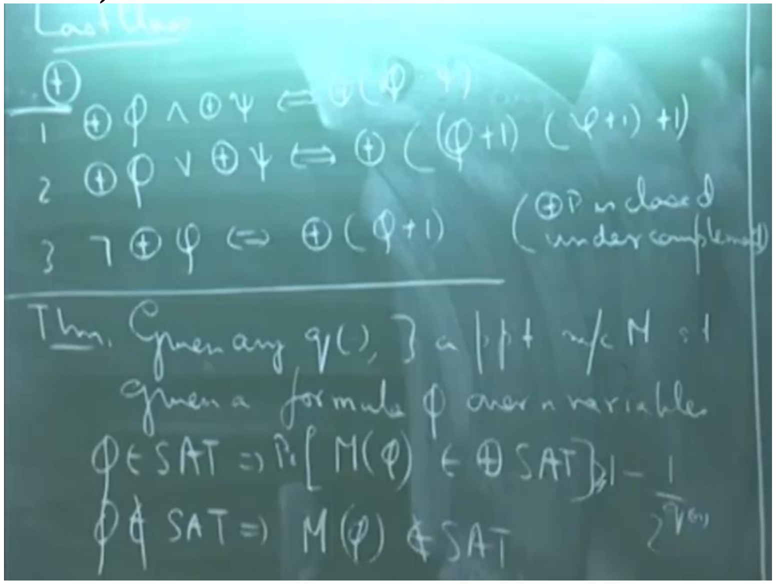
Computation Complexity Theory

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

Module No # 05
Lecture No # 25
Amplified version of Valient-Vazhirani Theorem

(Refer Slide Time: 00:22)



Last class

$\oplus$

1 $\oplus\varphi \wedge \oplus\psi \Leftrightarrow \oplus(\varphi\cdot\psi)$

2 $\oplus\varphi \vee \oplus\psi \Leftrightarrow \oplus((\varphi+1)(\psi+1)+1)$

3 $\neg\oplus\varphi \Leftrightarrow \oplus(\varphi+1)$ ($\oplus$P is closed under complement)

Thm. Given any $q()$, $\exists$ a ppt m/c M s.t. given a formula φ over n variables

$\varphi \in SAT \Rightarrow \Pr[M(\varphi) \in \oplus SAT] \geq 1 - \frac{1}{2^{q(n)}}$

$\varphi \notin SAT \Rightarrow M(\varphi) \notin \oplus SAT$

So we saw the following last class, we saw the parity quantifier. And we saw some properties of this quantifier namely that parity of 2 formulas, parity Phi and parity Psi is equivalent to parity of this formula Phi dot Psi. So we how to define this new formulae, based on the 2 and these 2 are equivalent. So similarly parity of Phi or parity Psi is equivalent to parity of Phi+1 dot Psi+1+1. So this +1 is nothing you just have a trivial formula.

Which has only 1 satisfying assignment and you look at the sum of that with Phi or Psi whatever it is. And thirdly not parity of Phi is equivalent to parity of Phi+1. So this shows that the class let us say parity P or any complexity class C. If I equivalently define a class parity C see how to define that, but for the time being let us only bother about parity P. So this shows that parity P is closed under complement.

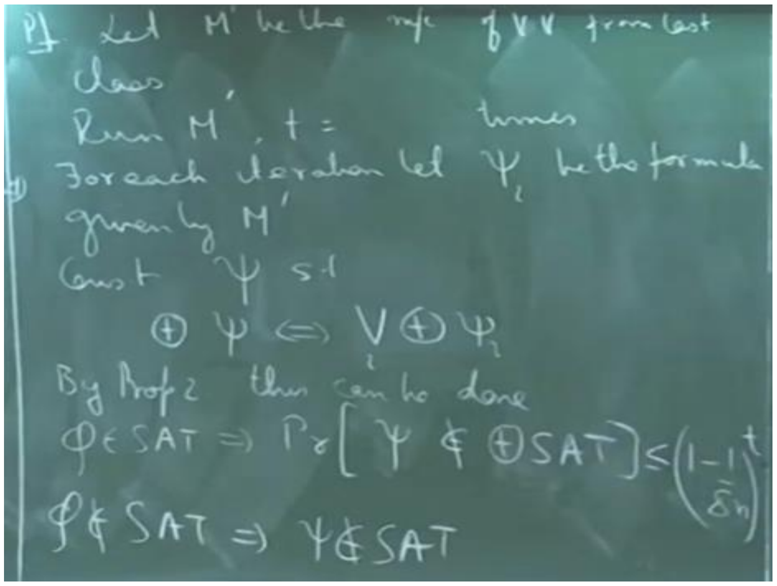
In other words we can without loss of any generality talk about a parity P, machines which accepts if there are, an even number of satisfying assignments. Because I can always add 1 extra assignment to get a new formula what do we have? So the idea as I said last time was to amplify the parity is a SAT version of valiant Vazirani. So that was the idea so let us write down what we want to do?

So the theorem is so given any polynomial q there exists a probabilistic machine M such that given a formulae Phi over n variables. If Phi is satisfiable then the probability that, so basically what this machine does is that it takes the formula and it outputs another formula. In the same manner as Valiant Vazirani but will see it slightly different. So M Phi will output another formula which belongs to parity SAT with very high probability 1 - 1 over 2 to the power this q n.

So whatever polynomial we want we can boost up our success probability to that much so this is greater. And if Phi does not belong to SAT then I can just write this as M Phi also does not belong to SAT. So it is not satisfiable with probability 1. So the idea is very simple, so what do we do? So what does the corollary of Valiant Vazirani give us? So the actual version of Valiant Vazirani said that there exists a probabilistic machine which I given any formula if that formula is satisfiable M will output another formula which has a unique satisfying assignment.

And if Phi were not satisfiable so this has no satisfying assignment in other words if it has a unique satisfying assignment. I can consider that to be as an instance of parity SAT with probability 1 over 8 n. With success probability so now I want to boost that 1 over 8 n to 1- 1 over 2 to the power q n for any q so the idea is again basically to take t trials so I want to take some polynomially many trials of that earlier machine.

(Refer Slide Time: 06:37)



Pf. Let M' be the m/c of V.V from last class

Run M', $t =$ times

For each iteration let ψ_i be the formula given by M'

Const ψ s.t

$$\oplus\psi \iff \bigvee_i \oplus\psi_i$$

By Prop 2 this can be done

$$\phi \in SAT \Rightarrow \Pr[\psi \notin \oplus SAT] \le \left(1 - \frac{1}{8n}\right)^t$$

$$\phi \notin SAT \Rightarrow \psi \notin SAT$$

So the proof is so let M prime be the machine of Valiant Vazirani from last class. So we construct our machine M as follows so run M, t equal to some number of times so we will see what that t is. And for each iteration let Psi i, be the formula given by this is M prime given by M prime. So we have some t formulas and what do we know? So we know that so now construct a formula Psi which is the r of no.

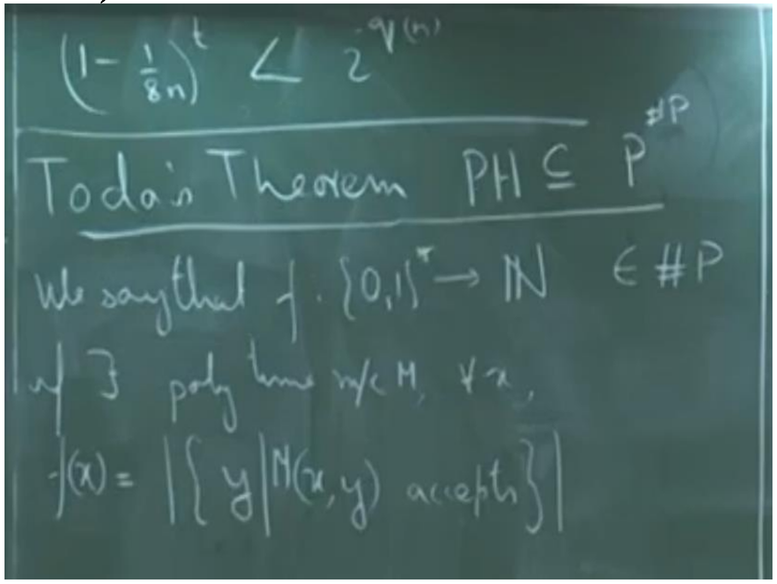
So I want to construct a formulas Psi such that this is odd if and only if I take the, r of all these parity is Psi I that is odd. So I will construct a Phi in this manner and the reason why I construct such a Phi is because of property 2. I mean I have all these individuals Psi i's and I am taking a, r of them. So I can always construct a, another formula Psi whose parity is equivalent to the, r of the parity of all these r's.

And not only that the other property that we have is that if the initial formula if Phi was not satisfiable then all these Psi i's, are also not satisfiable. So no matter how many r's we take it will always be 0. So by property 2 this can be done so now what do we want our t to be so basically what is the probability of error here? So Phi is in SAT implies that the probability that this Psi that we construct.

Suppose this is not an instance of parity SAT how much is the probability of that happening? So this does not belong to parity SAT if none of these belong to parity SAT. And what is the probability that none of these would belong to parity SAT? 1 - 1 over 8 n for each 1 of them because their success probability is 1 over 8 n so failure probability is 1-1 over 8 n so the probability that this happens is less than 1-1 over 8 n whole to the part t.

So in other words the probability that Psi would belong to parity SAT is 1- is greater than or equal to 1- this quantity. And if Phi is not satisfiable then by what I said earlier this is also not satisfiable by our construction. So now we can very easily see what t we want? So we want a t such that it must be greater than that quantity so in other words the failure probability should be less than this quantity.

(Refer Slide Time: 12:03)



$\left(1-\frac{1}{8n}\right)^t < 2^{-q(n)}$

Toda's Theorem $PH \subseteq P^{\#P}$

We say that $f : \{0,1\}^* \rightarrow \mathbb{N} \in \#P$

if $\exists$ poly time m/c M, $\forall x$,

$f(x) = |\{y \mid M(x,y) \text{ accepts}\}|$

So we want a, t such that 1 - 1 over 8 n to the power t should be less than 2 to the power - q of n that q is a polynomial. So I think taking T like something like n times q of n will work. So if we fix to t to be n dot q of n or maybe some, constant times n dot q of n. Let us say may be 10 or something 10 n q of n that should satisfy this inequality. So this is still fine so this is still a polynomial.

So this is a standard way of boosting the probability of any algorithm which has 1 sided error. So now let us step back a little bit and so we said earlier that we will look at several applications of this Valiant Vazirani theorem. So 1 application we saw last time were under certain conditions we got an RP algorithm for any problem in NP so there is also a very famous application and this is popularly known as Toda's theorem.

So this was proven in 91 and this is actually a Godel prize winning word. So Godel price is 1 of the top prizes in theoretical computer science and due to this work of Toda in 98 he got the Godel price. So we will look at this proof may be in the next couple of lectures. So what he showed was that a polynomial hierarchy is contained in P to the sharp p**.** So you have a polynomial time machine which has oracle access to a sharp p function.

And not only does it access this oracle it accesses this oracle only once. So during the, suppose you are given any problem in PH the polynomial time machine during its execution it computes just 1. Let us say a sharp SAT instance so let us say it creates some Boolean formula and it asks the oracle how many satisfying assignments does it have? So based on whatever answer the oracle gives it continues its deterministic computation.

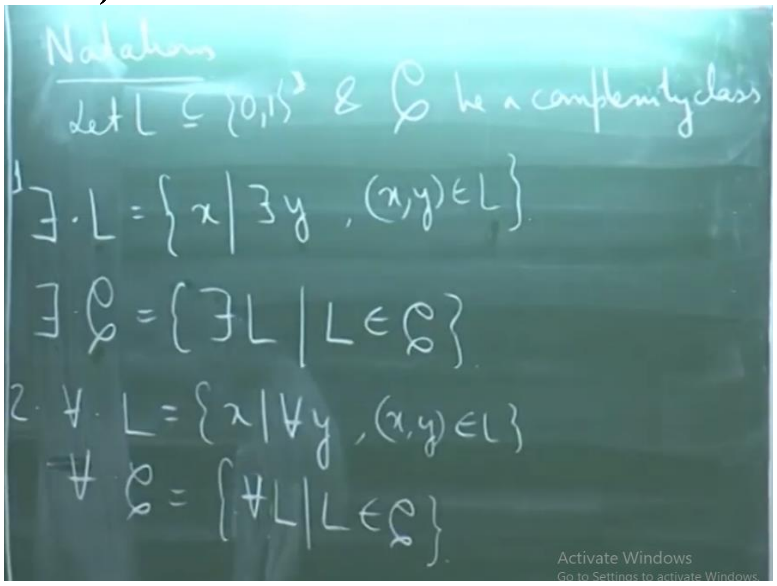
And is able to solve the PH problem so this is quite strong actually because so if you recall. So why is this important so sharp p I defined it last time know? So sharp p is so let me define it so sharp p, I think I defined it last time. So it is the class of functions f let me define it this way so we say that so let us say we have a function from 0, 1 star to, n. So this belongs to sharp p so this is a function class this is not a language in the traditional sense if there exist.

So what was the definition did I give definition based on NP or just a number of certificates? **S**o there exists some polynomial time machine M. So let me just complete it anyway poly time machine M such that for all X it is the cardinality of the number of y's. Such that x, y so M given x, y accepts so for every string X it is basically the cardinality of the number of certificates that x has for that machine.

So let us get back to Toda’s theorem so this is what he had shown so what we will do is we will prove this theorem in 2 parts. The first will show that yes a constant number of trials is equivalent to 1 trial yes in fact even a polynomial number of trials. In the case of P, I think even polynomially number of trials can be made equal to 1 trial because you are just well maybe; not, maybe.

It probably depend where it is being called but yeah certainly if you have a machine in this class P to the power sharp p which makes constant number of queries to the sharp p oracle. It can be simulated by another P to the sharp p machine which only makes 1 query. I think even login will suffice but again I am not sure without a proof but anyway. So let us look at some notations and definitions before I begin and I will then give you the roadmap as to how the proof proceeds?

(Refer Slide Time: 19:32)



Notations

Let $L \subseteq \{0,1\}^*$ & $\mathcal{C}$ be a complexity class

1. $\exists \cdot L = \{x \mid \exists y, (x,y) \in L\}$

$\exists \mathcal{C} = \{\exists L \mid L \in \mathcal{C}\}$

2. $\forall \cdot L = \{x \mid \forall y, (x,y) \in L\}$

$\forall \mathcal{C} = \{\forall L \mid L \in \mathcal{C}\}$

So let so whenever we talk about L we will just mean L to be a language and whenever we talk about script C we will mean that to be a complexity class. So the language there exist L is equal to the set of all strings x such that there exist a string y such that x, y belongs to L. So given a language L we define this new language there exist still which is defined this way. So you can immediately see that NP can be defined as there exists dot P.

So there is some machine so there is some language in P such that, there exists a certificate which makes that instance belong to that language. So this is 1 way and similarly so this is 1 so similarly we define the complexity class there exist C as the set of all languages of the form there exist L where L is in C. So this is how you define the 1 way of defining the complexity class NP so NP is nothing but there exist dot P.

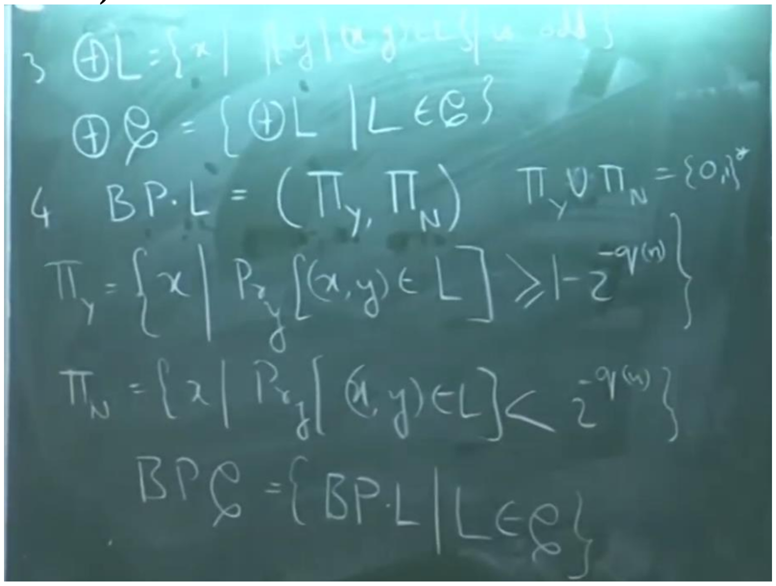
No there exist L is just a language so maybe you can say that SAT is there exist dot some language in polynomial time. For example suppose if you define the following language which given a formula and a satisfying assignment can check. If I am, not a satisfying assignment but given a formula and an assignment checks whether the formula evaluates to true or false on that assignment.

So this is a polynomial time computation no this is not you are saying that as the complexity class yes so there exist L is NL. But that is a different thing but that is not a log space machine. So if you are looking at the class L and if you are talking about there exist L it is basically set of all languages of the form there exists dot so let me so there exist dot. So I should not use the word L here maybe something like L 1 or something such that L 1 belongs to the complexity class L.

So here that there exist means that there exists a string such that x, y would belong to the language L 1. Since we are looking at the class L so I would want that string of course it has to be polynomially bounded. But so given Y so of course it is the class NL but is NL equal to there exist L. No that is not I mean that need not be true no, yes for NP that is not true that you can do with a verifier which has log space because if you take the same problem.

Let us say the problem where you are given a formula and assignment and you want to check satisfiability that is not a log space verifier. Anyway let us just continue with this and I will try to check that. So similarly we can have the for all quantifier. It is basically the same thing and for all C is the class of all languages for all L such that L belongs to C.

(Refer Slide Time: 25:13)



3 $\oplus L = \{x \mid |\{y \mid (x,y) \in L\}| \text{ is odd}\}$

$\oplus \mathcal{C} = \{\oplus L \mid L \in \mathcal{C}\}$

4 $BP \cdot L = (\Pi_Y, \Pi_N) \quad \Pi_Y \cup \Pi_N = \{0,1\}^*$

$\Pi_Y = \{x \mid Pr_y[(x,y) \in L] \geq 1 - 2^{-q(n)}\}$

$\Pi_N = \{x \mid Pr_y[(x,y) \in L] < 2^{-q(n)}\}$

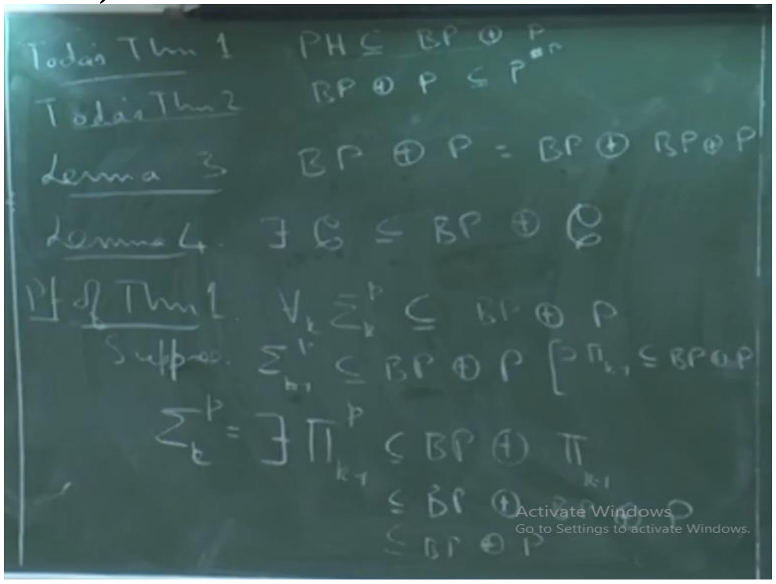
$BP\mathcal{C} = \{BP \cdot L \mid L \in \mathcal{C}\}$

So I will define 2 more quantifiers so parity of L is set of all strings x such that the cardinality of the following set y such that x, y belongs to L is odd. So it is a set of all strings x such that the number of certificates that it has is odd and parity C is parity L for L belonging to C. So all these are again traditional languages we also have the language defined as BP dot L. So this is a promise language in the sense that so BP dot L is defined as its set of yes instances and no instances.

Where the S instances are so what are the yes instances it is all x's such that the probability over y that x, y belongs to L is. So what do I want here so I will just take the definition of strong BPP for the time being. So if necessary we can relax it so this is greater than 1 - 1 over 2 to the power - q of n. And the no instances are set of all x such that probability x, y belongs to L is less than 2 to the power - q n.

And BP dot C is equal to the languages BP dot L such that L belongs to C. So let us look at the way we will precede with our proof so what we will show first is.

(Refer Slide Time: 28:27)



Toda's Thm 1 $PH \subseteq BP \oplus P$

Toda's Thm 2 $BP \oplus P \subseteq P^{\#P}$

Lemma 3 $BP \oplus P = BP \oplus BP \oplus P$

Lemma 4. $\exists\, \mathcal{C} \subseteq BP \oplus \mathcal{C}$

Pf of Thm 1. $\forall_k\ \Sigma_k^P \subseteq BP \oplus P$

Suppose. $\Sigma_{k-1}^P \subseteq BP \oplus P\ [\Rightarrow \Pi_{k-1} \subseteq BP \oplus P$

$\Sigma_k^P = \exists \Pi_{k-1}^P \subseteq BP \oplus \Pi_{k-1}$

$\subseteq BP \oplus BP \oplus P$

$\subseteq BP \oplus P$

So Toda's theorem let me call it 1 is will show that PH is contained in the class BP dot Parity dot P. Whatever that means and secondly we will show that this class BP dot parity dot P is contained in P to the power sharp p. So BP dot L is basically a language whose which is a promise language in some sense. So basically it has so if you look at the universe if you look at sigma star 0, 1 star a sub set of those strings belong to the I mean a sub set of those strings are the S instances of this language. And some other sub set; some other disjoint sub set are the no instances.

It is not that the complement of the yes instances, are the no instances necessarily. So the yes instances, yes so I mean this is just another way of writing whatever let us say definite. So when we define let us say BPP so if we look at the class C to be our class P then BP dot C is BP dot P is nothing but the class BPP. So how did we define BPP? So we say that given so we said that a language L is in BPP if for all X, X belong**s** to L implies that the probability of acceptance is greater than 2 third.

Or in the case of strong BPP it is greater than 1 - 2 to the power - q n and if x does not belong to l then the probability of acceptance is less than 2 to the power –q. So this is how we defined it so but what about other thing? So basically this has the promise that every string would either have this property or it would have this property. So that is how this thing is defined. So this is what is known as a semantic based class as opposed to syntactic class.

So what is happening here is that so I am just not telling you that the complement of the yes instances, are the no instances. I am enforcing that if something has to be no it has to satisfy some property and that has to happen for every string so every string must necessarily have either this property or this property. So that no it is a complement so here it is a complement but it is not a complement based on the property of the machine.

So I am not saying that for all x I will accept x if x has this property and otherwise I will reject x. So it is not a it is not getting rejected based on the property of that machine it is basically getting rejected by an extra stronger property. So every string yes every string either belongs to this set or belongs to this set. But there is something more, strong that is there in the definition of that language is sigma star.

So this is true for all x actually so maybe I should write it here that Pi Y intersect pi and of course that follows from the definition. But this is more important so this is equal to 0, 1 star in this case so it has to happen that either of these 2 inequalities are satisfied. The promise is in the machine I mean the machine has this property that it will either accept it with a very large probability or it will accept it with very small probability.

It will not have something in between so that is the promise in it. So the first part is basically what we will focus on today and this basically uses in some sense an extension of the Valiant Vazirani theorem that we proved. So let us see how we will prove this? So what I will prove is will prove 2 Lemmas. So I have theorem 1 and 2 so let me call this Lemma 3 and Lemma 4. So lemma 3 states that BP dot parity P is the same as BP dot parity dot BP dot parity dot P.

So no matter how many such pairings you have it basically boils down to just 1 pair and Lemma 4 will prove that there exist C is contained in BP dot parity dot C. So if you have a there exist quantifier it can be replaced with a BP dot parity quantifier. And this basically follows from Valiant Vazirani and as we will see may be next class. So now suppose we have these 2 Lemmas can we prove theorem 1.

So let us see proof of theorem 1 based on these 2 Lemmas so what is PH? So I can just say that so let us consider. So we will prove this by induction so we will show that for all k sigma k P is contained in BP dot parity dot P ok so when K is = 0 this is nothing but the class P. So that trivially is in the class BP dot parity P, suppose sigma k - 1 P is contained in BP dot parity dot P ok. So then what we can say about sigma k so what is sigma k?

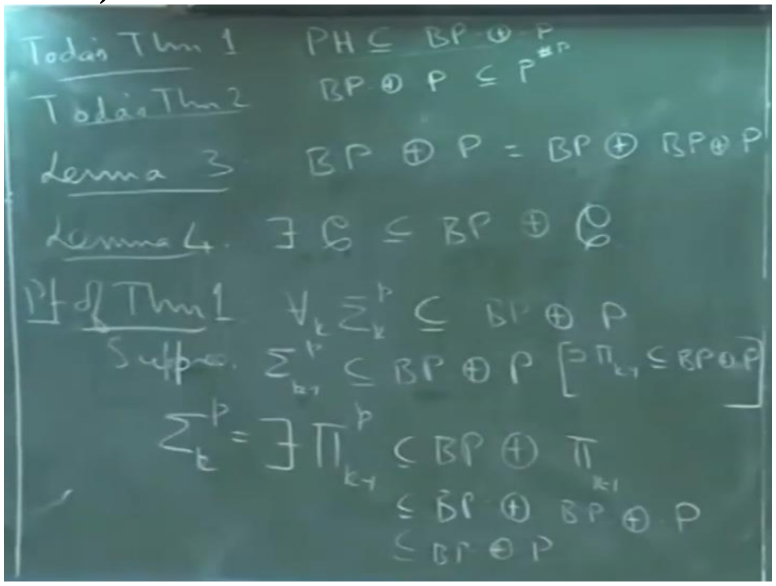
So it is there exist pi K - 1 P we can do that also but actually the property that we will use here is that this BP operator is closed under complement. I mean I can always flip the answer in the end because so it is not I think I need to give a formal proof of that. So if I just apply this Lemma here what I have is this is contained in BP dot parity dot pi k – 1. We can again apply actually we can do that again actually.

But pi K - 1 no actually there is a problem so pi K - 1 is for all sigma k – 2 here I am only stating this for there exist C. So the property that is used here is that this is closed under complement. I mean if sigma k - 1 is in BP dot parity P then pi k - 1 is also in BP dot parity P. We cannot. Because we are only stating this lemma for there exist so pi k - 1 is for all C but actually it does not matter.

I mean basically since it is closed under complement we can also have a, for all version but so let me just mention it here and maybe we look at the proof next time. That this implies that pi k - 1 is also contained in BP dot parity dot P. So now I can just write this as BP dot parity dot BP dot parity dot P. And by Lemma 3 what we have is? This is contained in BP dot parity dot P so that proves theorem 1 for us.

So the point is how do we prove Lemma 3 and Lemma 4 so again let me just give so Lemma 3 needs certain non-trivial steps. So let me just give the steps today and we will look at the proof tomorrow.

(Refer Slide Time: 40:27)



Todas Thm 1 $\quad PH \subseteq BP \cdot \oplus \cdot P$

Todas Thm 2 $\quad BP \cdot \oplus \cdot P \subseteq P^{\#P}$

Lemma 3 $\quad BP \cdot \oplus \cdot P = BP \cdot \oplus \cdot BP \cdot \oplus \cdot P$

Lemma 4. $\quad \exists \cdot \mathcal{C} \subseteq BP \cdot \oplus \cdot \mathcal{C}$

Pf of Thm 1 $\quad \forall_k \; \Sigma_k^p \subseteq BP \cdot \oplus \cdot P$

Suppose. $\quad \Sigma_{k-1}^p \subseteq BP \cdot \oplus \cdot P \; [\Rightarrow \Pi_{k-1} \subseteq BP \cdot \oplus \cdot P]$

$$\Sigma_k^p = \exists \Pi_{k-1}^p \subseteq BP \cdot \oplus \cdot \Pi_{k-1} \subseteq BP \cdot \oplus \cdot BP \cdot \oplus \cdot P \subseteq BP \cdot \oplus \cdot P$$

So I just call it claim 5 so the first claim is that parity of any complexity class C is equal to parity dot, parity dot C. Similarly BP dot C is = BP dot, BP dot C and finally BP dot parity dot C is = parity dot BP dot C. So suppose if we have these 3 claims then Lemma 3 follows quite easily see 1 direction is trivial. I mean the left hand side is contained in the right hand side trivially. What we want to show is that BP dot parity dot BP dot parity dot P is contained in the left hand side so by let us say claim 7 I can switch these 2 quantifiers.

So this gives us BP dot BP dot parity dot parity dot P so this is by claim 7. And now I use claim 5 and 6 to get the following this means that BP dot parity dot P by 5 and 6. So that is at least the first component for us provided that we have these 3 claims. And again these 3 claims are not difficult I mean you can just try out maybe the first 2 are quite trivial third-one is some little bit of work but we look at that next time.

Third-one no it is, not difficult I mean it is just again by probabilistic argument. I mean you probabilistically argue that there will be some good certificate with high probability. So when will it so basically the difficult part of these proofs is writing out the definitions and properly deducing the next step. I mean there is no nothing magic happening here but the next Lemma so Lemma 4 is more non-trivial.

But since we have already proved Valiant Vazirani so we will see that it is actually an extension of Valiant Vazirani so we will be able to do that may be tomorrow. And theorem 2 is again so it uses some other ideas so we will stop here today.