us look at this problem this a variant of halting problem for non deterministic turing machine, which includes time parameter. Formally, the language, when I am looking at you see we are considering a non deterministic machine M, and a string input string w, of course we are considering the encoding of that, because I am giving you a language. And then of course this ambrasend symbol that at the right 1 t, that means you know this is the time that this many steps.

Because we are giving this in the unary representation, you know that M and w, we would have given that using the alphabet 0 and 1; you know that we have considered an encoding of a turing machine using a sequence of zeros and ones. Similarly, a non

deterministic machine also, we just encode it the give the input and M, so this is encoding of that.

And then this separator tells you after this what are the, so many ones I am giving, that is the number of steps that I am expecting, that M should accept w in that many steps All right. So, this problem N 0 is what, given a non deterministic turing machine M on a string w, whether M accepts w in t or fewer steps. So formally, we write a language concerning that is this way.

(Refer Slide Time: 23:44)



So, we prove that this language is NP complete, to show the language is NP complete N 0 I give you a polynomially balanced language N 0 dash, and observe that this is in p, and moreover this N 0 dash quotient with dollar sigma star, whatever is the underlying alphabet of course is N 0. So, that is how we consider this N 0 dash, we look at this how do we defined. So, define the language N 0 prime, N 0 is what is that, what we are going to give this M w that t the time of course, 1 power t we write that means so many ones, but there with respect to time.

Now, the dollar is a separator for the certificate, and what is the certificate we are I am going to give here is, encoded configurations of M and w you look at. So, all this C 0 C 1 C t dash is a this string satisfies, you know the following five as statements what is that, M is an NTM non deterministic turing machine Q sigma delta q naught and w is their

input. Now, what are these c i (s), c i (s) are configurations of M, so all these c i (s) and C 0 is the initial configuration of M with w as input, that is what is written this way.

So, q naught blank w blank, so this is what our initial configuration, and this C t dash is a halting configuration, that means you are in a halting you are in the halting state h, and with something on the tape with x a y for some x and y in sigma star a in sigma, the current reading symbol is a. And more over, what is this sequence C i, C 0, C 1 and so on, when from C 0 in one step you are getting c 1, from c 1 you are getting in one step C 2 and so on. So, c i gives C i plus 1 in M in one step for all these.

And then the number of steps here the number of configurations that I am writing it here, the number of steps here the t dash is less than equal to t, whatever is that number t we are given. So, let us consider this language, so you look at once again, what we are what is the certificate we are going to give, we are giving the computation history of M on w, and we are considering all possible configurations, which leads to a halting configuration from the initial configuration within you know t steps, that is what we are considering in this language.

This exactly gives you the hint of the exercise I have I had given you, this kind of thing you consider, and then you can observe that for the converse of the previous result. Now, you look at this N 0 dash is the polynomially balanced language and is in p, how do we say that, why this is polynomially balanced, because any string you consider here as of the form x dollar y is in this N 0 dash, what is x here it this encoding of M and w at the rate 1 t, so that is what is x.

Now, x dollar y is in N dash, then what do we require what is the condition, the length of y should be less than equal to some polynomial time by the polynomial of the length of x. Now, you look at this configurations, how many configurations we have, t number of configurations, how I am and go going to encode the configurations, you encode appropriately like you know the turing machine the transitions are un coded as blocks of zeros and ones.

And similarly, you know in a configuration, you know that there is a state component, and the input that is what is essentially a configuration, and always the input w it changes what, say for example w a 1 a 2 a n in each step, what increment of at most once a symbol, like if I go towards and I may increase by one more symbol, if it is the case. So,

in each step either you are going to take a right move, you know or making a printing step, so each step with respect to w, you know I am not making any larger change as where as the configuration this thing concerned the length of the take concerned.

And then you see this transitions from with respect to this, so a state component is there if you look at a configuration, the state component is there and the input. So, w is a input, and the change what we are going to have, whatever is the way that you encode each symbol, you know this is with respect to the w. This C 0 is in a within a polynomial time, with respect to what is called the length of x, because the entire thing as x we have.

And how many such configurations that we are going to give? We are going to give configurations which are less than equal to t the number of configurations. So, since this parameter t is part of x, you see this configuration let me say, each configuration with respect to w is of maximum, say k length. And now, you have k t dash this and once again deemphasizing that this configuration with respect to w, I would have may be, let me put a parameter this is k mod w.

Suppose, this is how when we have encoded which is coming up, and then since t is already a parameter inside x you see, I am taking t dash steps, so this is also with respect to that a polynomial times. And thus, whatever is the certificate we are going to give here, that means x dollar y is in N 0 dash, then the length of y what we are going to give the certificate, it is less than equal to you know with respect to the input that x, it is a polynomial I would say polynomial bound with respect to the parameter mod x.

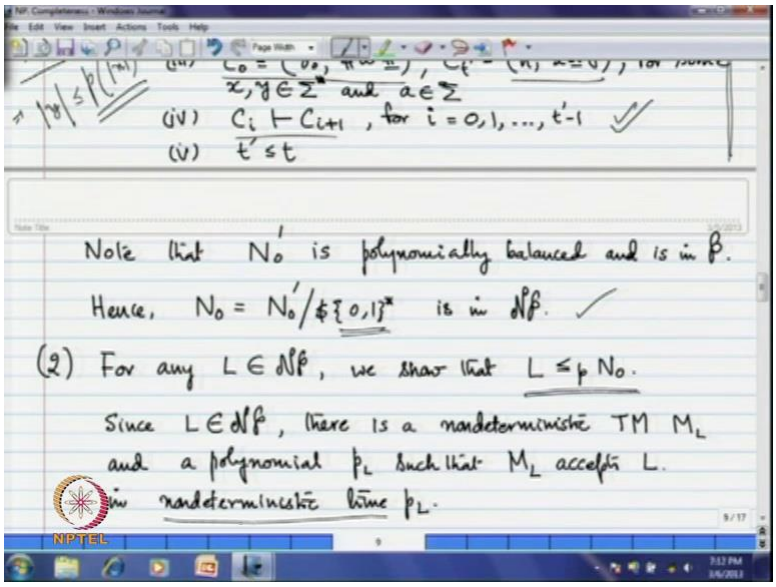
So, this is therefore polynomially balanced and it is in t also, why this is in t, because everything is available now, only thing what I have to verify you know once I have printed this certificate, that this is a non deterministic turing machine w is in sigma star. So, these are the things that you have to look at, and then since these are the configurations of N, you have to just cross check that once this C 0 is of this form, this C 0 whether you are gain to get C 1 or not, how do we know this that is from the machine M, because the transitions of M are also available.

So, C 0 to C 1 whether you are going to get or not we to have see C 1 to C 2 you are getting or not you have to see and so on C t dash, this is the step that you have to verify. So, on this input when I am going from C 0 to C 1, I have to carefully verify whether this is following the transitions of M or not, that is what I have to cross check. And if this C 0

is the initial configuration and C t dash is the halting configuration that way, and the number of such configurations whether they are less than equal to t or not.

So, all these conditions that we have to verify, for x dollar y the string which is in N 0 dash, so you can do this in polynomial time, because this is just we have to go back and forth on this and that how many times, because the number of transitions that you have to verify here.

(Refer Slide Time: 31:51)

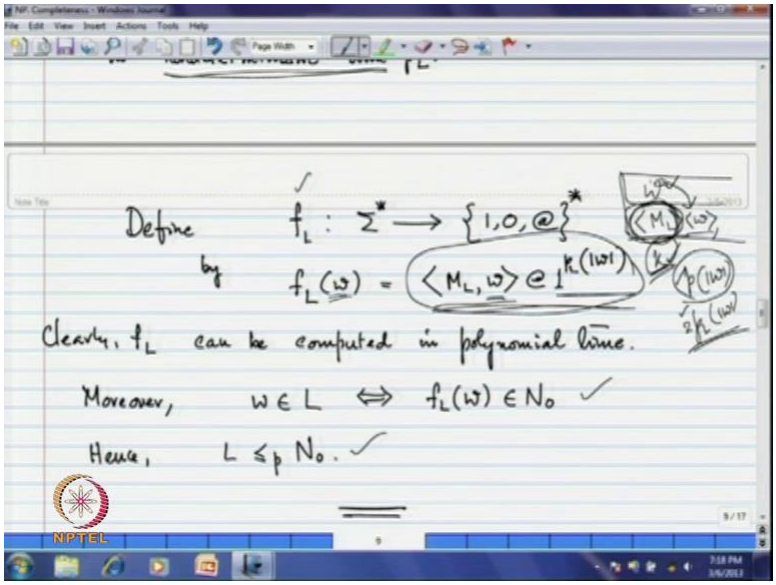


So, you can observe carefully, you can give little more details and observe this N 0 dash is the polynomially balanced language and is in p. And moreover here, the way that we have constructed is N 0 is N 0 prime by dollar 0 1 star, and writing 0 1 star here with the assumption that, and encoding these configurations using the sequence of zeros and ones again, like the way we have encoded turing machine. So, dollar 0 1 star that is what is N 0. So, this will be in p, because once there is a polynomially balanced language in p, I will use that result, and observe that this N 0 is the language in NP All right. So, the language N 0, we have observed that this is in NP using that result.

Now, the second thing what have to observe, you take any language in NP, I have to give a polynomial time reduction to N 0 that means I have to show this L is less than equal to p N 0. Let us see, how do we do that, since L is in NP there is a non deterministic turing machine M L, and a polynomial p L such that M L accepts L this is a definition of NP language, All right excepts L. Of course, in non deterministic time p L, because the

polynomial p L for an non deterministic machine, we say this is a non deterministic time, so this is a definition.

(Refer Slide Time: 33:34)



Now, what do I do, I give a polynomial time reduction here, so reduction means what I have to do, I have to give a turing computable function that reduces L to N 0, I to give a turing computable function which reduces L to N 0. And that computable function, we should be that reduction should happen in a polynomial time, these two things that I have to look at, how do we do that. I define the function f L from sigma star to 0 1 at the rate star, because the encoding of turing machines that we are doing with 0 and 1.

And anyway we are separating the time with respect to this at the rate symbol. So, therefore these three are taken into count, All right. And now, how do I define that, you take any string w and sigma star, first I simply print M L, because M L is in my hand. So, for L there is corresponding non deterministic turing machine M L, so there is a constant that is a with me. So, I print M L encoding of that and whatever is the w that you are giving me, the same w you know encoded. So, this ((Refer Time: 34:51)) we consider encoded.

And then what do the time I give associate to this is, because this M L accepts w in non deterministic polynomial time p L, I just give that much time the time here is p L of mod w, so this is the time we are associating to this. Now, let us look at this, f L can be prepare this, so given w as input to a turing machine the turing machine you give, can we

create such an output in polynomial time with respect to the length of w. You look at, what we have to do here, what I have to essentially do is I have to print the encode of M L.

You can consider may be two tape turing machine, further time being say I will just print this is a constant one, I have printed encoding of this, then the w encoding you will also print, because w is available here. So now, this M L for every string this is always constant, this w corresponding to that I have to print. So, this is say for example, printing M L for any w, say for example k number of steps. And now, this w you encode it for example, you know depending on the type of coding that you have for each symbol that is with respect to length of w.

You will have some polynomial; say some polynomial time length of w that is how you will be coding this. And after this what I have to do, I have to print so many ones, which are you know how many ones I have to print p L mod w, this what I have to print. You see clearly p L is a polynomial, I have to print that many ones, so that means, printing that many ones means twice p L mod w steps, you would require. So, some constant number of steps to print the encoding of M L, and then some polynomial you know bound with respect to mod w to print the encoding of w.

And thereafter I have to print at the rate symbol that means, again two steps there and then to print this many ones I have to take these many steps. You look at, this is a polynomial p L is a polynomial, p is a polynomial, this k is a constant. So thus, you can see, this can be done in polynomial time, and this you can manage using a turing machine, what is the meaning of that this f L the function f L, given w on the tape of a turing machine.

You know, you creating this f L w that you can manage using a turing machine, in polynomial time, that means this f L is a turing computable function and can be computed in a polynomial time. So, now you observe the w is in L, if and only if, this f L w is in N 0, so string w is in L then f L w is in N 0, because there this will be accepting that is what is exactly this. Since, L is in NP there is a non deterministic turing machine M L, you know that accepts L in non deterministic polynomial time p L.

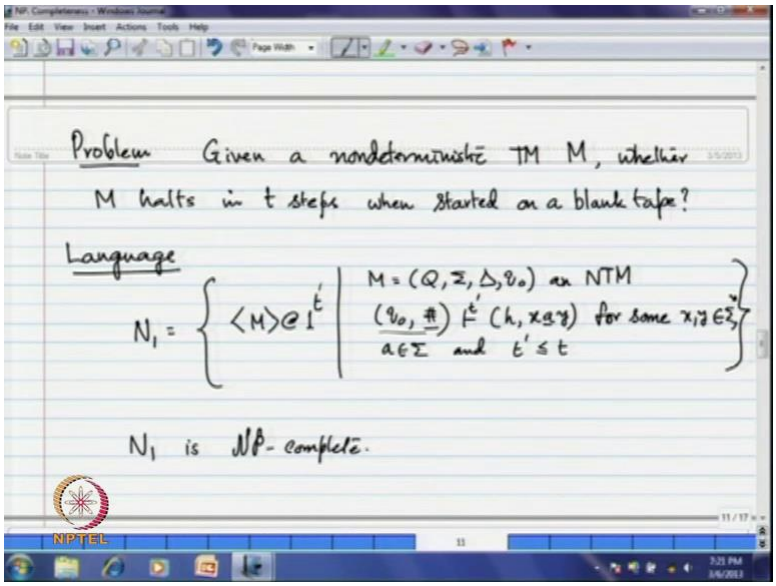
So, this f L w is in N 0, and whatever is that we have created here, if this is in N 0 then w has to be, you know w is accepted by M L in non deterministic time p L. So, w is in L, so

we have this situation; that means what, this is a turing computable function. And you are computing this in polynomial time, moreover this condition is also satisfied that means this f L is a polynomial time reduction from L to N 0, that means L is less than equal to p N 0.

Now, what is L? L is an arbitrary language in NP. So, you take any arbitrary language in NP, this procedure there is a very elegant simple procedure, you see that can be that L can be reduced to N 0 in polynomial time, so what is that. So, this is the second condition for NP complete to establish a language is NP complete, so combining is to, now we can conclude that N 0 is NP complete. So, if you look at the way that we have handle this undecidablity some in certain language are undecidable, we have started with halting problem.

Now, analogous problem of that halting problem that we have introduced and established, that it is NP complete. Now, as an exercise, what we have done that certain variance of halting problem that we have considered. So, those variance that you could establish that they are undecidable, how did we do that, so halt from halting problem, you have reduced to those variants. The halting problem, you give a polynomial, you give a reduction to the targeted problem that variant, here what we are going do, that reduction we have to verify that it in polynomial time. So, this polynomial time reduction is required here.

(Refer Slide Time: 40:13)



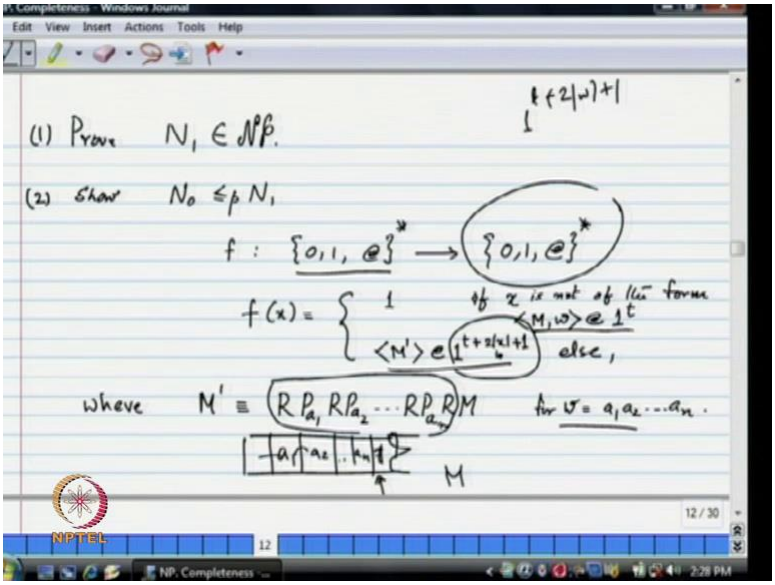
So, now I give you such similar exercise, we look at given an non deterministic turing machine M, whether M halts in t steps, when started on a blank tape. So, this kind of problem we have considered in undecidability also, given a turing machine M whether M halts, when you start that in a blank tape. Now, an analogous problem here, what is that, I give a non deterministic turing machine M, and of course a time parameter also. So, it say t in t steps, whether you know M halts, when you start that machine you know blank tape.

How do I write the corresponding language, say let me called it as N 1, N coding of M and the time and separating that there its symbol. So, this is the string, where M is an non deterministic machine and when you start this on the blank tape, that means this is initial configuration. And in t s there steps, it will halt in something on the tape that x a y for some x and y in sigma star a is sigma, and this t this t dash should be less than equal to, so within t steps. So, I am this writing t dash within t steps, All right.

We can observe that, again N 1 is also NP complete, you see if you look at the history of NP complete problem; no there are very important problem several you know, decision problems concerning certain optimization problem. Although since, that we are going to discuss, but you look at for the classroom concerned, we look at such a nice simple languages that anyone can sit and establish that they are NP complete.

And they approach is also, you know what you have practiced, the reductions in undecidability. So, what I what you have to do here, first we have to establish that N 1 is in NP, that is in NP language. And then now since we are already knowing that N 0 is NP complete, you can make use of you know that N 0 ((Refer Time: 42:22)) now it is sufficient to give a polynomial time reductional say from N 0 to N 1.

(Refer Slide Time: 42:41)



So, you can target to do this, thus an exercise to establish that this is NP complete, regarding polynomial time reduction from N 0 to N 1; I give you a small hint here. So that, you know you can complete this problem by showing that there is a polynomial time reduction from N 0 to N 1, so that N 1 is NP complete, what do I suggest you look at in N 0, what do we have in N 0 we have you know encoding of non deterministic machine and a string. And then you know the respective number of steps within that you wanted to pursue.

So and corresponding to this, what we have to assign? We have to assign a non deterministic turing machine, which when you start on empty tape it has to finalize, whether this is N 1 within the given number of steps, what do we suggest here. You take a string, because the encodings I am just assuming, you will be able to make with 0 1 here, because as earlier in the turing machines what we are writing this at the rate symbol is extra that we are using, just to separate between the turing machine code and the time.

So now, you take a string x in this 0 1 at the rate this star take any x, if that is of the form some non deterministic turing machine M, and string w input string w, and sometime you said the rate un powered t. If this is not of this form, then you simply assign somes and some the string which is not in you know in N 1 form this. In N 1, we know that at least at one once at the rate symbol has to come, when I he assign someone or whatever, it is not clearly in N 1. So, what I am doing, if a string if a string x is not of this form that M w at the rate 1 power t, then I will simply assign some string which cannot be in N 1.

And if that is of the form, then what do we do some for some non deterministic turing machine M w all these, I will assign the string, I will tell you how what is this string, just what do I suggest. You construct, you take it turing machine non deterministic turing machine M dash, what we what is our desired intention. Because, when you start on empty tape then given number of steps, whether this will halt or not that is what is our this thing question.

So, you simply from the current cell, because you are starting on the blank tape, so the current cell, you take a right move and print a 1 here, and take a right move print a 2 here and so on a n, what is this a 1, a 2, a n that is the string w, because this is x is the form M w form, M w at the rate 1 power t what do I suggest. First you print this string a 1, a 2, a n and take a right move, so now the reading and writing at position is this, so this part can be managed by this portion of this M dash, and then simply you connect to the machine M.

So, if you consider this non deterministic turing machine M dash, what exactly we are doing, whatever is the part of input w that you will first prepare it on the tape, and then connect it to M. Now, to prepare the string, how much time you would require the length of x length of this w of course, this is w. And you know plus one more step essentially, so 2 mod w plus 1 steps that this will take, you 2 mod w plus 1 steps this will take to prepare this a 1, a 2, a n. And then when you start M here this will pursue this input within this t time or not, that is what we have to cause.

So, the total time we are giving here is 1 t plus 2 mod w plus 1 time. Suppose, if you give this, now you can observe that whenever this is in, whenever this M w at the rate on 1 t is in N 0, if and only if you know the f of x. So, whenever x is that means x is in N 0, if and only if, you can see f of x is in N 1. And you can observe that these reduction is in

polynomial time, because you can look at with respect to input size, how much this is coming N. So, with this hint you know with this kind of function, I can ask you to complete this exercise that to show that N 1 is NP complete.