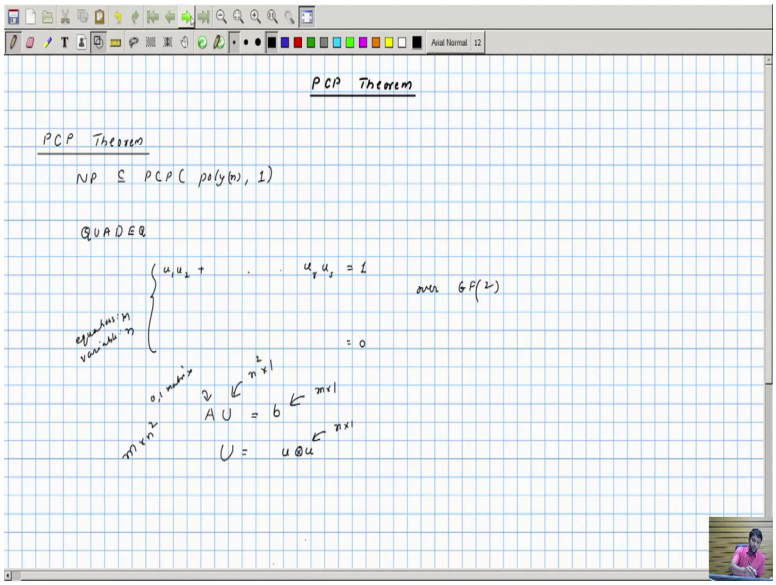
Randomized Algorithms
Prof. Benny George Kenkireth
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 38
Probabilistically checkable proofs – III

(Refer Slide Time: 00:28)

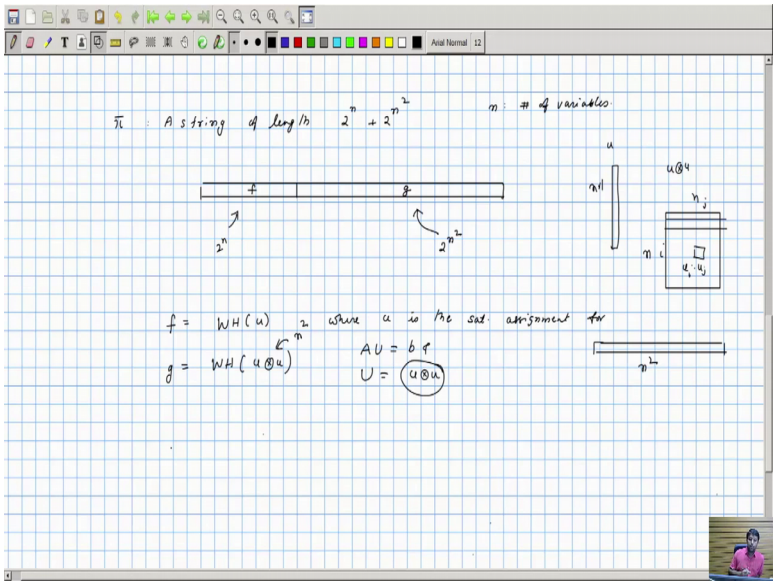


So, we wanted to prove with the version of PCP theorem. It says that NP is contained inside PCP poly n comma 1 ok. In order to do this, we were using Walsh Hadamard code words. See we are looking problem of QUADEQ; there in, you are given a family of quadratic equations ok. So, we have m equations and the number of variables is n ok. And these equations are over GF 2 which means, all the operations that you are doing are Modulo 2 operations. Now, this as we can think of this as solving the system of equations A U is equal to b, where A is a matrix. It is an n sorry, it is an m cross n square matrix and U is an n square cross 1 matrix and b is an m cross 1 matrix ok.

So, we want to solve this equation such that, the matrix U is equal to the tensor product of some n cross 1 matrix or n cross 1 vector ok. So, QUADEQ is NP complete. So, this we can somehow produce a certificate to show that A U represents b and U is a tensor product of small u with itself. Then, we can say that if it produce a PCP certificate which says this.

Then, we can show that NP is contained inside PCP with this particular parameters ok. So, one certificate for QUADEQ would be the assignment of the variables ok, but if you look at the assignment of variables what variable takes what value. That will be too long check 0 to query all the bits of the proof. Instead, we will expand this assignment into a large word or a large string such that by following a certain protocol on the string. We can be assured with a reasonable accuracy that A U is equal to b and U b equal to U answer U as a solution ok.

(Refer Slide Time: 03:42)



So, the proof in PCP, the verifier will have access proof and that proof we will think of as pi which is a string or binary string of length 2 raised to n plus 2 raised to n square ok. So, this long string we will think of it as having one part of length 2 raised to n and the other part other will be of length 2 raised to n square. And you will assume that the first part, is a linear function f and the second part is a linear function g. In other words, this linear functions are essentially nothing, but the Hadamard codes or Hadamard encodings with certain the strings ok.

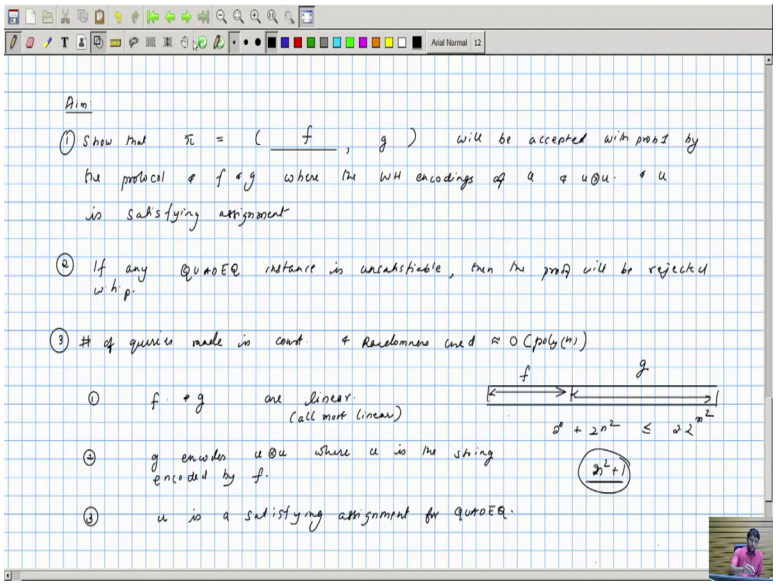
So, here n is a number of variables ok. So, the function f we will see it as the Walsh Hadamard encoding of u where, u is the satisfying assignment. For the quadratic systems A U equals b and U equals u cross u ok. So, if there was a satisfying assignment, then the Hadamard, Walsh Hadamard encoding of that is the first part and the second part we will see as the Walsh Hadamard encoding of u tensor u ok.

So, u tensor of u is going to be thought of as a matrix which is being converted into a single string ok. So, if u is a vector is a n cross 1 matrix then u tensor u will essentially be a matrix; n cross n matrix whose i jth entry will essentially be u i dot j ok, and this matrix if you think of it as a single vector of size n square by some canonical encoding, you can think of that single vector as the second part of the proof ok.

So, take the matrix of size n square cross n square and then b will append row after row and what you get there is the string g ok. So, string g you can think of it as you tell any position in the u cross u matrix, in the u tensor u matrix that can be queried by means of the function g.

u cross u is an n square sized vector the Walsh Hadamard encoding of that vector will be what is g ok. So, that is the way the proof string is going to be. Now once this proof string has been given the verifier is going to query some part of the proof string and if this is indeed the correct pro string, in the sense it was the encoding of a satisfying assignment then your verifier or the protocol will say that it is in fact, a satisfying instance and if it is not a satisfying instance we will show that whatever be the proof that is being supplied with high probability that proof will be rejected ok.

(Refer Slide Time: 07:46)



So our aim is as following as follows. Show that pi equals lets think of it as f followed by g will be accepted with probability 1 by the protocol. If f and g were the Walsh Hadamard encoding of u and u tensor u and u is a satisfying assignment. And the second

part is, if any formula or QUAD formula or QUADEQ system, for any QUADEQ instance is unsatisfiable. In other words, its every possible u is a bad, I mean if you look at any possible you know all those will give rise to incorrect proof strings ok. So, those are precisely the bad instances or the unsatisfied instances. So these unsatisfiable instances will result in the poof being rejected. So, if any QUADEQ instance is unsatisfiable then, the proof will be rejected with high probability and the number of queries made is some constant and randomness used should be order of poly m ok.

This is what we need to show we have to design a protocol which satisfies these three requirements ok. So, we will give the overall high level view of a protocol. Protocol basically checks the three things the first thing that the protocol checks is that f and g are linear ok. So, we are given a string of length 2 power n plus 2 power n square. If you look at the first 2 power n bits is the proof, that will is what we call as an f and this next 2 power n square is what we call as g you will verify that these two are linear. That is the first part ok. So, in other words we are checking that these are let us say Walsh Hadamard encoding of some string.

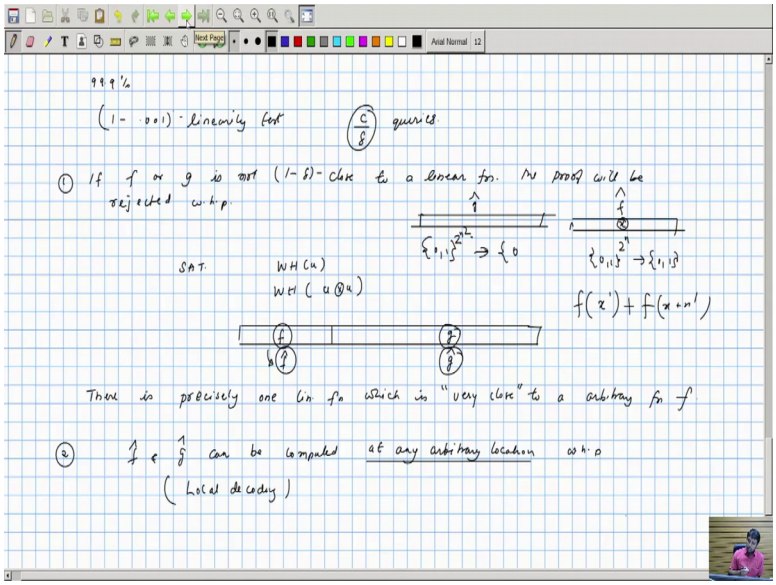
Argued earlier, that checking whether it is a Walsh Hadamard encoding is very difficult. It will require too much time and therefore, instead of linear we will just we will be satisfied if we can verify that they are almost linear ok. If they are not almost linear then our algorithm will quickly find out that they are not almost linear ok. The first verification is that they are almost linear and then we will check if g encodes u tensor u where u is the string encoded by f. And the third part we will check is u is a satisfying assignment for QUADEQ ok. These are the three things that our protocol will check and verify that that number of queries made by this protocol is constant and the randomness used is o poly n ok.

Now, if you want to look at one random location inside this string each location can be. So, you can think of this to be let us say less than 2 into 2 power n square and each location if you want to access, each address each location has an address of size n square plus 1 ok. The address of the each location is surely less than n square plus 1 because look at all with strings of length n square plus 1 that will be greater than 2 into 2 raised to n square. So, each address is going to be of length n square plus 1.

So, if you are querying only constantly many locations then, each of those addresses some poly n and therefore, it is a querying random locations you querying just constantly many random locations the total amount of randomness would be limited ok. The number of randomness used is limited to the extent of 4 poly n. So, that is we will just ensure that the query is going to be constant number of bits ok.

So, and if these tests are passed that is if f and g are linear and then g encodes u cross u and uses satisfying assignment then it means that the formula is satisfiable. If any of these tests fail then, we will declare the formula as unsatisfiable. And for all the satisfying instances these tests will be satisfied for all the unsatisfied instances these tests, at least one of these tests would fail with high probability. That is what we will show. Now, how do you check if f and g are linear? We need to check if they are almost linear because of the properties of Walsh Hadamard encoding, we could just check at a few locations ok.

(Refer Slide Time: 14:35)



So, if we wanted let us say n an accuracy of 99.9 percent, we can we look at a 1 minus 0.001 linearity test. This would require c by delta queries ok. So, this c and delta it really does not depend upon the input size ok. This is of our choose the 001 or the delta is our choice. So, depending upon the accuracy that you want you can decide how many times you will have to run the linearity test ok.

So, you can run the linearity test and once you are done the linearity test f and g you can on both these parts, the first 2 n bits and the next 2 raised to n square bits if you run the linearity test and if the functions were actually linear then of course, the linearity test would say that they are linear. They are not actually linear if f or g is not 1 minus delta close to a linear function. Then the property of linearity test states that would be detected with high probability ok. The proof would essentially be rejected with high probability and if the instance was in fact satisfiable you can just give the correct satisfying assignment u or its encoding ok.

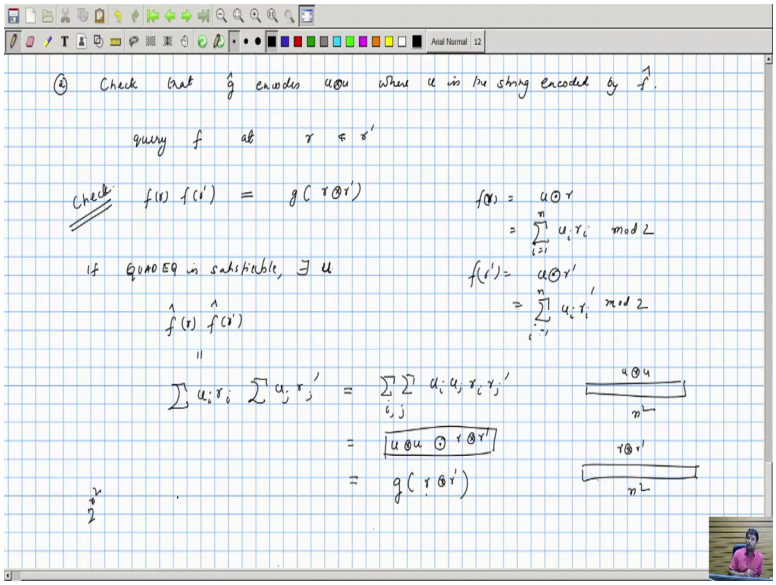
So satisfying instance, you can give WH of u and WH of u tensor u ok. If these were contaminated and presented, then the test would surely be a success, but if they were not even 1 minus delta close to a linear function, the linearity test will detect with very high probability ok. So now, what we have is we can assume that our bit string that is supplied to us has two parts and the first part is very close to a linear function the second part is also very close to a linear function ok.

So, if you assume that the adversary is trying to cheat us it has to surely give some functions which is very near to a linear function and some other function which is very near to a linear function and that is precisely one linear function which is very near to any almost linear function. This is precisely one linear function which is very close to a arbitrary function. So f is not suppose f is not a linear function then in the vicinity of f the neighbourhood of f that is only one function which is linear nearby means, they agree on many bits.

So, that function let us call it as f cap and let this be g cap. So, if the adversary is trying to cheat us he has to give some function which is not linear, but even from that not linear function we can infer a function f hat and g hat which has to be linear ok. So, if our test said that these functions are linear, we know that they are encoding the function f cap and g cap ok. So, for the next test is the first test the next test we can assume that f cap and g cap is what we have. I mean, have f hat and g hat were what were supplied to us ok. If you want to locate the value of f hat at one particular location we can do a local decoding. So, f hat and g hat can be computed at any arbitrary location ok; f hat is a function from 0 1 to the power n to 0 1 ok.

So, and this function can be I mean you can compute the value of this function at any point with high probability. You can compute it correctly with high probability because of local decoding ok. Wherein, if you have to find the value at x, you find the value at x prime and x plus x prime ok. Add the values you will get the value at x ok. So, that is called as local decoding. So, f hat and g for g hat that is going to be a function from 0 1 to the n square to 0 1 sorry, 0 1 to the 2 raised to n ok. So, this 0 1 to 2 raised to n (Refer Time: 20:23). f hat is a function from 0 1 to sorry, f hat is a function from 0 1 to the 2 raised to n to 0 1 to 0 1 it is a string of 2 raised to n. And g hat is a function from 2 raised to n square to 0 1 ok.

(Refer Slide Time: 21:02)



Now, we need to verify that g actually encodes u cross u. That u is a string that the second check is basically check that g hat encodes u cross u or u tensor u where u is the string encoded by f hat ok. So, what we have to do for that is query f at say two locations, at r and r prime ok. So, they can compute at f r and f r prime ok. And this should be equal to g of r tensor r prime ok. So, this is the check that we will perform. So, if this check fails is equal to then we will say that this test is passed otherwise; we will say this test is failed. We want the certain accuracy we can run it multiple times ok. Let us look at one test and if the function f and g were of this kind ok.

So, if let us say QUADQ is satisfiable. Then, there is a u that u can be given by the prover in to the proof string is basically reduced by means of u. And if you look at f r and

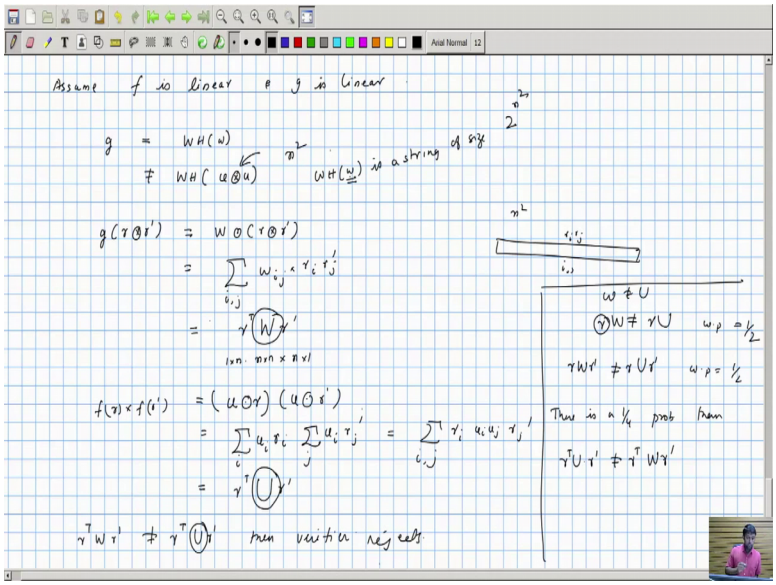
f r prime, what are these? f r is nothing but u product with r ok. It is an encoding of the string u an at location r that is just u dot new product with r. This product was defined as u i r i summed over i varying from 1 to n and computed mod 2 ok. And f r prime is going to be u product r prime and that is going to be summation i varying from 1 to n. u i r i prime mod 2. Take the product of this, the product would be ok.

So, we were doing this for f r. So, basically querying the proof string and looking at location half ok, but we know that this is going to be an almost linear function and the actual function was f hat. So, if you think of a function f hat this must be equal to summation u i r i summation u j r j prime and this, by rearranging you can see that this is equal to double summation over i comma j u i u j r i r j prime ok; and this is nothing but if you look at the tensor product as u with itself.

That is going to be a string of length n square and if you take the tensor product of r with so, this is u tensor u and if you take r tensor r prime, you will get another string of length n square and this value is going to be just the product of that ok. So, if that is the, so this product is just its a u tensor u multiplied with r tensor r prime and that is just nothing but in the string g whatever is at the r tensor r prime position ok r tensor r prime is some particular location. This g is a string of 2 raised to n square and each position is indexed by 2 vectors r and r prime. Once you variate it over all positions, you will get the entire string.

So the value is nothing, but g r tensor r prime. So if the string so what this shows is if the string supplied by the prover or the proof string is actually a linear function f cap, then this test would automatically be satisfied. Now, it was not a linear function which had these properties ok, g did not encode u, u tensor what will happen? So, we will look at that particular case.

(Refer Slide Time: 26:08)



We can assume that f is linear ok. We know it is almost linear and therefore, there is a unique f hat and that f hat can be inferred from this function and the since we use the local decoding, if the advisory or the prover was trying to cheat then we could basically detect that with high probability.

So, we will assume that once the first test is done, f linear and g is linear ok. So, in this if the prover tries to cheat we will catch them. So, this once we have done the first test this is something that we can assume. And now suppose, assume that whatever is a string that f inputs that does not that string if you take a tensor with itself that string is not encoded by g will this happen. Now, if that is the case then what we will have is this string that is encoded by g ok. So, g is the Walsh Hadamard encoding of some particular string we call it as W, because its linear it is the encoding of some particular string.

So, let us say g is equal to W and Walsh Hadamard encoding of W and is not equal to Walsh Hadamard encoding of u tensor u. So, u tensor u is some string of length n square, its encoding is going to be some string WH of is a string of size 2 raised to n square and. So, this W is not equal to u tensor u. That is the situation that we are. Want to see we can detect that ok. So, look at g r tensor r prime. You were querying it as some up random location. Now g tensor r r prime will essentially be equal to W product with r tensor r prime. So, if you write this in expanded form this should be equal to summation i, comma j W i j ok. i j will take n different values.

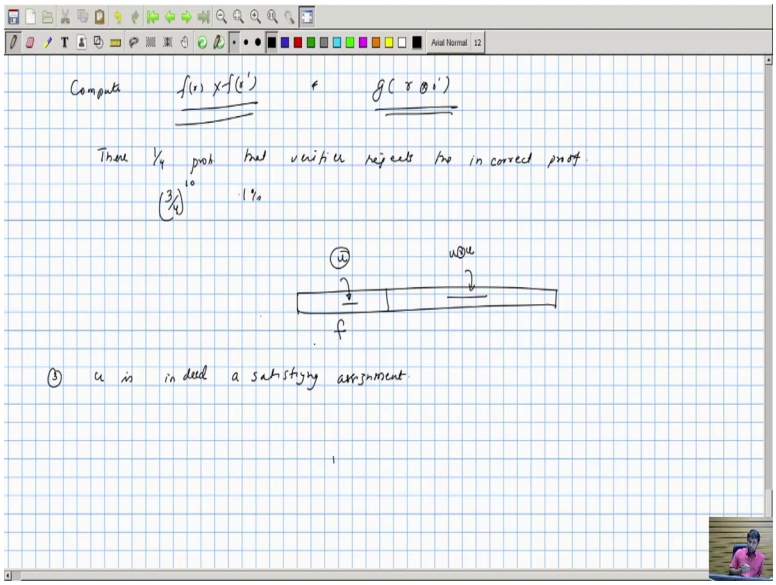
So, that is going to be n square different entries times r i r j ok. The tensor product of r with r prime is going to be some vector of length n square whose i jth entry will be r i r j ok. Sorry, r i r prime j. So, this is the product of g r r prime. We can do this because g is a linear function any linear function as a Hadamard, Walsh Hadamard code word and Walsh Hadamard code word is produced by taking product of this kind and this is nothing but you can write this in matrix form as vector r multiplied by W multiplied by r prime writing in this way thinking if this is r transpose ok.

So that is a 1 cross n matrix into an n cross n matrix into an n cross 1 matrix ok. So, you can think of this in the following way. If you look at f r times f r prime, f being a linear code that is going to be equal to u and for some particular u it is going to be u times r and this multiplied with u times r prime and this is equal to summation u i r i. i going from 1 to n and j going from again 1 to n, u i r j prime. This by rearranging terms is nothing, but summation of all possible values of i j r i u i u j times r j prime and in matrix form this can be written as again r U or r transpose U r prime. There U is now a matrix which is a tensor product of U with itself ok.

So, what we are assuming is that this W is not equal to U ok. W is not equal to U we want to say that this test will detect it with high probability ok. So, if r W r prime is not equal to r U r prime then, a verifier would have reject it. We want be look at the probability of this happening ok. So since, so again here what will come handy is the random sub some principle. So W is not equal to U you take any word random word r r W is not going to be equal to r u with probability equal to half ok. So, U is the tensor product of small with itself and W is some other string of length n square ok.

So, since these things are different strings. If W is not equal to u then that r W is not equal to r U with probability half and for each of those r. So look at any particular r chosen randomly there is a half probability of that when multiplied with U and W is different string. For each of those strings r W r prime is going to be not equal to r U r prime with probability equal to half ok. So, total for any arbitrary choice of r and r prime there is a one fourth probability that r U r prime or r transpose u r prime is not equal to r transpose W r prime ok.

(Refer Slide Time: 33:03)



So basically, what that means, is compute f r times f r prime and g r tensor r prime for some random values r and r prime. If the functions were not proper if the verifier was whoever was trying to cheat, then these are going to this product is going to be different from g r tensor r prime with probability greater than or equal to one-fourth.

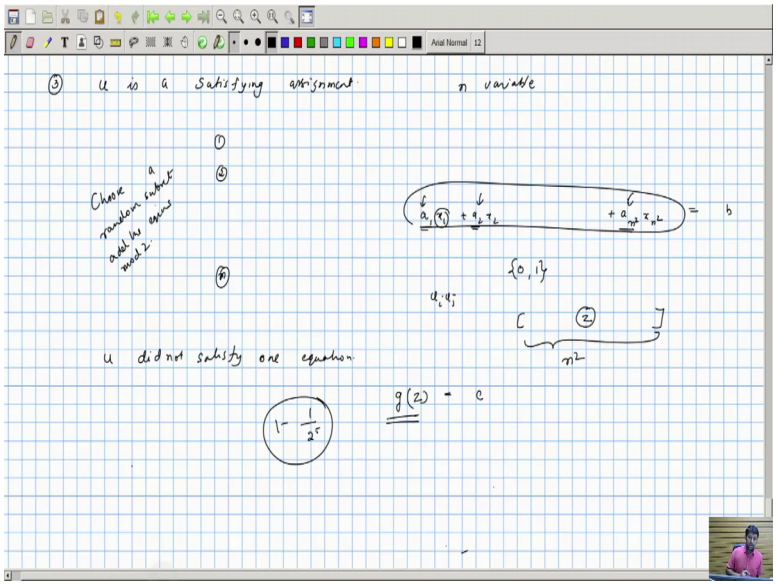
So, there is a one fourth probability that the verifier rejects the incorrect proof ok. Here incorrect would mean the string g is not a proper encoding of the string that f encodes that happens with probability one fourth. You can try it for ten different trials there is a it's only a three fourth chances of acceptance. So, if you try 10 times then the chance of acceptance is as low as three fourths raised to 10 and that is something like less than a 0.1 percent chance ok.

So, we can; so now, if first step and the second step comes through; that means, whatever is being given as the proof its first part is a linear function second part is also a linear function and the first part encodes a particular string and the second part encodes the tensor product of that string with itself. The encoding mean the Walsh Hadamard encoding of those strings ok. Third check that we will do this string u is indeed a satisfying assignment ok.

This is easy to check if we were allowed to query lot of decisions ok. We could look at entire string u and when from the function f, if we were allowed to query n locations we can extract the assignment plug it in inside our quadratic equation system and see if it is

satisfied or not. But that is too many queries because the number of queries is now proportional to the number of variables, but we want to have constantly many queries.

(Refer Slide Time: 35:36)



Now, one way to check this would be to extract the satisfying assignment or the assignment from f and check the formula check the system of quadratic equations, but that involves too many queries you want to do only constantly a many queries. If the assignment had n variables, if you extract all the n variables that would require something like 2 n variables it is dependent on the number of variables in the equation but we do not want it to dependent on that.

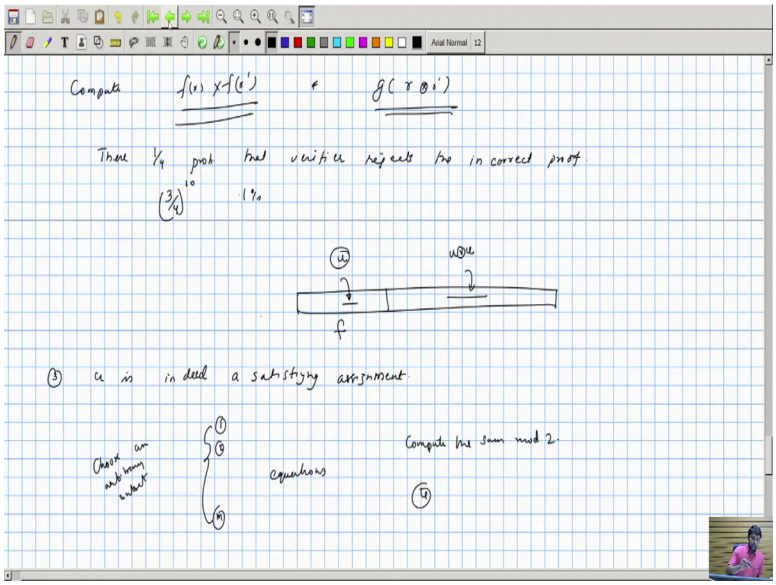
So, what we will do is we will look at the equations. Suppose there are n equations, we will extract a random subset and add the equations mod 2 and then you will get new equation and we will check means, equation that is just one equation ok if that one particular equation is satisfied or not? Suppose u is an assignment which did not satisfied even one particular equation. So, you did not satisfy one equation ok. Then when you extract it and sum it this particular way, there is a half probability that will be detected. So, that equate with the random equation that you have made will be unsatisfiable. This random equation was formed by selecting a random subset.

If u was not a satisfying assignment to the original system of equation then, the new equation will be unsatisfiable probability at least half ok, but this random equation let us say that is obtained by choosing a particular subject let us say a 1 x 1 plus a 2 x 2 plus a n

square x n square ok. Because each of the variables here is a term of the form u i u j and this should be equal to some particular number ok. But this is nothing but if you look at this vector all a i s are some element of 0 1 ok. So, if you look at the vector the n square long vector with these entries, I will call it as z ok. Your g z is precisely this particular sum ok. So, this sum is nothing, but g z. That z is a vector obtained by looking at these individual x ok. So, you can query at g z equation and if g z is equal to whatever was the sum of these equations ok.

If g z is equal to let us say c when you will add them up you will get some particular value that is equal then you will say that u is satisfying assignment otherwise, it is an unsatisfying assignment. So, by just doing one query you can check whether u is a satisfying assignment. If this u was indeed satisfying assignment the query would tell yes it is a satisfying assignment, if it is not a satisfying assignment that is at least a probability half of finding that this is not a satisfying assignment. You repeat it 5 times then the probability becomes probability of success becomes 1 minus 1 by 2 to the power 5 ok.

(Refer Slide Time: 39:22)



So, these three tests ensures that the function f and g are linear and g encodes the tensor product of whatever f encodes and whatever f encodes is the satisfying assignment. Once these three are checked, we know that our quad e q is satisfied ok. So, what we have argued is any instance of QUADEQ can be converted into a proof string pi and in pi, you

can just query constantly many locations and constantly many queries requires only poly n number of random bits ok. So, that brings us to the end of the proof of week P C P theorem.