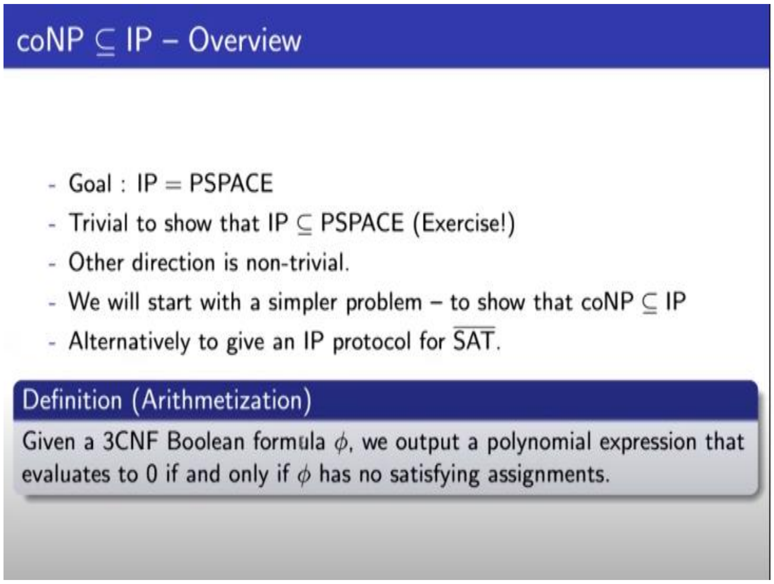
Computational Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture -34
SAT is in IP (Sumcheck Protocol)

So, welcome everyone. So, this is the 34th lecture. What we will discuss in this lecture is we will discuss some check protocol, which is a protocol to show that the unsatisfiability problem is in IP. So, basically we will be giving a protocol an IP protocol for SAT bar.

(Refer Slide Time: 00:42)



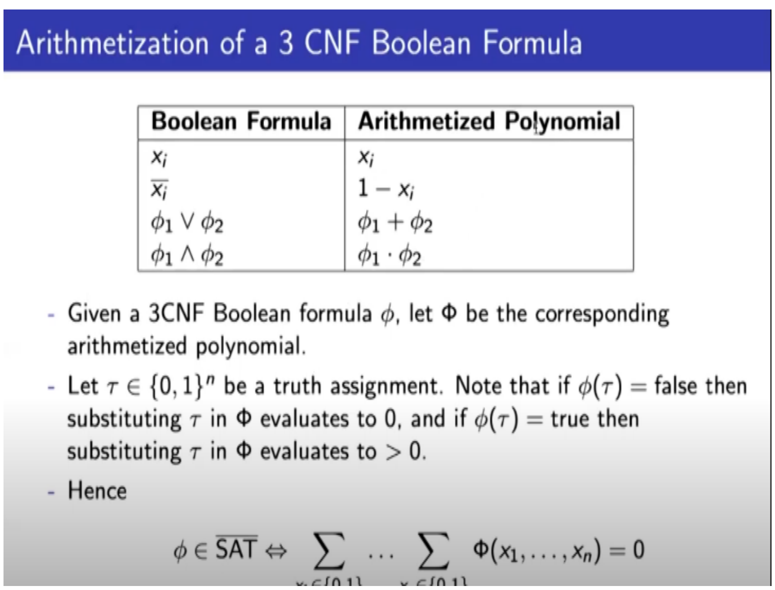
So, just to give a little bit of background and recap from the last lecture. So, what we saw towards the end of last lecture is this claim that IP equals PSPACE. So, recall that IP is the class of all languages for which there is an interactive proof and PSPACE is the class of all languages for which there is an algorithm that uses polynomial amount of space. So, we want to show that IP is equal to PSPACE. Now one direction of this claim is quite easy to show that IP is in PSPACE.

So, essentially what we have to show is that each round of a protocol so we can actually simulate in polynomial space each round of a protocol because I will just leave that part as an exercise. So, first figure out how do you simulate one round in PSPACE and then it can be easily generalized to how you would simulate n rounds in PSPACE. The other direction is basically the non trivial direction that is how do we show that PSPACE is in IP? So, instead of starting off with the proof that PSPACE is in IP what we will start off with is a simpler problem.

So, we will try to show that coNP is in IP first. And recall that SAT bar is a complete problem for coNP. So, essentially coming up with an IP protocol for SAT bar would be sufficient to show that coNP is contained in IP. So, that is what we are going to do today. So, how do we give IP protocol for SAT bar? So, the first step of coming up with a IP protocol is this concept called arithmetization of a Boolean formula.

So, given a 3CNF Boolean formula we will output a polynomial expression that evaluates to 0 if and only if phi has no satisfying assignments. What we will output is a polynomial. So, I will tell you what I mean by polynomial expression over here? So, it is basically a polynomial which is evaluated at certain points. And this expression will evaluate to 0 if and only if the input formula has no satisfying assignments. How do we achieve this?

(Refer Slide Time: 03:17)



How do we arithmetize a 3CNF Boolean formula? So, we will do this arithmetization in a recursive manner. So, firstly suppose if the Boolean formula is a single variable x i then the arithmetic formula the arithmetic polynomial is just the polynomial x i, the single variable polynomial x i. If the Boolean formula is not of x i then the polynomial will be 1 - x i. If the Boolean formula is the conjunction of two formulae phi 1 or phi 2 then the corresponding arithmetized polynomial will be phi 1 + phi 2.

And finally if the polynomial if the formula is phi 1 and phi 2 then the arithmetized polynomial will be phi 1 multiplied with phi 2. So, this is the most natural way in which we can arithmetize a Boolean formula. Now let us try to see what can we say about the corresponding arithmetized polynomial? So, suppose if we are given a Boolean formula phi let capital phi be the corresponding arithmetized polynomial that we get by doing this arithmetization.

So, what can we say about capital phi? So, consider any truth assignment. So, a truth assignment is basically just a n bit string consisting of zeros and ones. So, consider any truth assignments or truth assignment is one which is I mean, it is a bit string which we can think of as a string which is assigning values to every variable in a Boolean formula. Now if I plug in the truth assignment tau to the Boolean formula phi and if that evaluates to false then observe that substituting tau in the arithmetized polynomial capital phi will always evaluate to zero. Why is that?

So, if phi evaluates to false on the truth assignment tau what that means is that every clause so this is a 3CNF Boolean formula. So, there is at least one clause so it is basically a conjunction of clauses. So, there is basically at least one clause where all the three literals are zero if I plug in tau. Now if I look at the corresponding arithmetic polynomial it is a product of all these individual clauses.

Now if there is a clause which is evaluating to zero then if I look at the arithmetic polynomial corresponding to that clause, it is basically a sum of the three literals. If it is a sum of the three literals then note that, that linear polynomial will also be evaluating to 0 on that truth assignment. So, therefore if I multiply it with whatever I want, I mean, whatever I have after that it does not matter it will always evaluate to 0.

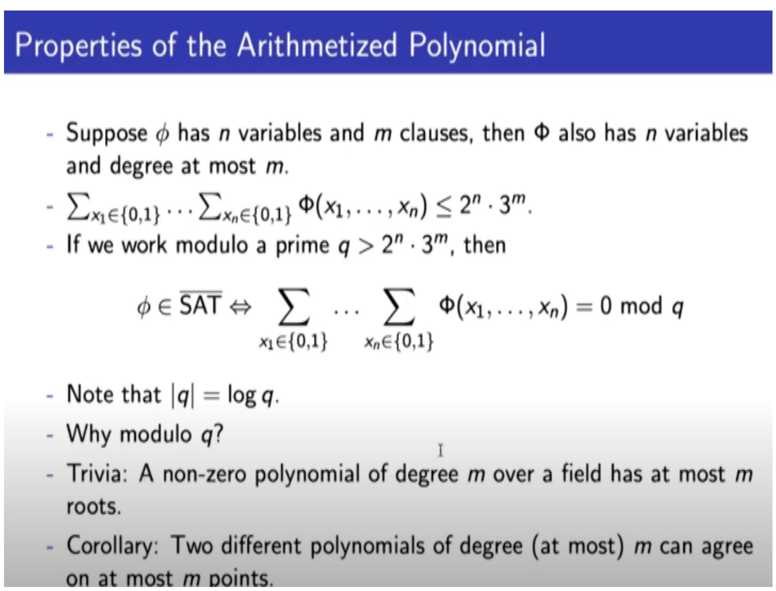
On the; other hand if tau makes phi evaluate to true then if I substitute tau in the polynomial capital phi that will evaluate to a quantity that is greater than zero. Again the reason is the same because each clause will evaluate to a positive quantity. Here also each clause the polynomial corresponding to each clause will evaluate to some positive quantity. And the product of all these positive quantities will give me a number which is again greater than 0.

Therefore, by this observation what we get is that if phi is unsatisfiable in other words, if there is no truth assignment which makes phi evaluate to true, then if I look at the sum of capital phi over all possible settings of x 1 to x n taking values 0 and 1 it will always evaluate to 0. So, basically I am taking the sum over all possible truth assignments. For each and every truth assignment it will evaluate to 0.

And on the other hand if I satisfiable then that means that there is at least 1 truth assignment for which it evaluates to true which means that on the right hand side if I plug in that truth assignment this polynomial will evaluate to a quantity greater than 0. It never evaluates to a quantity that is less than 0. So, there is no canceling off. It will just keep on summing up. So, that is what this thing means.

So, please convince yourself. So, what I said is not very hard to observe but please make sure that you do convince yourself that whatever I have said here is perfectly clear to you. So, this is what I meant is the expression. So, this polynomial expression evaluates to 0 if and only if phi is unsatisfiable. Now what else can we say about this arithmetic polynomial?

(Refer Slide Time: 08:40)



So, let us start with the degree of this polynomial. So, let us start with the number of variables.

So, that is easy. So, if small phi has n variables capital phi will also have n variables, that is quite easy to see. If phi is a Boolean formula on m clauses note that the degree of capital phi will be at most m because each clause is a linear polynomial. So, each clause the polynomial corresponding to each clause has degree one.

Now since there are m clauses I am taking a product of these m clauses. The degree will at most go to m. It can of course be less than m also if there are some cancellations but it can never be more than m. Now what is the maximum value of this expression? So, first of all, suppose if I plug in one truth assignment what is the maximum value that this polynomial can take? So, for that let us again go even further below.

What is the maximum value of a clause? So, a clause has three literals. So, the maximum value that a clause can take is 3 if all the literals, let us say evaluate to 1. Now there are m clauses. So, the maximum value of this polynomial on a given truth assignment is 3 times 3 times 3 all the way up to m. So, which is 3 to the power m. There is a maximum value of this one at 27 and the total number of possible truth assignments all possible values of x 1 up to x n is 2 to the power n.

So, the maximum value of this entire expression is at most 2 to the power n into 3 to the power m. So, what we do here is we will be working with this arithmetized polynomial for the rest of this proof. And not only we will be working with this arithmetized polynomial, we will be working with this arithmetized polynomial modulo a prime q. So, we choose a prime q that is larger than 2 to the power n into 3 to the power m.

Now since q is bigger than this quantity then if this expression evaluates to 0 it is the same as saying that this expression evaluates to 0 module q. And note that when I talk about q I mean, although the value of q is exponential it is 2 to the power n to 3 to the power m, but the number of bits required to represent q is only log q which is still a polynomial. So, the question that arises is why are we working modulo a prime q? Why do we want to do this?

The reason for that is 2 fold. So, if we are actually working modulo a certain number it helps in maintaining the value of the same. It helps in maintaining the value of the intermediate expressions that can arise to be at most q. So, we never go beyond q. So, this entire expression, of course, we know will never go beyond q. It is at most 2 to the power n into 3 to the power m. But maybe there might be some intermediate expressions which might be going bigger than this number.

So, we want to avoid that. So, that is the reason why we always work modulo q because we do not know. So, initially, we do not know how large all of these things can be. So, if we fix this q and if we are always working module q, then we are guaranteed that no number that we will be considering will be bigger than q. That is the first reason and the second reason is that we want to actually work in a field.

And if we consider a prime number then f q which is the set of all numbers from 0 to q minus 1 actually becomes a field. So, I am not going to discuss what a field is, what are the properties of a field and so on. But you can actually look that up. I mean, you should know that but if you do not know you can look it up. So, just to give you some examples I mean, the set of all real numbers is a field, the set of all complex numbers forms a field, the set of all rational numbers forms a field and so on.

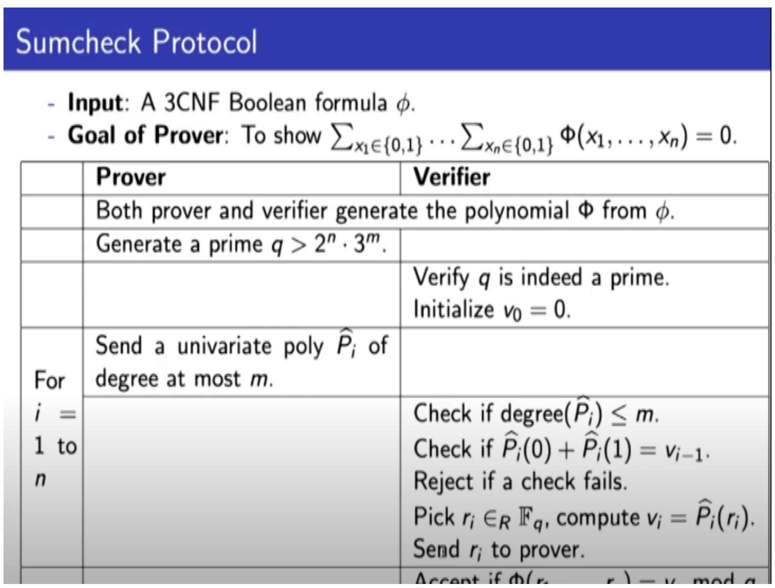
So, the field basically has a property where you have an addition defined in it, you have a multiplication defined in it, the addition has an additive inverse and multiplication also has a multiplicative inverse except for the zero element. So, all these properties are true for these sets that I give. On the other hand if you look at the set of all integers that actually does not form a field because although you have addition and multiplication defined in it and there is an additive inverse of each number but there is no multiplicative inverse.

For example, the multiplicative inverse of two is half; the multiplicative inverse of three is one third. But these numbers are not present in the set of all integers. So, just some examples any. So, we actually want to work in a field that is the reason why we choose a prime number q. Now,another trivia from algebra. So, if you have a non-zero polynomial of degree m over any field so that polynomial will have at most m roots.

So, any non-zero polynomial of degree m has at most m roots and a corollary to this is that if we take two different polynomials of degree at most m they can agree again on at most m points. Because if I just take the difference of these two polynomials the roots of that difference polynomial are essentially the points on which these two polynomials will agree on. Again this is something that I will of course not be proving in this course.

If you know the proof of this from your algebra course, very good. If you do not know you can just take this as a word of faith and again if you are interested in this you can of course look up any good book in algebra and you should be able to find a proof of this. So, there are many different ways in which this can be proven. But anyway, so this is essentially all of the mathematical knowledge that we will be needing for our proof.

(Refer Slide Time: 15:24)



Now let us come to the sum check protocol. So, what is the IP protocol for SAT bar? We are given as input a Boolean formula phi, a 3 CNF Boolean formula phi and we want to device a protocol between a prover and a verifier and the goal of the prover is to finally is to somehow convince the verifier that sum of all these terms so sum of x 1 over 0 1, sum of x 2 over 0 1 up to sum of x n over 0 1 of the arithmetized polynomial capital phi is equal to 0.

So, the prover somehow wants to convince that this formula is not satisfied. So, we know that if this is equal to 0 then the formula is not satisfiable and if this is not equal to 0 then the formula is satisfiable. We know that. So, the goal of the prover is to somehow convince the verifier. Now what is the protocol that they come up with? So, here is the protocol. So, we have the prover and the verifier.

So, first what they do is both the prover and the verifier will generate the arithmetized polynomial capital phi from small phi. So, that is quite easy to generate. Because it just has to substitute instead of NOT it will substitute 1 minus that variable, instead of OR it will substitute plus, instead of AND it will substitute multiplication and of course just a small point over here. The verifier actually does not mean to explicitly store the polynomial.

It can always generate this polynomial and if it wants to evaluate this polynomial on a certain truth assignment, it can always plug in those values onto that polynomial and it can evaluate it. So, that is not very difficult. So, it can always do that evaluation at any point of time. So, that is still a polynomial time computation. So, both the prover and the verifier generate capital phi. Now we will be working with capital phi.

Now what the prover does is the prover first generates a prime number q that is bigger than 2 to the power n into 3 to the power m. And note that the prover always has an arbitrary amount of computational power. So, that is what we assume. So, it can generate a prime number that is greater than 2 to the power n into 3 to the power m. But the verifier verifies that this is indeed a prime number that again can be done in polynomial time.

It can certainly be done in randomized polynomial time. But because of the primality result of Agarwal, Kayal and Saxena now we know that this can also be done in deterministic polynomial time. But I mean, the verifier has randomized powers. The verifier is an arithmetized polynomial time machine so it is not necessary that it has to do this deterministically. It can do this in a randomized manner also.

Anyway, that is just a side trivia. So, what the verifier does is it checks whether q is indeed a prime if not it will reject and then it initializes v 0 to 0. So, the best short for the prover is to actually send a prime q that is bigger than this number. Only then will the verified accept. Now the verifier initializes this v 0 to 0. After this what the prover and the verifier does is they play this game for n rounds.

So, for i is equal to 1 to n for each of these rounds, the prover will send a univariate polynomial P i hat of degree at most m to the verifier. The verifier will do some computation based on what the polynomial is and this thing will continue for n rounds. So, what does the verifier do again? So, note that P i hat is a univariate polynomial. So, it is a polynomial in one variable of degree at most m. So, what the verifier does is first the verifier will check whether indeed the degree of this polynomial is at most m.

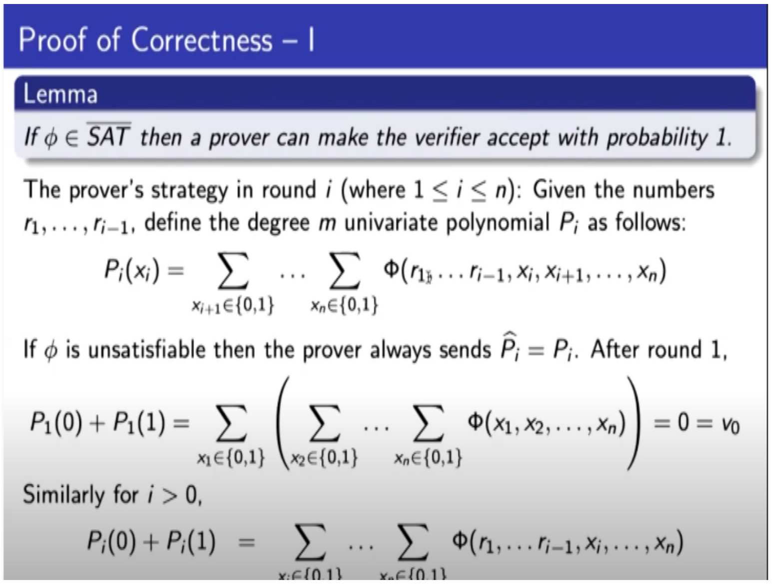
If not it rejects then it checks if P i hat of zero plus P i hat of one is equal to v i minus one or not. That is for the first round when i is equal to one it will check if P one hat of zero plus P one hat of one is equal to v 0 which is 0 or not. So, in each round it will check whether it is equal to v i - 1. If this check fails again it will reject. Suppose if this check succeeds then what the verifier does is it randomly picks a number r i from f q, any number.

So, this notation stands for picking a number uniformly and randomly from f q. It will plug in that number onto the polynomial P i hat that it has received from the prover. And whatever that substitutes to that is basically will be the v i, basically the v i for the next round. So, note that in every round it will use the v i from the previous round. So, it is this v i and it sends this number r i to the prover and the protocol continues.

So, after i is equal to 1 we go to i is equal to 2. So, prover will send a univariate polynomial P 2 hat of degree at most m to the verifier. Again the verifier checks if P 2 hat has degree at most m, then it will check if P 2 hat of 0 plus P 2 hat of 1 is equal to v 1 or not. If the check fails again the verifier rejects. If it succeeds then it will again generate a current random number r 2, compute the number v 2 and continue this manner.

Finally after n rounds the verifier will check if phi of r 1 to r n so phi of all these numbers r 1 up to r n is equal to v n mod q or not. If indeed this is equal to v n mod q the verifier accepts otherwise it will reject. So, this is actually the entire protocol to check whether phi is unsatisfiable or not. Now how do we prove the correctness of this protocol? So, to prove the correctness of this protocol we have to show both directions.

(Refer Slide Time: 21:53)



So, the one direction is that if phi is indeed unsatisfiable. So, note that in an IP protocol for a language if phi is indeed unsatisfiable that will lead belongs to the language then the verifier should always accept with probability 1. So, there is a strategy of the prover by which it can make the verifier except always. So, what is that strategy? How do we prove this? The strategy is actually quite simple.

So, the prover strategy is that in every round i it will pick the polynomial P i hat to be this particular polynomial. So, I define a polynomial P i univariate polynomial. So, observe that this polynomial will be a univariate polynomial. So, I define the polynomial P i on the variable x i as follows. So, for variables x i plus 1 up to x n I take the sum of phi by substituting both the values 0 and 1 for all those variables.

So, for x i plus 1 up to x n for all these variables I plug in both their values 0 and 1 and I take their summation. And for the variables x 1 up to x i minus 1 I plug in the values r 1 up to r i minus 1. So, the only indeterminate variable over here is x i. So, therefore this P i is actually a polynomial in x i. This is the only integral variable. All the other variables note that have been determined.

So, the first i minus 1 variables I am plugging in the value r 1 to r i minus 1 and for the last i plus 1 to last n minus i variables I am actually plugging in both the value 0 and 1 and summing over them. Now for if i is indeed unsatisfiable then what the prover will always do is it will basically be sending P i hat as P i. This is actually the best strategy of the prover because we will see that if the prover does this then the verifier has no option but to always accept.

So, after round one let us see what happens. So, if the prover sends P 1 hat as P 1, then P 1 of 0 plus P 1 of 1. So, this is actually the computation that the verifier is doing is sum of so this is basically P 1 now. It is basically the sum of 0 and 1 of x 1 of this expression. So, this expression is basically what? So, P 1 is nothing but this particular expression over here. Now this is actually equal to 0. Why is it equal to 0?

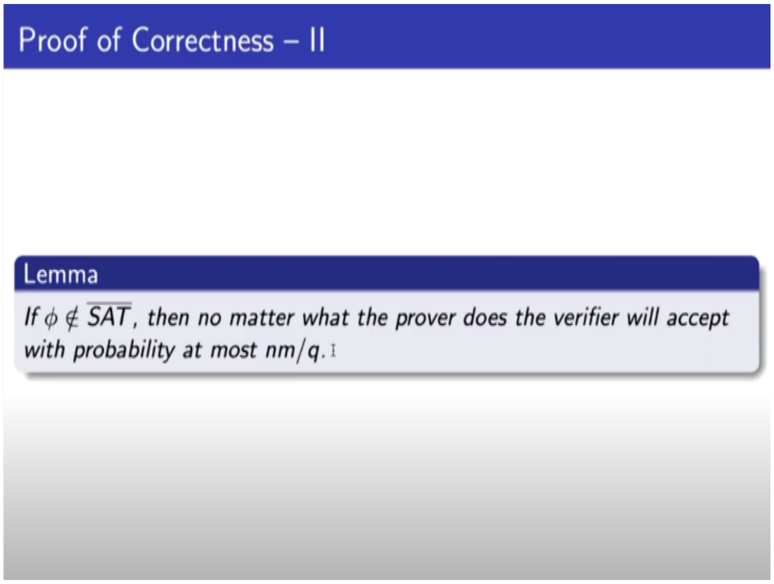
Because phi we know is unsatisfiable. If phi is unsatisfiable we have seen that this expression is this polynomial expression evaluates to 0 and that is what our v 0 was. So, after round one the verifier has to accept. Now what happens after a round i where i is greater than 0? So, take any arbitrary i greater than 0. The verifier does P i 0 plus P i of 1. This is actually equal to phi of r 1 to r i, x i, x i plus 1 to x n.

And I am summing over x i up to x n of 0 1. So, x i summed over 0 1 to x n summed over 0 1. So, I am just bringing in this x i inside. So, here if you look at this polynomial x i is not present but since I am summing over 0 and 1 so that is why I brought in x i. Now what does this evaluate to? So, note that this is exactly what P i minus 1 of r i minus 1 was from the previous round. Because this was the polynomial from the previous round;

And in this polynomial I am plugging in r i minus 1, which is what will give me P i minus 1 and r i minus 1 from the previous round and that is exactly the value v i minus 1. So, because of this the verifier again has no option but to accept whatever the prover is giving after every round i. And finally you can verify that even after the nth round when it does the final check even there the verifier will end up accepting.

So, please convince yourself. So, please work this thing on paper and convince yourself why the verifier is accepting? So, this is nothing but just rearranging the terms and I mean making sure that whatever is the value that is coming up is actually equal to whatever it got from the previous round. So, that is it. So, this is actually the easy direction that if phi is not satisfiable then the prover can always make the verifier accept by following this strategy. Now we look at the other direction.

(Refer Slide Time: 27:01)



So, if phi is satisfiable that is if phi does not belong to SAT bar then no matter what the prover does, no matter what polynomials the prover sends in each round the verifier will only accept with probability at most nm by q. So, the prover cannot make the verifier accept with the probability that is greater than nm by q. So, this is the non-trivial direction and not very nontrivial.

But we have to actually consider every possibility for the prover. The prover can actually send any polynomial it wants to the verifier because ultimately recall that the goal of the prover is to somehow make the verifier convince that this thing evaluates to 0. Now if phi is satisfiable then we know that this does not evaluate to 0. So, the verifier I mean, the prover has to somehow come up with these polynomials P i hats.

For example, if the prover sends P i hat which are essentially these polynomials then of course, there will be one round where the verifier will end up rejecting. It will not come to zero. So, the verifier will reject it. So, this strategy will certainly not work. Maybe now there is some other strategy. So, what we will prove next is that no matter what strategy the prover takes the verifier will never accept with the probability greater than this amount.

So, that is all that I had to say for in this lecture. So, we look at the proof of this second lemma in our next lecture. Thank you.