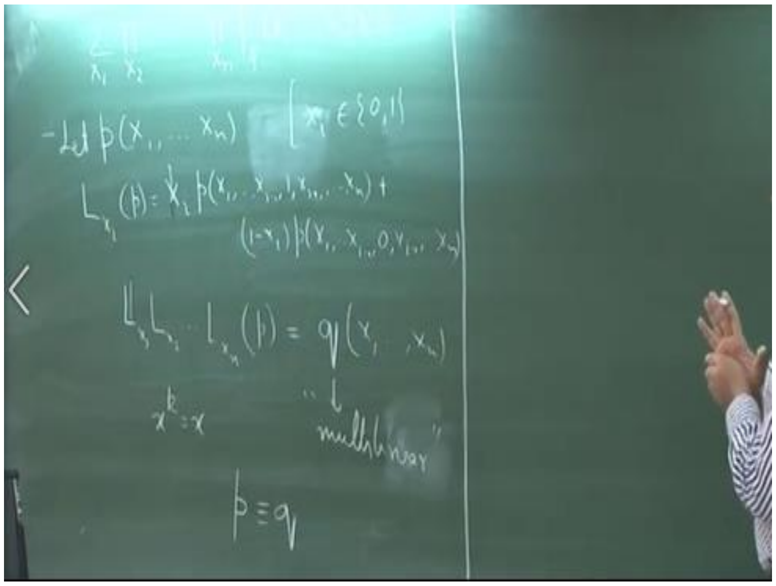
Computational Complexity Theory

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

Lecture -36
Introduction

So, we saw last time that sharp P can be simulated using an IP protocol and we were in the middle of showing how key space in particular the TQBF problem also has an IP protocol. So, I will just give a sketch of the argument I will not go into the details because the idea is basically the same, I will just point out the extra work that the protocol needs to do. And then we look at something else. So, for the TQBF problem what the prover needs to convince the verifier is the following.

(Refer Slide Time: 01:05)


$x_1 \quad x_2 \quad x_n$
$-$ Let $p(x_1, \ldots x_n)$ $[x_i \in \{0,1\}]$
$L_{x_i}(p) = x_i\, p(x_1, \ldots x_{i-1}, 1, x_{i+1}, \ldots x_n) +$
$(1-x_i)\, p(x_1, \ldots x_{i-1}, 0, x_{i+1}, \ldots x_n)$
$L_{x_1} L_{x_2} \cdots L_{x_n}(p) = q(x_1, \ldots, x_n)$
$x^k = x$
multilinear
$p \equiv q$

That the following expression let us say that the last quantifier is a for all quantifier so we have a product of the polynomial is not 0. So, he wants to show that this whatever is the quantified Boolean formula psi that is given that is that belongs to TQBF. In other words that is that evaluates to true so this polynomial should evaluate to non-zero. So, this is what the aim of the prover is;

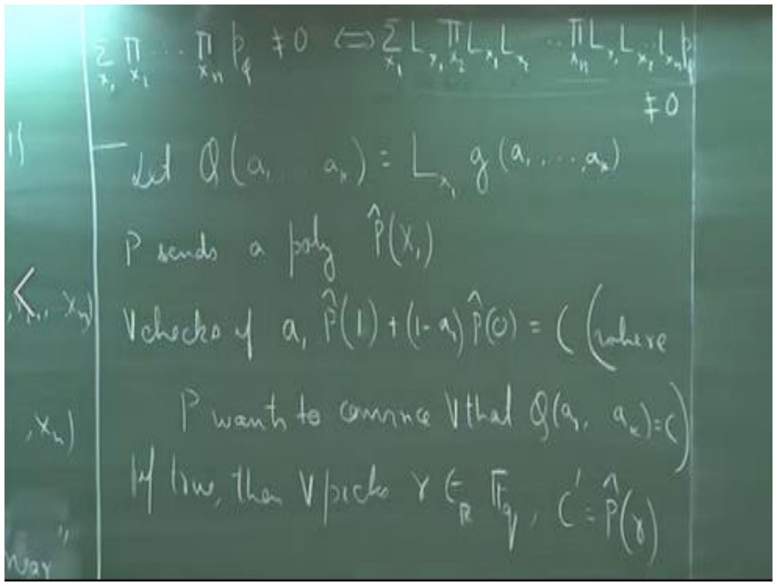
And so let us define so I did define this last time but let us just recall. So, suppose we have a polynomial p, so let p be a polynomial in n variables and define L x i of p to be equal to x i p x 1 through x i - 1 1 x i + 1 through x n +1 - x i p x 1 through x - 1 0, x i + 1 up to x n. So, this is called the linearization operator on i, so basically what this operator does as we saw last time with an example is that it just so any term any monomial in this polynomial p where x i has a degree higher than 2 it just reduces that degree to 1.

And the property that this operator has is that, so suppose this operator is operated on a polynomial p in fact so if we operate the following operators on the polynomial p L x 1 L x 2 so on till L x n on p. So, this will give us some polynomial q again in x 1 through x n with the nice property with the nice property that q is multilinear. In other words, it is linear in each variable. So, it is a multivariate polynomial but it is linear in each variable as opposed to p.

And what we have is that if let us say each of these variables took their values from the set 0 1 then these two polynomials are identical, and identical is in the sense that for every setting of x 1 through x n they will give the same value. So, this is what we saw last time and this basically follows from the trivial observation that if you take, if you look at the equation x to the power k is equal to x then for x belonging to 0 1 this equation is always true.

So, this is basically the additional fact that we will use to show that the prover can convince the verifier that this is true, so and if this is not true in other words if this is equal to 0 then with very low probability the prover will be able to convince them. So, we will do is we will replace the following equation as follows.

(Refer Slide Time: 06:15)



$$\sum_{x_1}\prod_{x_2}\cdots\prod_{x_n} p_\phi \neq 0 \iff \sum_{x_1} L_{x_1}\prod_{x_2} L_{x_1} L_{x_2} \cdots \prod_{x_n} L_{x_1} L_{x_2} \cdots L_{x_n} p_\phi \neq 0$$

Let $Q(a_1 \ldots a_k) = L_{x_1}\, g(a_1, \ldots, a_k)$

P sends a poly $\hat{P}(x_1)$

V checks if $a_1 \hat{P}(1) + (1-a_1)\hat{P}(0) = C$ (where P wants to convince V that $Q(a_1, \ldots a_k) = C$)

If true, then V picks $r \in_R \mathbb{F}_q$, $C' = \hat{P}(r)$

So, sigma x 1 pi x 2 pi x n of p phi is not equal to 0 if and only if sigma x 1 L x 1 pi x 2 L x 1 L x 2 and so on till pi x n L x 1 L x 2 up to L x n of p phi is not equal to 0. So, basically what we do is that we intersperse these operators with the appropriate linearization operators. And the idea behind is that whenever it is actually for sum this is not required this is kind of redundant, because we observed last time that summation does not increase the degree of the polynomial.

But still, I mean we will just keep it for the sake of completeness. But whenever I am applying this product operator what can happen due to this application is that the degree of the resultant polynomial might double up. But then when we apply these two operators successively what that makes what that will enable is it will bring down the degree of the resultant polynomial in x 1 and x 2 down to 1 and the same thing happens for all the other operators.

So, now the protocol is again very similar so the protocol is the same as what we had for sat bar or sharp P, that is we will strip one operator at a time. So, what was the thing for sharp P so the prover center polynomial, which was supposed to be equal to this entire thing. So, that is what the prover wants to convince the verifier but the verifier is skeptical, so what he does is? He evaluates whatever the prover sends on 0.

And if that is equal to whatever value the prover had started off with. So, in this case the prover will start off with 0. So, that is the case then he evaluates that polynomial on some random point keep that the resultant value and send back the random variable back to the prover. Then in the next step the prover wants to convince that this polynomial or this expression is equal to a certain polynomial.

So, one so you can think of these as an operator and in each iteration of the prover verifier interaction an operator gets stripped down. So, we know how to strip down in the case when the operator is a sigma operator or when it is a pi operator. So, suppose it is one of these linearization operators. So, what is the corresponding protocol? So, let Q sum a 1 to a k be equal to sum L x 1 let us say I mean I can always rename my variable so that i is the first variable of let us say something like g a 1 through a k.

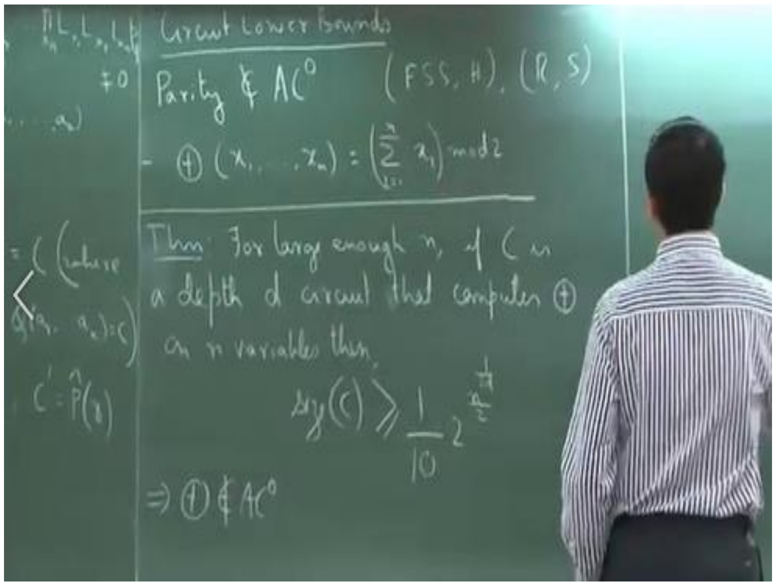
So, then prover sends a polynomial let us say P hat we will check the following V checks, so this is a polynomial in x 1. So, V checks if a 1 P hat of 1 + 1 - a 1 P hat of 0 is equal to whatever is the prover trying to convince is equal to C where, where P wants to convince V that is equal to C. So, if this check is satisfied by the prover then he will do the again the same thing, so he will pick if true then V picks again it is the same thing.

V picks r randomly from the field f q computes the next C prime, the C prime for the next round which is equal to P hat of r and then sends back r to the prover. And in the next round the aim of the prover is to convince is to send another polynomial such that it will strip the next operator and it will also convince the verifier that whatever is the correct expression I mean in the case that it is a linearization operator whatever is the P hat for the next iteration.

So, P hat 1 + 1 minus whatever is that operator a 1 P hat 0 should be equal to C prime and it proceeds in the same manner. So, again I mean there is nothing much different that is happening as far as the protocol is concerned it is the same thing but the crucial fact was basically this observation. So, whenever x belongs to 0 1, so this is basically the crucial fact. So, I will not go into the details of the error analysis you can read it from the book.

So, any questions? So, the next thing that we will do is very different for in flavor from the kind of things that we have been seeing in the last few lectures.

(Refer Slide Time: 14:46)



Circuit Lower Bounds

Parity $\notin AC^0$ (FSS, H), (R, S)

$\oplus(x_1, \ldots, x_n) = (\sum_{i=1}^{n} x_i) \bmod 2$

Thm: For large enough n, if C is a depth d circuit that computes $\oplus$ on n variables then,

$$\text{size}(C) \geq \frac{1}{10} 2^{(\frac{n}{2})^{\frac{1}{d}}}$$

$\Rightarrow \oplus \notin AC^0$

So, the next thing that we will do is what are known as circuit lower bounds and more specifically the result that we will show is that the parity problem does not belong to the circuit class AC 0. So, this is one of the first and one of the most important absolute lower bound results that is known in complexity. So, as I said many times it is very difficult to prove that something is not possible whether it is in complexity or in any theoretical computer science area.

Because if I want to show that something is not possible I have to basically argue over all algorithms what I have to argue is that there is no algorithm in this universe which can do I mean of course the algorithm will have some restricted amount of resources. But then it is not able to perform the task that we wanted to do. So, in this case what we want to show is that suppose if we take a circuit which has constant depth polynomial size and unbounded Fannin.

No matter what that circuit classic it will never be able to compute the parity function. So, let us recall what the parity function is so the parity function I will just denote it with this symbol, so the parity on n variables is just the sum of the individual bits modulo 2. So, if it sums up to an even number it is 0 otherwise it is 1. So, it is a very simple function and of course I mean we know that there exists NC1 circuit family.

I mean there exists an NC 1 circuit which is able to compute parity quite easily. But when we come down to AC 0 this is not possible. So, there have been many proofs of this result, so the first sort of proof was due to a several people actually in different works and in so the first set of proofs were due to first sax and sipser(())(17:48) and subsequently it was improved by hastard(())(17:55) letter on.

So, this was in the early 80s, and the way they came up with this proof was that in in hastard paper he showed a switching Lemma. So, basically, he showed that if I randomly restrict the variables of a function, so suppose if I say that I have a function on x 1 to x n, and x n randomly restrict some of these variables, that is I fix the values of some of the variables randomly. So, I get another function.

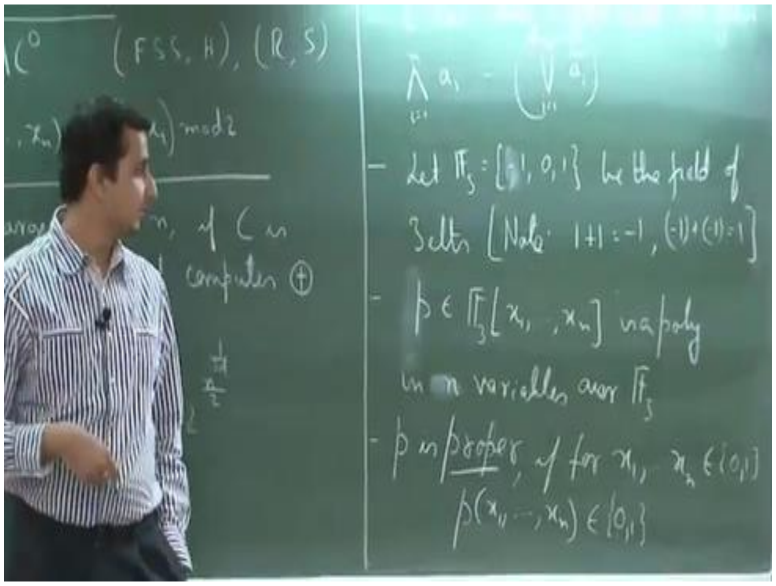
So, that function will be I mean in non-technical terms if I have a random restriction then we can show that so there will be a lot of difference between these two functions up to some I mean if you look at it from a certain point of view. So, I do not want to go into the details of that argument. The reason why I mentioned this earlier result was because we will not look at this proof, the proof that we will look at is one that was given by Rasbora and Smolinsky in the late 80s.

And the idea of this proof is that so what they argue they argue in two parts, in the first part they argue that if we have any AC 0 circuit, so an AC 0 circuit can be approximated very well using a certain kind of a polynomial. So, whatever is the function that is being computed by AC 0 circuit that can be approximated by a nice polynomial. And in the second part they show that the parity function cannot be approximated up to any good approximation factor using such nice kind of polynomials.

So, we will make these terms formal and together these two claims show that parity is not in SC 0. So, basically what we will show is that? We will give a proof by contradiction so suppose if there is an AC 0 circuit for parity then we will get certain contradictions. So, let me write the main theorem that we are going to prove. So, for large enough n if C is a depth d circuit that computes parity on n variables, then size of C is at least 1 10th of 2 to the power n to the power 1 over 2 d divided by 2.

So, in other words if d is a constant, so suppose d is 10 or something then we immediately see that the smallest size circuit that is able to compute parity must have size at least 2 to the power order n for any constant d this is 2 to the power order n. So, I have an immediate corollary of this result is that this immediately implies that parity is not in AC 0. So, the main theorem that we will prove over todays and the next lecture is this theorem.

(Refer Slide Time: 23:01)



$\bigwedge_{i} a_i = \neg\left(\bigvee_{i} \neg a_i\right)$

- Let $\mathbb{F}_3 = \{-1, 0, 1\}$ be the field of 3 elts [Note: $1+1=-1$, $(-1)\cdot(-1)=1$]
- $p \in \mathbb{F}_3[x_1, \ldots, x_n]$ is a poly in n variables over $\mathbb{F}_3$
- p is proper, if for $x_1, \ldots, x_n \in \{0,1\}$ $p(x_1, \ldots, x_n) \in \{0,1\}$

So, the first thing that we will assume is that C has only Not and OR gates. So, why can we assume this? Because using the Morgan's law we can replace any AND gate because suppose if we have an AND gate, so this is the same as so, by just introducing a two more levels I can replace any AND gate with an set of not and or gates. So, we look at the field f 3, so let f 3 and we will just denote the elements in f 3 as -1, 0, and 1 be the field of.

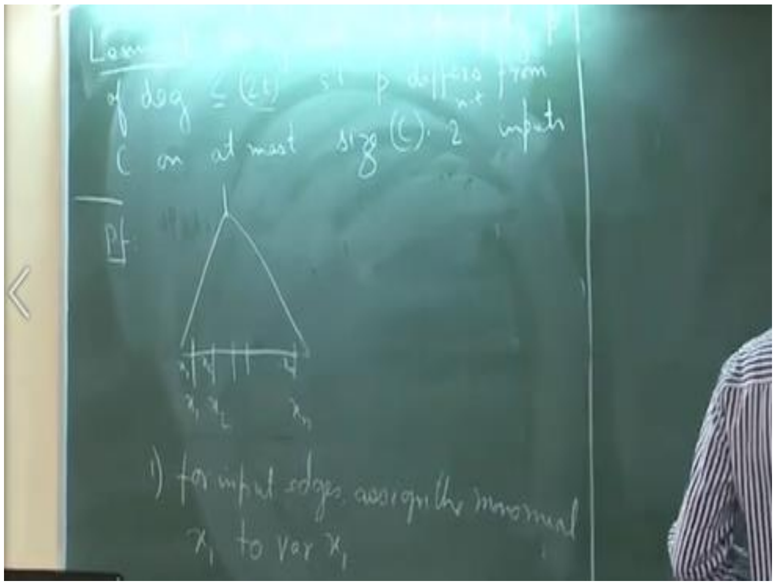
So, as I said during our discussion of the IP protocol that fields have this nice property that, they are closed under addition multiplication and they contain the inverses of these respective operations as well. So, I mean so f 3 I mean by definition is the elements 0 1 and 2 but here just we are using the symbol - 1 to denote the element 2. Because it makes things little more convenient for us.

So, since we are denoting the element 2 as -1 just note the following properties that 1 + 1 will be what? So, 1 + 1 will be -1 and -1+ - 1 will be 1. So, the remaining equations are pretty much obvious so this is the only non-obvious computation that is happening here. So, the next idea is that a, so P in f 3 x 1 through x n is a polynomial in n variables over the field f 3. So, it takes its values from f 3 and the coefficients of the terms also lie in this field f 3.

So, we say that P is a proper polynomial if for x 1 through x n belonging to 0, 1 P of x 1 through x n also lies in 0,1. So, suppose if we restrict the variables to take values only from 0, 1 since we are working in this field f 3 there is no guarantee that the value of the polynomial will also lie in 0, 1. But if that happens then we will call such a polynomial to be a proper polynomial, nothing much.

I mean just so one good thing about proper polynomial is that you can think of this as a Boolean function, you can think of this as a function which is a so Boolean function is a function which takes values from 0, 1 and it outputs a value in 0, 1. So, proper polynomial although it is allowed to have values in the field f 3 it has this restriction that it is in 0, 1. So, it has a nice connection to Boolean functions.

(Refer Slide Time: 28:13)


Lem
of deg ≤ (2t) s.t. p differs from
C on at most size(C) · 2^{n-t} inputs
Pf.
x_1 x_2 x_n
1) for input edges assign the monomial
x_1 to var x_1

So, the first Lemma that we will show is that as I claimed earlier that if C is an A C 0 circuit then it can be approximated well using proper polynomials. In other words there exists a proper polynomial of degree at most so this is for all t so given an integer t greater than 0, there exist a proper polynomial of degree 2t to the power d. Such that proper polynomial let us call it P, such that P differs from C on at most size of C times 2 to the power n - t inputs.

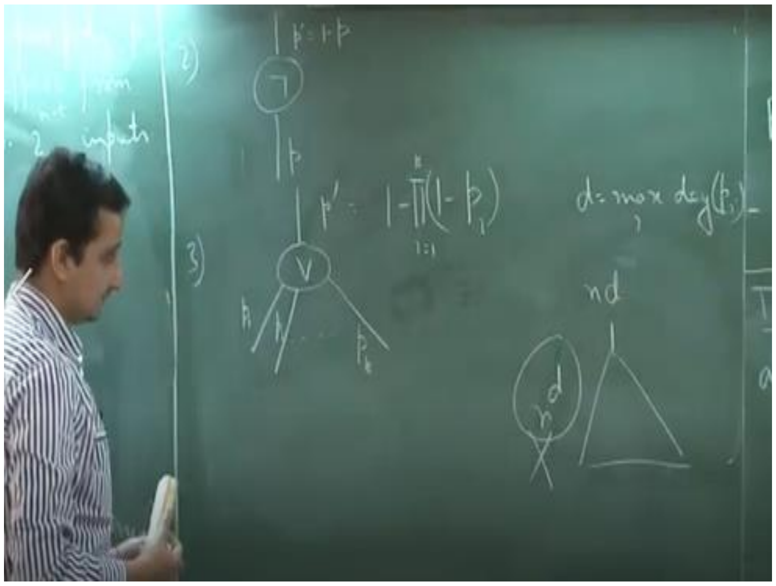
So, totally since we are looking at n variate functions the total number of possible inputs can be 2 to the power n, so what we are claiming here is that if you start with an AC 0 circuit then for any t greater than 0. There exists a proper polynomial having degree at most 2t to the power d such that P will differ from the circuit on at most so many inputs. So, any questions so far? C is the same circuit that we started off with.

So, what we want to show is that if there is a circuit C of depth t that computes parity then size of C is greater than this, so this is what we want to show. So, we start off with a circuit C having depth d that computes parity on n variables and our goal is to show that size of C will be greater than this quantity. So, the first claim that we make is that for any t greater than 0 there is a proper polynomial that approximates t up to a very large factor.

So, how do we do this? So, essentially this is a proof by an inductive construction so what we do is given C, so let us say this is our circuit C and these are the input gates so we these are the input edges and finally we have an output edge. So, what we will do is we will inductively associate a proper polynomial with each edge of my circuit. So, the polynomial that we will associate with the input edges is basically the polynomial x i.

So, for x 1 we associate the monomial x 1 where x 2 will give the monomial x 2 and for x n will give the monomial x n. So, the first thing is that for input edges assign the monomial x i to variable x i and we will argue for each of our cases that it gives us a proper polynomial. So, clearly x i is a proper polynomial because if I mean it is the identity polynomial. So, if it is 0 or 1 it will just give back the same value.

(Refer Slide Time: 34:02)



Secondly suppose if we have a NOT gate and let p be the polynomial associated with the input of the NOT gate. So, what do we associate with the output of this not gate? We want to associate a polynomial that will somehow compute what the NOT gate does. So, we will associate 1- p so let us call it p prime so this is equal to 1- p and again, so again clearly p prime is also a proper because if p is proper then 1- p again takes value in 0 and 1.

And the second thing is that if the degree of the input is if degree of P is d then degree of 1- P is also d. The third case is the non-trivial case, so suppose if we have an OR gate that has on its input edges the polynomials p 1, p 2 so want let us say p k. So, we want to associate a polynomial to the output edge. So, what is the obvious polynomial to apply in this case? So, I want to write down a polynomial p prime such that p prime will evaluate to 1.

If and only if at least one of these evaluate to 1. So, we can set that as one minus you can just do this going from i is equal to 1 2 k and that will compute it correctly. But let us just analyze what will be the degree of this polynomial? What can be the degree of p prime? How large can it be? So, in general since it is an AC 0 circuit k can be as large as n, so this degree therefore can be as large as so suppose the maximum degree of the p i is where d.

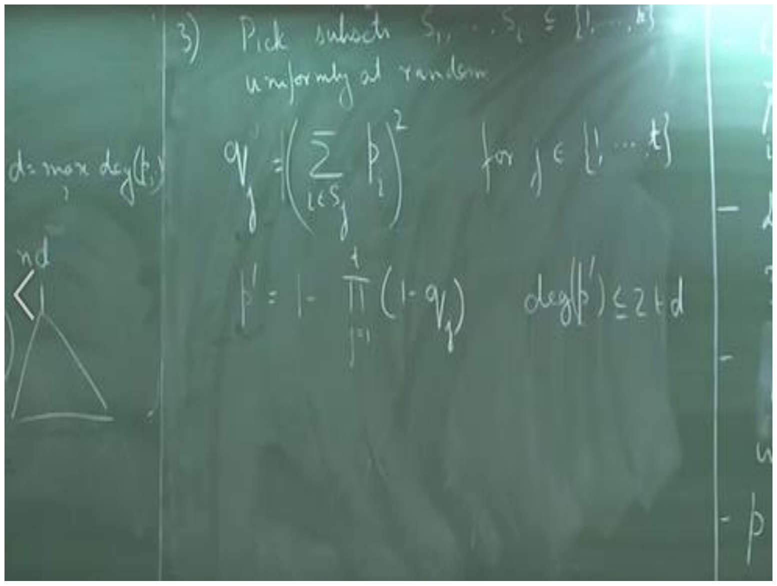
So, then this can be as large as how much d to the power k or I mean k can be as large as n so this can be as large as d to the power n. Sigma of pi so suppose you have only three gates that have one and the remaining gates are all 0, so sigma of the pi's is will be sum of those three gates that have one and since we are operating over f 3 that will output 0, 1 + 1 + 1 is 3 and modulo 3 that is 0.

So, summation will not work I had summation or something or some other computation which had such low degree complexity indeed work, then it would have been good for us but this is kind of the smallest polynomial that is that will compute out. So, the problem here is that so now let us analyze what will be the output of or what will be the degree of the output polynomial. So, we have a depth d circuit and, in each depth, we can compute an OR gate.

So, we start with at the base level where we have all degree ones so from degree 1 no this is not d to the power n this is d n, because with each product I am just adding a degree d to it. So, what will be the suppose if I have a circuit C and if we apply this polynomial to an OR gate what can be the potential degree of the polynomial at the output edge. We are multiplying it by n, so we start with one and we have a depth that is equal to d, so maybe d is not the correct thing to use.

So, finally what can it be? So, it is n times n times n times n so it is it can go up to n to the power d. So, this is quite large, so this is clearly something that is not allowed because ultimately what we want is to have a polynomial that has degree at most 2 t to the power d for any t greater than 0 and n can be much larger than the t that we start off with. So, this thing does not work.

(Refer Slide Time: 39:48)


3) Pick subsets $S_1, \dots, S_t \subseteq \{1, \dots, k\}$
uniformly at random
$q_j = \left(\sum_{i \in S_j} p_i\right)^2$ for $j \in \{1 \dots t\}$
$d = \max_i \deg(p_i)$
$p' = 1 - \prod_{j=1}^{t} (1 - q_j)$ $\deg(p') \leq 2td$

So, what we do in the case of or gates is that suppose if n or if we have an or gate which has inputs p 1 through p k, so we will pick subsets S 1 up to S t. So, these are subsets of 1 to k uniformly at random. So, whatever that p that is given to us we will pick t subsets and they will be chosen uniformly at random that is for each subset we will independently decide whether to put on a particular index i or not.

So, I mean we can just do a coin toss I mean for each k we will just do a coin toss if it comes out head we will put in that i and if it comes out tail will not put in that i. And this way we will construct our t subsets and then we will define q j as equal to sigma i belonging to S j p i whole square for j in 1 to t. So, for each of those subsets I will sum up the corresponding p i that I mean such that the indices of those p i are obtained from that subset and then take the square.

And now I will just apply this OR function so now I will define this p prime to be equal to 1 minus product of the j s, so j going from 1 through t 1 minus q j. So, clearly what we see here is that now p prime is basically an or of these q j's so if any of these q j's evaluate to 1 then p prime will evaluate to 1 and if all of the q j's are equal to 0 then p prime will be 0. And if you just analyze the degree if we have this polynomial as the output polynomial of an OR gate.

And if let us say d is the maximum degree of our p i's then the degree of so what will be the output, I mean the degree of the output polynomial. So, let us say the maximum degree of the p i’s is d. So, what will be the degree of q j? 2 d. And what will be the degree of p prime? 2 td. So, the maximum degree of p primes is 2 td. So, now if you have a depth d circuit so we start off at the base layer with monomial that is polynomials of degree one.

So, by the time we go to depth d the polynomial's degree will not be more than 2t to the power d. So, we will say stop here and we will complete the error analysis next time. But this is just one part of the proof. In the next part we also have to show that parity cannot be approximated in this manner. So, here what we are saying is that AC 0 can be approximated by a proper polynomial will show next that there is no proper polynomial that approximates parity in a well fashioned.