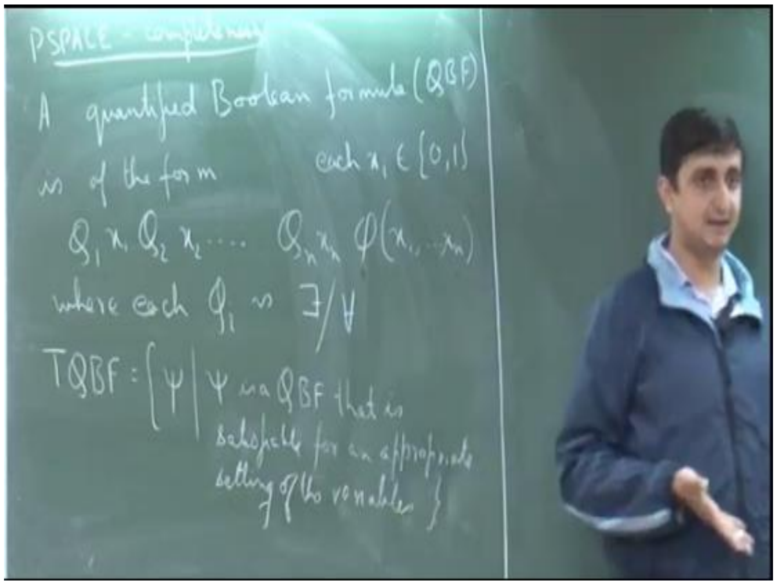
Computational Complexity Theory

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

Lecture -10
Introduction

So, let us get started last time we were discussing the polynomial hierarchy. So, I will get into the discussion in a few minutes but before that I just want to digress a little bit and talk about something else. We talked about space bounded complexity classes and we saw the class PSPACE. Let us look at a complete problem for PSPACE. I will not prove the completeness part in our lecture but I will motivate as to why it is a complete problem and if you are interested in the proof you can always go back and read it is there in your text.

(**Refer Slide Time: 01:04)**



PSPACE - completeness

A quantified Boolean formula (QBF) is of the form $\quad$ each $x_i \in \{0,1\}$

$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \, \varphi(x_1, \dots x_n)$

where each Q_i is $\exists / \forall$

TQBF: $\{\psi \mid \psi$ is a QBF that is satisfiable for an appropriate setting of the variables $\}$

So, we have seen, what a Boolean formula is and what does it mean to say that a Boolean formula is satisfiable. Suppose if we have a Boolean formula phi, we say it is satisfiable, if there exist an assignment to the variables of that formula which makes the formula evaluate to true or 1 however you want to think of it. So, suppose what we are saying is there exist an assignment to the variable.

Suppose the formula has n variables what I am saying is that each of those variables can take a value 0 or 1. Instead of quantifying each variable with existential quantifier I can generalize this idea and talk about variables which are bounded by the universal quantifier. So, instead of saying that a variable can take any value between 0 and 1, I say that well the formula has to be true for all possible values that the variable can take.

Whether it is 0 or 1 I do not care it has to evaluate to true. I mean a very simple example is suppose if you look at the probably the most simplest formula let us say just a single variable x 1. This formula means if you think of this as a formula is always satisfiable. Because if you set x 1 to 1, it gets evaluated to 1. But if I wants this formula to be satisfiable for all possible values of x 1 then we see that phi is not satisfiable because if I assign 0 to x 1 clearly it is not satisfiable. So, this is what is the distinction between these two ideas.

Let us try to encompass this idea into a formal definition. So, a quantified Boolean formula which in short, we will denote as QBF is of the form some Q 1 x 1. We will assume that the formula in n variables Q 2 x 2 so on till Q n x n such that we have some formula phi on n variables where each Q 1 is either an existential quantifier or a universal quantifier. When do we say that this formula evaluates to true if for the appropriate setting of the Q i.

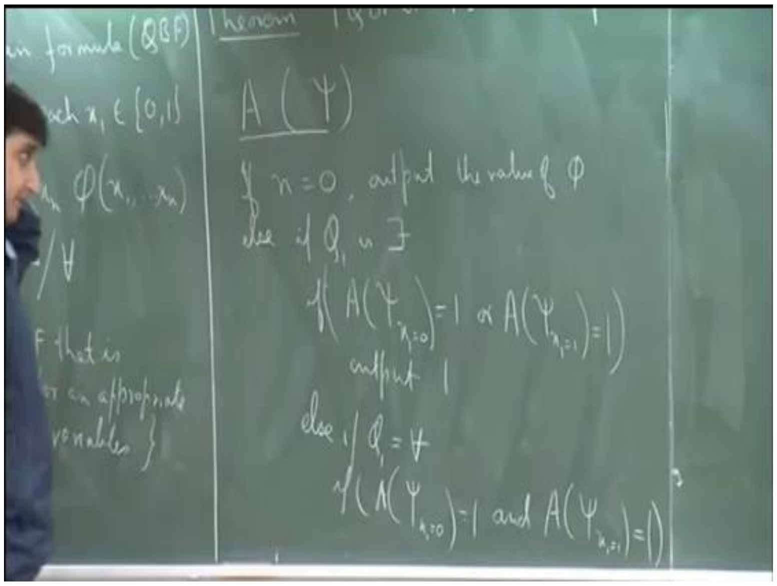
For example, if Q 1 is an existential quantifier then I allow x 1 to take any value 0 or 1. For any setting of the value I will consider the formula phi. If Q 2 is an universal quantifier, then I want this formula to be satisfiable for all possible values of x 2, that is whether it is 0 or 1. So, this is what I mean by a quantified Boolean formula. And the language TQBF so, T stands for true is the set of all quantified Boolean formulas.

Let us call them psi such that psi is a QBF that evaluates to or I should not say evaluates that is satisfiable for an appropriate setting of the variables. I mean quantifiers are already fixed but depending on the quantifiers I am giving values to the variables that is what I mean here. So, if it is a there exist even if it is an existential quantifier it can take any value and if it is a universal quantifier it should be true for all values.

That is what I mean. Universal quantifier 0, 1 they are all Boolean variables. So, we are looking at Boolean formulas. So, each x I and take values between 0 and 1. And we can also assume in addition that phi is a formula in 3 CNF. It is a conjunctive normal formula with three literals per clause. That does not make a difference, is this language clear to everybody? So, clearly, we can see that this is a generalization of the 3 psi language.

Because if I set all my quantifiers to existential quantifier that will basically mean that this is nothing but just the language psi.

(Refer Slide Time: 08:20)



```
A ( $\Psi$ )
if $n = 0$, output the value of $\phi$
else if $Q_1$ is $\exists$
        if ( $A(\Psi_{x_1=0}) = 1$ or $A(\Psi_{x_1=1}) = 1$ )
            output 1
    else if $Q_1 = \forall$
        if ( $A(\Psi_{x_1=0}) = 1$ and $A(\Psi_{x_1=1}) = 1$ )
```

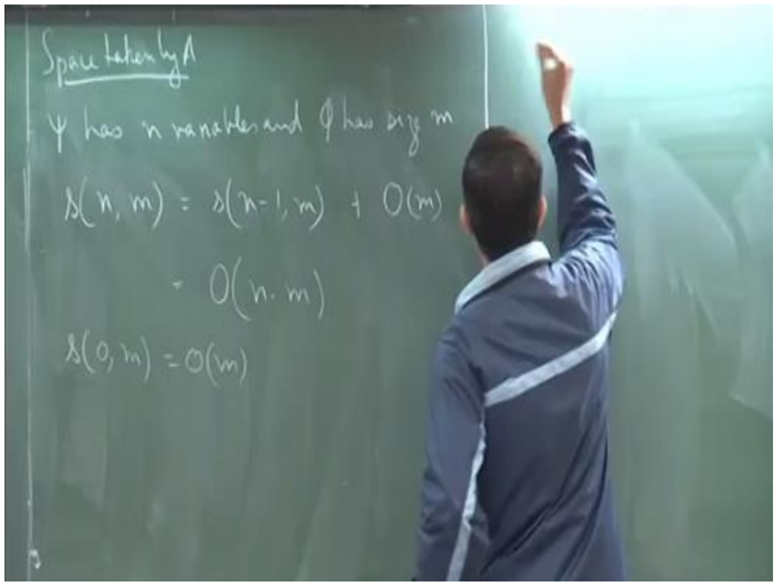
So, the theorem is that TQBF is PSPACE-complete. I will just discuss the containment part that why is TQBF contained in PSPACE in other words we need to give an algorithm. So, how do you give an algorithm for this language a PSPACE algorithm? That is one way to look at it. Let us give a recursive algorithm and you can also give an iterative algorithm but probably that will be more difficult to construct because it will be an exponential time algorithm.

So, let us give a recursive algorithm, let me call this algorithm A, which takes sum QBF formula phi and then tries to answer yes or no depending on whether phi is satisfiable or not. If n is equal to 0 that is it does if phi does not have any variables, so we are also considering the special case when, I mean a formula consists only of constants 0's and 1's. So, it says and an or of constants in which case it is easy to evaluate.

So, then just output the value of, whatever value you get by evaluating phi it is just constants. Else that is if n is greater than 0, if n is greater than 0 then we know that it has at least one variable. There is a quantifier associated with that variable if Q 1 is an existential quantifier what we do is we check the following two formulas. That is, we check the formula phi with x 1 set getting the value 0.

So, if A of psi with x 1 set to 0 = 1. We can just say that also because this is just some value. This is 1 or A phi x 1 = 1 is 1. So, either of these 2 return A 1 then we output 1. So, now we again go back to this if else if Q 1 is a universal quantifier then we check if A phi x 1=0 is 1 and A phi x 1=1 is also a 1. It is a very standard intuitive algorithm. The point is how much space does this take?

(Refer Slide Time: 13:09)



Space taken by A

ψ has n variables and ϕ has size m

$$S(n, m) = S(n-1, m) + O(m)$$
$$= O(n \cdot m)$$
$$S(0, m) = O(m)$$

So, the thing again is that space can be reused so when computing phi and I am making recursive call to phi with x 1 set as 0 and phi with x 1 set as 1. I can reuse the space in both these recursive calls. Let us say that for a formula with n variables and of size m. So, suppose phi has n variables psi and phi has size m then the space taken with n, m as parameters is the space required for n-1, m plus sum constant times m.

Because we need this much amount of space to basically write out this expression on the tape and then run the algorithm on it. Because basically what we are doing in this algorithm is that we need to write out what the new formula would be on which I am running my recursive calls. So, for that I need some constant times m space. Basically, if you look at one iteration how much space am, I requiring here.

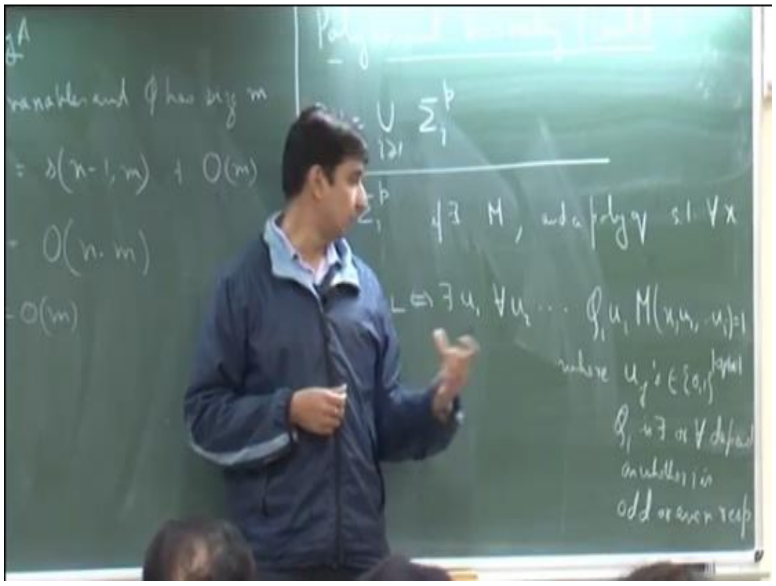
So, to make a recursive call to A this is the amount of space I am requiring and the same space can be reused here also. The only thing that I need to remember what is the bit that this guy had output. So, that is the only thing that I need to remember. But then I need to so to be able to run a on a particular formula I need to know what that formula is. So, that is why I am just assuming that I write out that particular formula on the tape and that requires some m space.

I guess you can make this some constant. You have this thing globally written on your input tape you can always refer to that and probably you can replace this with some constant. So, if you solve this what you will get is something like order n times m but what you are saying so then you have to look at what the base case is so the base case is when you have 0 variables so, s of 0, m will require some size m.

So, what you are saying then will require n + m. But yet does not quite matter because we have sufficient space. The hardness part also can be shown again not with much difficulty its basically again looking at the configuration graph and then looking at adjacent configurations and coming up with a formula that encodes the definition of an edge between adjacent configuration. So, you write a formula which is 1 if there is an edge between two pair of configurations and which is 0 otherwise and you appropriately encode it into a thing.

I will not go into that part so let us come back to our discussion of polynomial hierarchy and let us see how this thing fits in.

(Refer Slide Time: 17:40)



So, what was the definition that we saw last time? It is the union over all i greater than or equal to 1 of the classes sigma i p. And what was the definition of sigma i p? It is basically a generalization of the class n p where instead of looking at just one the existence of just one certificate you look at an alternating manner whether there exist different certificates with the universal and the exit existential quantifier.

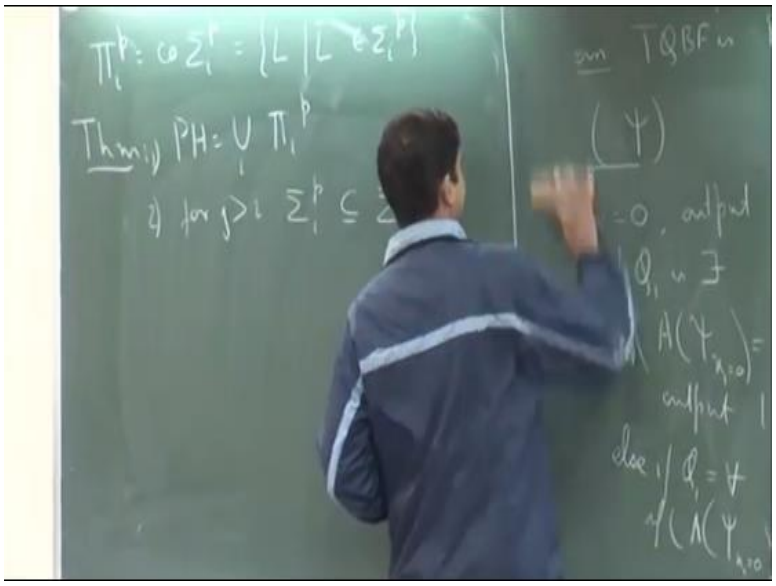
Let us just write that definition down once more. So, L is in sigma i p. If there exist a machine M and a polynomial Q such that for all strings x, x belongs to L if and only if there exist u 1 such that for all u 2 so on till some quantifier Q I u I. The machine M with x, u 1, u I evaluates to 1. Where all the uj's have length Q of mod x and Q i is an existential or universal quantifier depending on whether I is ordered even, so this was the definition.

And one of you asked last time that why do we look at alternating quantifiers? I mean what if we have the same quantifier for 2 consecutive strings? Can anyone answer that? Exactly so I mean if we have two strings with the same quantifier 2 consecutive strings then basically, we can just treat both those strings as just a single certificate with that particular quantifier. Let us say if I had a there exist u 1 and there exist u 2.

I can think it off as the concatenation of u 1 and u 2 with the existential quantifier. So, what is the blow up? The blow up is only in the length of the string but then I mean we have that freedom then we can choose a polynomial Q appropriately which is just twice the length of this polynomial. We do not care what that polynomial is as long as it is a polynomial. So, a language is in this class if there is some machine and some polynomial.

It is just that we will have a new Q prime and maybe a new machine which will deal with that. And the same thing happens if I have a consecutive universal quantifier as well. And it is not only for 2 if I have any sequence of quantifiers which are the same, I can always club those strings together and put the quantifier preceding it. So, that is why always without loss of generality we can assume that the quantifiers alternate. That changes the definition of the class. So, we will look at that so similar to sigma i p.

(Refer Slide Time: 23:06)



We define the class pi i p which is just a complement of sigma i p, that is it contains the set of all languages L such that L complement is in sigma i p. And now I think that answers your question. So, if we look at the class pi i p. We can also now since I have defined pi i p as the complement of sigma i p I can state this as a theorem that a language will belong to pi i p, if there is some polynomial time turing machine M and a polynomial Q such that for all x, x will belong to the language.

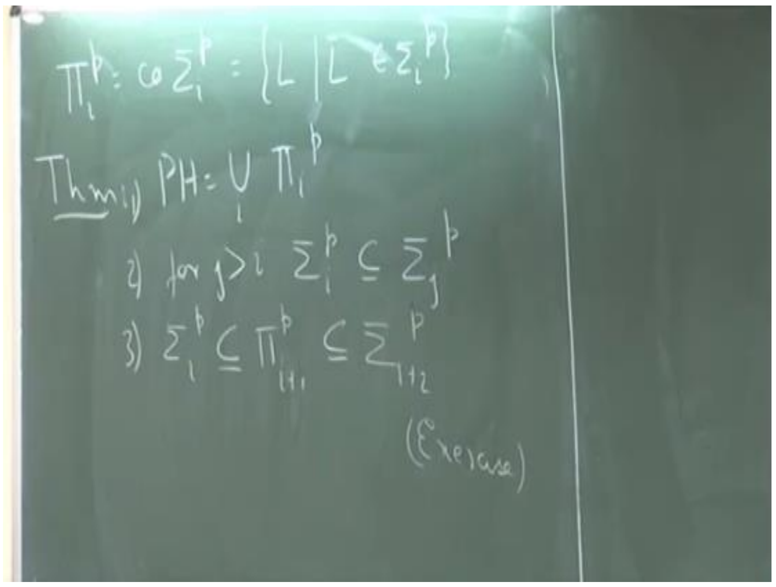
If and only if for all u 1 there exist u 2 and basically again the quantifiers will not or they will alternate and they will be i such quantifications and the machine accept the string with all those certificates. The difference is that what is the initial quantifier that I am starting off with. So, therefore again we can define pi h also in the class PH also alternatively with respect to the classes pi i ps.

So, it is the union over all i’s of pi i p, and again these are very easy to see also what can we say about the various sigma i's and the various pi i's? What can we say between let us say 2 sigma let us say you have a sigma i p and a sigma j p? For j greater than i, sigma i p is contained in sigma j p. This follows from the definition. Because if I, have a higher j I can just ignore all the additional strings.

So, that we do not know, we know that there is a containment but whether it contains any additional languages that is something again that we do not know. Again, that is a good question and people believe that all these containments are actually proper. People believe that within the polynomial hierarchy, so basically these are referred to as levels of the polynomial hierarchy. They believe that all these levels are actually different from each other.

There exist problems in each of these levels which are not contained in smaller levels, but that is something which is not known. We will actually come to that so there are complete problems known for each of these levels and actually that is not something very easy to see I can just state that if you just look at a generalization of sat again. So, what was sat? Basically, SAT was having certain variables and you look at does there exist values to those variables which makes it satisfiable. But instead of that if you just generalize it. Let we write this here.

(Refer Slide Time: 27:44)



$\Pi_i^p = co\Sigma_i^p = \{L \mid \bar{L} \in \Sigma_i^p\}$

Thm: 1) $PH = \bigcup_i \Pi_i^p$

2) for $j > i$ $\Sigma_i^p \subseteq \Sigma_j^p$

3) $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$

(Exercise)

So, we can define the following language sigma i SAT is the, it is a set of all formula's phi. Such that there exists some vector u 1 such that for all u 2 and so on some quantifier Q i with a vector u i such that the formula phi with all these vectors u 1 through u i is satisfiable. So, I can treat the variables in this formula as some set of disjoint vectors where there exists some assignment to the variables in u 1 such that for all possible values that can be assigned to the variables in ~~U2~~ u 2 and so on all the way up to u i this formula is satisfiable.

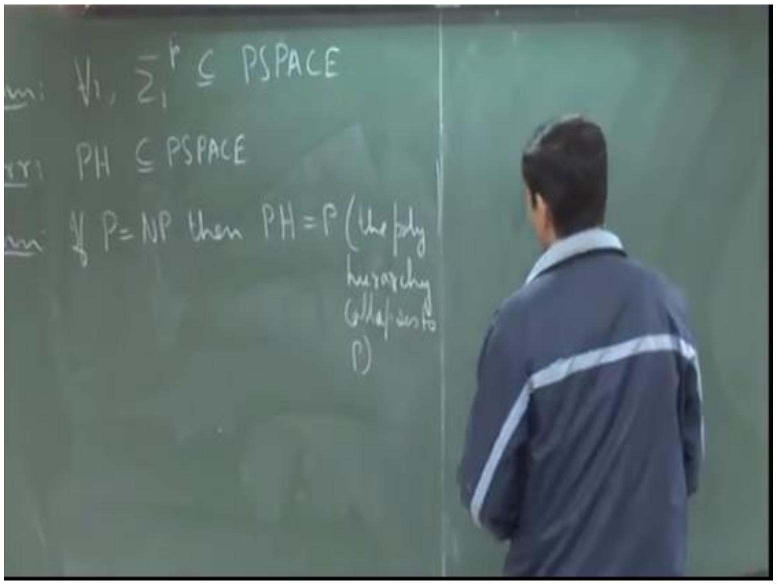
And this is complete for sigma i p. I think this is actually an exercise in your textbook to prove why this is complete. So, this is again not difficult to show but this is something which is kind of derived from the definition of sigma i p whether there exist other natural problems which exactly characterize these classes or not, that is something interesting and up to the second level of the polynomial hierarchy there are complete problems which are known.

There are other natural problems which are known to exactly characterize for example sigma 2 p. So, sigma 1 p is nothing but n p right for sigma 2 p they are unknown but for sigma 3 and 4 you can come up with artificial definitions but I do not know what exactly what you want. So, just coming back to this thing, what about let us say the relation between a sigma i p and a pi i p what can we say about these two classes.

Maybe for a different type; maybe let us say we if have some j. So, pi j p is call sigma j p. In other words what we can write is for any actually i + 1 if look at just pi i +1 p. So, this contains sigma i p. Because again what we can do is we can just forget the first quantifier. We have some other, still i quantifiers left which begin with the opposite quantification. And in the same manner this is contained in sigma i + 2 p.

I am stating these theorems but you guys should go and just work this out and convince yourselves. So, they do follow from the definitions but it is good to actually work them out. So, the reason why I stated why I digressed and talked about this PSPACE complete problem is that.

(Refer Slide Time: 33:01)



$\forall i, \Sigma_i^p \subseteq$ PSPACE

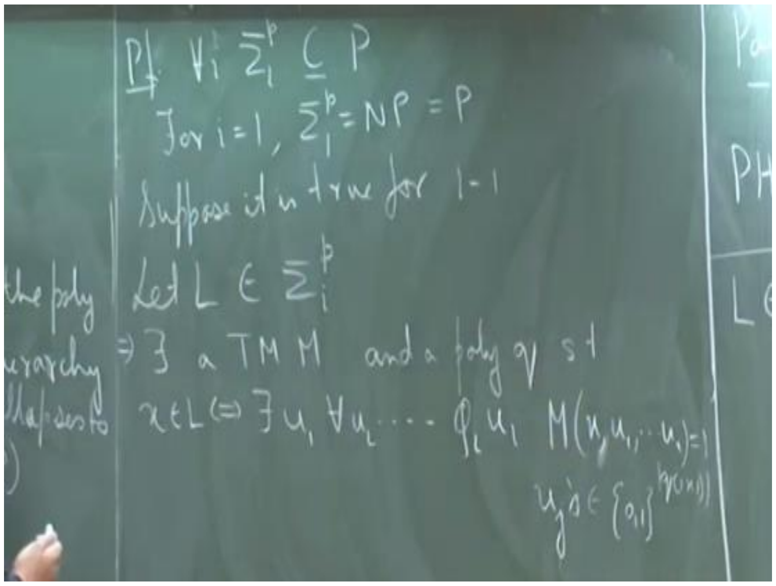
PH $\subseteq$ PSPACE

If P = NP then PH = P (the poly hierarchy collapses to P)

So, now again we can see quite easily that for all i, sigma i p is contained in PSPACE. Maybe I should not have erased that language and the reason for that is if you look at the complete problem for sigma i p that actually erased out. So, that is nothing but a special case of the TQBF language. So, it is just a special case where we are only looking for some small set of variable. And what we get as a corollary is that this entire class PH is also a subset of PSPACE.

There exists this entire plethora of classes and natural languages which sit between the class n p and PSPACE which makes a case as to why we study this hierarchy pi. Let us look at another theorem that if P = NP. Then the polynomial hierarchy also collapses to P. This is what is referred to as the polynomial hierarchy collapses. So, again this is something that is not believed because the hypothesis of this statement is not belief. But let us look at a proof of this very difficult but let us look at it.

(Refer Slide Time: 35:34)



Pf. $\forall i \; \Sigma_i^p \subseteq P$

For $i=1$, $\Sigma_1^p = NP = P$

Suppose it is true for $i-1$

Let $L \in \Sigma_i^p$

$\Rightarrow \exists$ a TM M and a poly q s.t.

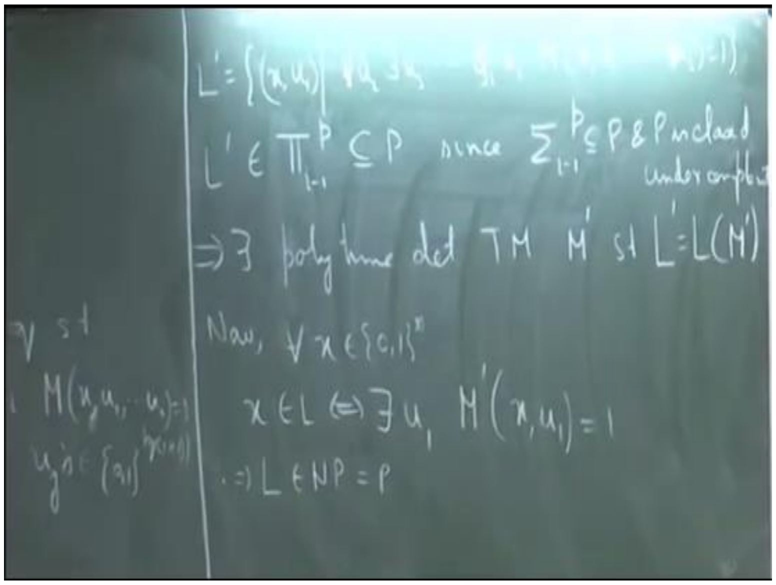
$x \in L \Leftrightarrow \exists u_1 \forall u_2 \cdots Q_i u_i \; M(x, u_1, \ldots, u_i) = 1$

u_j's $\in \{0,1\}^{q(|x|)}$

So, we will prove this by induction, so what we will show is that for all I sigma i p is contained in P. That will immediately imply that the polynomial hierarchy is contained in P. For I = 1 this follows from our hypothesis, if I is equal to 1 then sigma 1 p is by definition the class N P which is equal to P by our hypothesis. So, therefore the claim holds true. Suppose it is true for i –1 so how do we show it to be true for i.

Let L be a language which belongs to sigma i p. This means that there exists a polynomial time turing machine M and a polynomial Q such that x belongs to l if and only there exists a string u 1 such that for all u 2 some quantifier Q i, u i, and M on x 1, x, u 1 up to u i evaluates to 1. And all the u j’s belong to have size Q of X. Now we can define a language L prime based on this language.

(Refer Slide Time: 38:14)



$L' = \{(x, u_1) \mid \forall u_2 \exists u_3 \ldots$ [illegible] $= 1\}$

$L' \in \Pi_{i-1}^{P} \subseteq P$ since $\Sigma_{i-1}^{P} \subseteq P$ & P is closed under complement

$\Rightarrow \exists$ poly time det TM M' st $L' = L(M')$

Now, $\forall x \in \{0,1\}^*$

$x \in L \Leftrightarrow \exists u_1\ M'(x, u_1) = 1$

$\therefore \Rightarrow L \in NP = P$

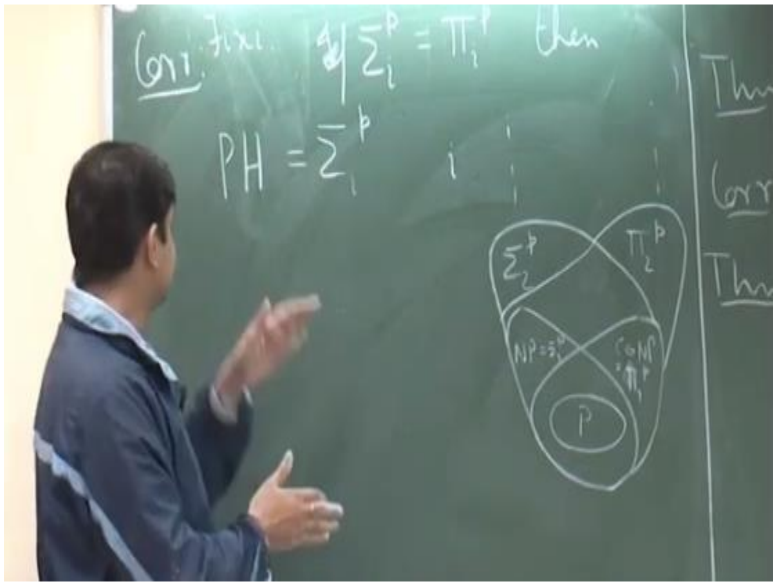
Let us define L prime which is the set of all tuples of the form x, u1 such that for all u 2 there exist u 3. Q i u i and M given X, u 1, u i evaluates to 1. I am just clubbing u 1 together with the given instance x. Now what can we say about L prime? It has i - 1 quantifier and it begins with a for all so therefore L prime belongs to pi i - 1 P. And since we had assumed earlier that sigma i p belongs to P.

Here we have assumed that sigma i - 1 is belongs to P, so since P is closed under complement pi i - 1 also belongs to P. This also belongs to P since it is that we still do not know. But we what we can say is that since this is equal to P and P is so this we know is closed under complement, well does not matter because P by definition is a subset of all these classes. So, we can write equal to also.

So, if this belongs to P that implies that there exist some polytime deterministic turing machine M prime such that L prime is equal to L of M prime. So, now let us go back to the language L prime so when do we say that language an instance is L prime. By our definition for all strings X we know that X belongs to l if and only if there exist U 1, such that M prime given X, u 1 = 1. So, that is what our definition of l prime was that there exists some u 1 which together with the string x and for all these certificates x gave us 1.

So, therefore if you just check this so this is nothing but the definition of NP. So, therefore l belongs to np which is equal to p. Most of these theorems that you would see they will have this similar kind of flavor because it is just basically cutting down on the number of steps in the polynomial hierarchy and then being able to prove something. Actually, we can generalize this theorem a little bit. I just state it as a corollary.

(Refer Slide Time: 42:57)



So, instead of assuming that t = NP even if you assume something much weaker that if so for all i if sigma i p = pi i p. Then what do you think we can say about PH? If I just fix some i. So, it can be for any i so basically do not confuse this for all I statement together with this what this means is that if you fix any i that is greater than or equal to 1. And if for that particular i sigma i p = pi i p what do you think we can say about PH?

Well not quite so PH will not quite all the way collapse down to p but it will collapse to sigma i of p. So, we have all these classes so let us draw small picture. We have the class p over here. And we have the class n p and the class co- NP. So, P is of course a subset of these two classes. This is nothing but sigma 1 p and this is pi 1 p. So, where does sigma 2 p sit? Sigma 2 p again encompasses both these and it is something like this.

This I have you can think of this as sigma 2 p and you have similarly pi 2 p and the hierarchy continues. So, each level of the hierarchy contains all the classes in its lower levels. And each level of the hierarchy I mean you can think that the sigma and the pi versions are in some sense separate. If for any particular level I if the entire polynomial hierarchy above it if for any particular level these two classes coincide then basically what is happening is that the entire polynomial hierarchy that is sitting above it will collapse to that particular level.

It does not imply that even the lower classes would collapse but at least we can say that the highest higher ones would collapse. It is not for one particular it is for any i. So, that is why so maybe can just rephrase the statement slightly differently. So, fix i, now if sigma i p = pi i p. Then it collapses to that. This i, that fixing can be anything. So, that is what I meant earlier. But there exist again, I do not I mean depends how you interpret it but there exist would mean that for some particular I, this is happening.

It should be fine as long as you understand what is happening so that is what is important. Again, this you can just try out as a proof the proof is nothing is just its very similar along these lines where you have to prove it inductively just you have to start your induction from this particular i, instead of from i = 1. So, I will stop here today.