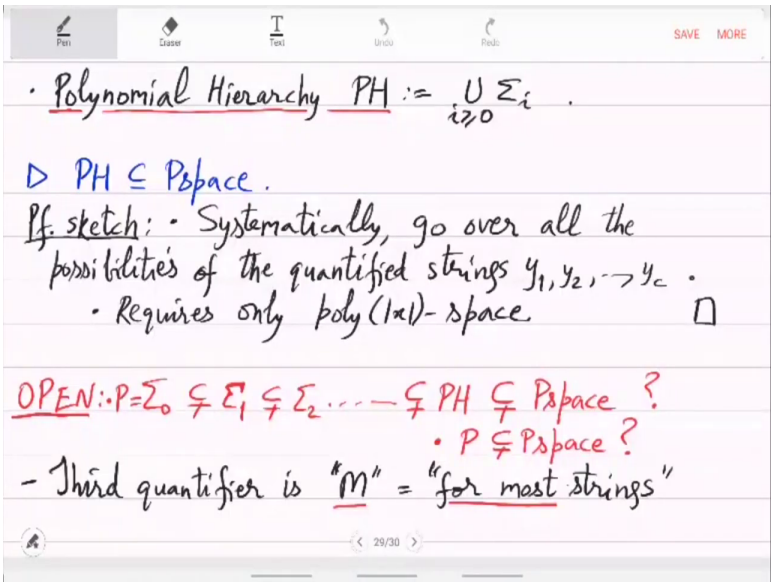
Randomized Methods in Complexity
Prof. Nitin Sexena
Department of Computer and Engineering
Indian Institute of Science – Kanpur

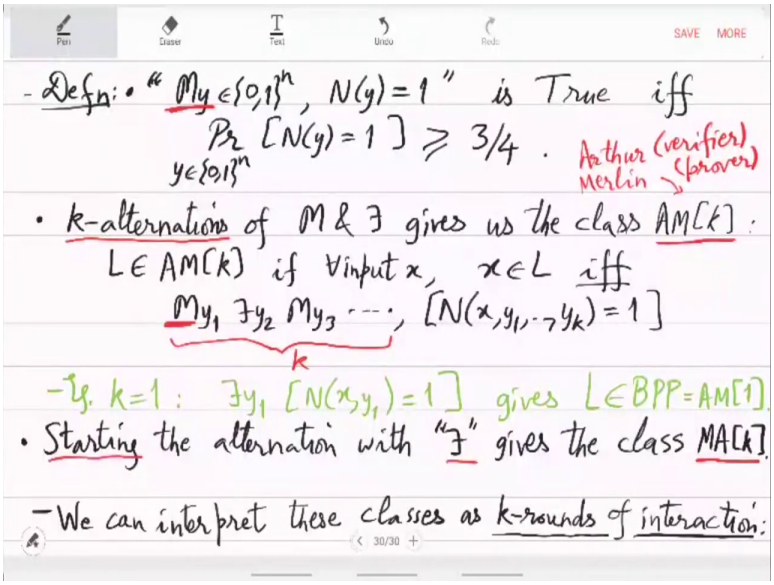
Lecture 04
IP = PSPACE

(Refer Slide Time: 00:15)



So in the last class we defined numerous classes new complexity classes based on quantifiers. So, they exist and for all quantifiers. So just to quickly recall Σ_1 we defined as there exist over P which is deterministic polynomial time and Σ_2 is the class there exists for all over P and Σ_3 is there exists for all there exists so alternating quantifiers. So this is a generalization of the NP class and all these classes are in P space it is an open question whether they are different.

(Refer Slide Time: 01:02)



Now we will introduce the third quantifier which is for most strings the m quantifier. So we say thatMy over N(y)is a true statement if N(y)= 1 for most of the strings y where most would mean 3/ 4 th. But you can also pick a constant above half this 3/ 4th is not very important. So now based on this we can define new complexity classes by using k alternations. So k alternations of M and there exist quantifier gives us the class AM.

So you will use they exist then most then there exist then most k of these k times and you will get the class AM[k].what is AM[k]? So this is a short form for Arthur Merlin. Arthur is the verifier and Merlin is the prover. So it is basically a game. So this is a game between Arthur and Merlin where there will be an interaction. So this is why I call this interaction based complexity classes. So the interaction will be that Merlin will send some string then Arthur will try to verify it and challenge Merlin then Merlin will another give another response and so on.

So it is a game between Merlin Arthur challenges Merlin, Merlin sends a response. After k rounds the goal of Merlin is to convince Arthur. So formally we say that a problem L or language LϵAM[k] if for every input x, x ϵL iff x is a string if and only if this game succeeds. So the game is; think of it in terms of quantifiers and these are key quantifiers. So think of k= 1 in that case what you are talking about is just there exist y_1 such that$N(x, y_1)$= 1.

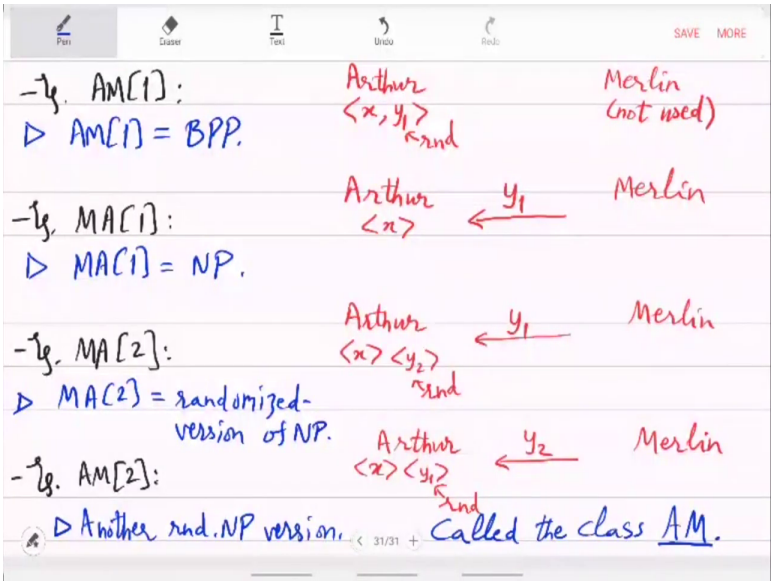
So you can think of it as a game between Arthur and Merlin where Merlin sends this response stringy_1 and Arthur will use the input x and this response to verify. So this gives the complexity class NP. So k =1 is NP it is the same class as so AM[1]= NP just 1 round. Now think of k =2. In k =2 there is now most quantifier. So, most y_1 there exist y_2. So now actually Arthur will randomly pick a string y_1 and then challenge Merlin after which Merlin will respond with y_2.

And then Arthur will run this algorithm$N(x, y_1, y_2)$ if the answer is 1 then Arthur will be satisfied otherwise Arthur will be dissatisfied and will actually not believe that x is in L. So you can think of this alternating quantifier business as a game between prover and verifier. So this is a useful intuitive way to understand these classes. So back to this definition starting the alteration with there exists gives the class MA.

So MA is slightly different from AM here we have started with M quantifier we started with m quantifier. So with that you get the AM class if you start with there exist quantifier then you will get the MA class. Intuition being that first Merlin is responding with a string y_1 and then the computation proceeds with Arthur. So we call it MA. If y_1 was a string randomly chosen by Arthur then we get the class AM.

So let us spend some time on the interpretation of these classes. So we can interpret these classes as k rounds of interaction. So how do you see that?

(Refer Slide Time: 08:09)



So let us start with again with AM[1]. So Arthur is just randomly picking y_1 and then running a deterministic polynomial time algorithm. And if you started with there exist that will be MA[1] which will give you NP. So you can see the big difference AM[1] is randomized polymer time algorithms while MA[1] is supposedly much harder it is actually NP. So the way you see this as a game or as interaction between Arthur and Merlin is as follows.

So in AM[1] Arthur has x and Arthur will randomly pick y_1 on one side you have Arthur. On the other side you have Merlin but Merlin is not used. Because in AM[1] there is only most quantifier there is no there exists quantifier. So Arthur just himself computes n (x, y_1) hence this is the same as BPP. Next example is MA[1] so what is the game? What is the protocol or interaction here? So here Arthur just has x and Merlin is challenged by Arthur.

So Merlin because MA[1] will start with there exist y_1 . So there exist quantifier associates with Merlin. So Merlin will send the response or the certificate y_1 and then Arthur will do the computation. And this game characterizes MA[1] is a NP, NP gets characterized by this game. What is MA[2]? What happens in 2 rounds? That would be more interesting. So, in here again Arthur Merlin, so Arthur starts with the string x remember MA will start with the quantifier there exist.

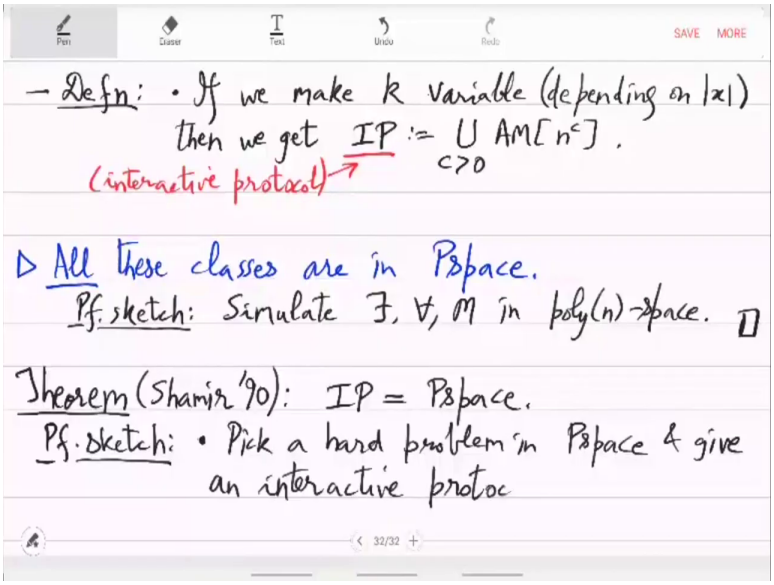
So Merlin will first send the response y_1 in MA[2] the second quantifier will be my_2. So this string has to be randomly picked by Arthur. So Arthur now has x string x input x then strings y_1 and y_2 where y_1 you should think over the certificatey_2 is a random string. And then Arthur will run the algorithm n on x y_1 andy_2. So the difference between this picture and the picture above MA[1] picture that is only in y_2 which is a random string.

So you can think of this as a randomized version of NP. So this is just like NP where the verification you are allowed to use are randomized polynomial time algorithm. And finally what is AM[2]? So again Arthur starts with the input x in A you have to start with most quantifier. So Arthur will pick y_1 this random string and then Merlin will send the response to Arthur's challenge which is y_2 and finally Arthur will do the computation on x y_1 and y_2.

So it is very similar to the above picture except that the order has been swapped. So in the above picture Arthur first got Merlin's response and then used the random choicey_2 in the second picture the order is swapped. So Arthur first guesses a first randomly chooses the string y_1 and then gets the responsey_2. So this is another randomized version of NP another randomized NP and it is called AM.

This is called AM and the class above MA[2] is called MA. So you already knew BPP, NP that is just AM[1], MA[1] when you do this interaction in 2 rounds then you get completely new complexity classes which are called MA and AM. Both are randomized versions of NP and we will now be working with them. So this is an important slide I will define this interaction based class which will be much bigger.

(Refer Slide Time: 15:42)



So if we make k then the interaction rounds. So if we make a variable so which means it depends on the size of the input till now it was constant 1, 2, 3 so on. But suppose we make it dependent on the size of the input. So if the input is longer then you have more interactions arbitrary many interactions. Then what do you get? Then you get the class IP := $\bigcup_{c>0} AM[n^c]$

So AM when key is large growing with input size n this gives us a really big complexity class called IP. So we have defined MA, AM and IP these three classes AM, MA seem to be much smaller than IP. As an exercise you can show that all these classes are in P space the reason is you can think in terms of quantifiers. So you just have normally many quantifiers alternating and there exist for all most.

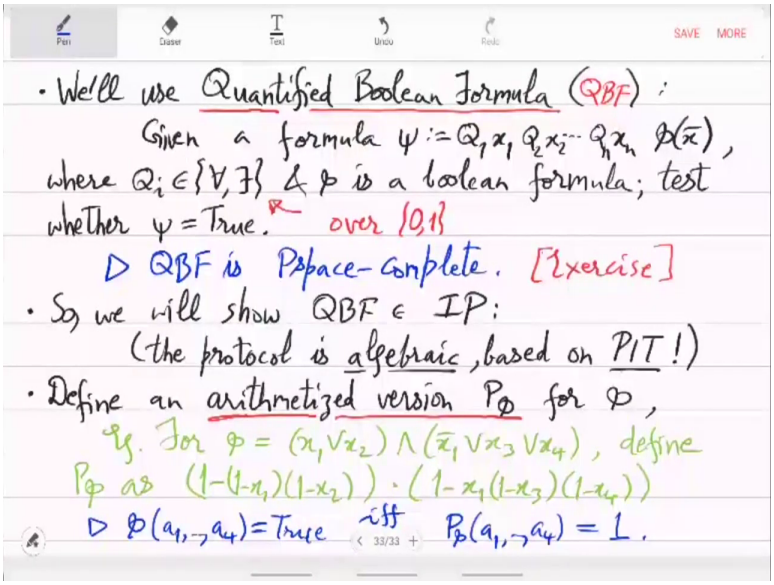
The choices that the quantifiers are making I mean the in that space you can go over all possible choices in polynomial space. So you can simulate there exist for all most in poly(n) space. Just go over all the strings of certain size and so you can get the value of there exist for almost quantifier and up to poly many quantifiers all this can be simulated. So I leave this as an exercise. Now what is more interesting is the question how far are these classes from P spaces?

Remember P space is a difficult class it has many hard problems. So you can ask the question how small is IP compared to P space and Shamir proved that IP = P space. So IP is actually not

smaller than P space it is exactly P space which is a very interesting fact and it has a non trivial proof as well. So we have already shown that IP is in P space what remains to be seen is why is P space in IP?

So let us pick a problem in P space which is the hardest and show that in IP. How do you show it in IP? By giving an interactive protocol for Arthur Merlin in and which uses poly many rounds poly and many rounds. So pick a hard problem in P space and show it in and give an interactive protocol that is the idea. So if you have taken a complexity course before basic complexity you would remember that there is a problem which is complete for P space which means that if you solves that problem. Then you solve every problem in P space. So that problem is quantified boolean formula.

(Refer Slide Time: 21:04)



So let us define that. So as the name suggests in this problem you are given an instance you are given a boolean formula. And on the boolean formula there are quantifiers there exist for all and as many as you like. There is no bound on the number of quantifiers alternating quantifiers or the question that you have to answer by doing computations is whether the given input QBF is true.

So given a formula psi where you have variables quantified $\psi: = Q_1 x_1 Q_2 x_2 \ldots\ldots Q_n x_n \phi(\bar{x})$So Q_i are quantifiers for all there exist andϕ is a boolean formula and quantifications. So you have to test whether ψ =true the quantification is over{ 0, 1}. So when we say or when you are given

there exist x_1 the question is whether you can setx_1=0 orx_1= 1 and when you are given for all x_1 then you have to set x_1=0 and x_1=1 and check both the cases.

So the universe here is obviously false true or {0, 1} and with this understanding or this semantics is ψ =true . So what is known is QBF is P space complete. This is a very nice exercise if you have not seen this before basically you can write down a recursive algorithm for QBF which will require only polynomial space. So QBF is in P space but more importantly any P space problem can be converted to QBF. So these for this you have to use turing machine encoding.

Any turing machine that solves a problem in polynomial in n space that you can rewrite as a formula ψ instance of QBF, so this I would not go into those details because that is really basic complexity. If you are interested you can look at the proof in the book but those proof details are not needed here. So we know that QBF is P space complete. Let us use that fact so it suffices to show that QBF ϵ IP that there is an interactive protocol this Arthur Merlin based protocol that solves QBF.

So basically Arthur is the king, Merlin is the advisor. Arthur wants to check whetherψ is true? So Arthur will keep asking Merlin questions Merlin will Merlin is a smart guy marlin will keep advising the king Arthur. But Merlin may be making mistakes the king has to check whether Merlin is making mistakes or is Merlin is telling the truth. So there will be a protocol interactive protocol multiple rounds at the end of which Arthur the king will have certain amount of confidence thatψ= true.

If ψ was false then Arthur will ultimately find the fallacy in Merlin's arguments with high probability. So the protocol will be algebraic surprisingly and it is based on PIT. So this protocol will actually use polynomial identity testing kind of a algorithm. In fact in particular it will use the polynomial identity lemma. So for that let us first arithmetize the QBF problem. So QBF problem as it is given seems very combinatorial or set theoretic.

So let us make it more algebraic. So define an arithmetized version P_ϕ for the boolean formula ϕ. How do you arithmetize a boolean formula? Let us see an example instead of giving the full definition. So suppose you have this formula $\phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_3 \vee x_4)$

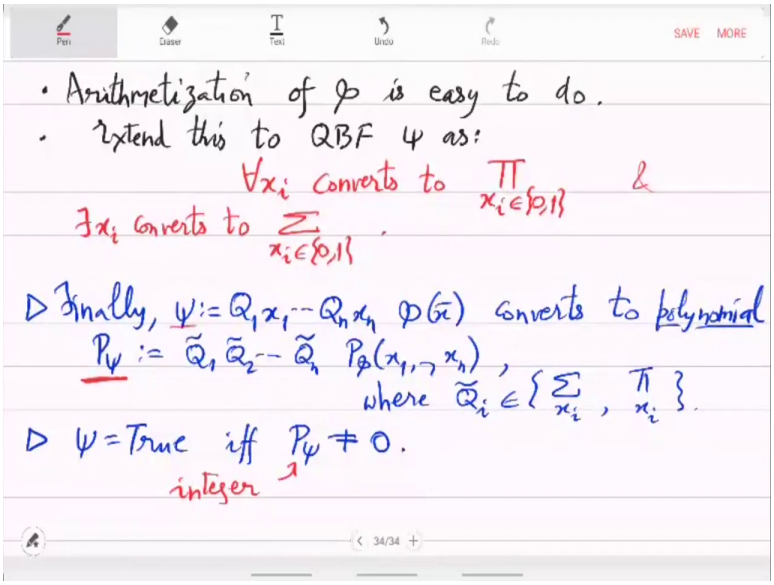
So define P_ϕ as; so basically this boolean formula you want to make it up into a polynomial with variables x_1, x_2, x_3, x_4. So we will do this clause by clause so the first clause you write as I am basically first clause is true only when one of of x_2 is true. So true think of true as the value 1 and false as the value 0 and with that understanding you could write (1-(1-x_1)(1 $- x_2$))

That is the first that is the arithmetized version of the first clause note that if both x_1 x_2 are 0 then this polynomial evaluates to 0. 1 -1 times 1 which is 0 in all other cases it is 1. So it this is a genuine or faithful interpretation reinterpretation or rephrasal of the clause and then similar trick for the second clause. So second clause again $(1 - x_1(1 - x_3)(1 - x_4)$ this will be 0 this polynomial will be 0 only when x_1 =1,x_3=0,x_4=0 in which case you can check that the second clause is false.

So second clause second clause is false if and only if this second polynomial is 0 and P_ϕ is just the product of these 2 polynomials and you can check that $\phi(a_1,\ldots,a_4) =$ true iff $P_\phi(a_1,\ldots,a_4) = 1$ that is the specific property we want in arithmetization. Advantage is that instead of boolean formula ϕ we will work with a polynomial P 5 it is a simple polynomial it has a simple representation and you can see that every coefficient is an integer in P 5.

And this I mean I gave an example but from this example you can see the general definition of p 5 general arithmetization you can easily extend from this basic idea.

(Refer Slide Time: 32:01)



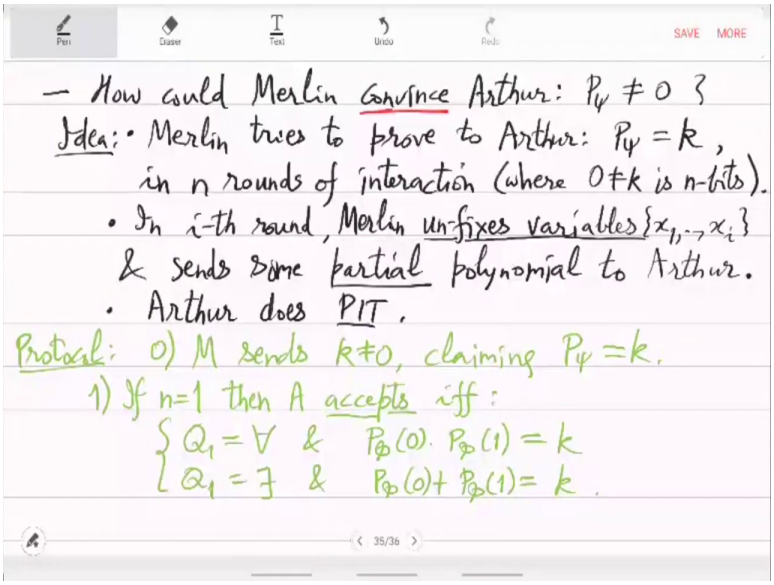
So arithmetization of ϕ is easy to do and then we can extend this to QBF ψ. So what you do is for allx_i for all is both the valuesx_i=0 and x_i = 1 should be satisfied. So we can look at the product $\prod_{x_i\epsilon\{0,1\}}$ & $\forall x_i$ converts to $\sum_{x_i\epsilon\{0,1\}}$. So for all in the boolean word when you go to algebra you have to do multiplication and there exist in the boolean word when you go to algebra you have to add in on the space which is 0, 1.

So what you have after all these steps of arithmetization is this $\psi := Q_1 x_1 \ldots\ldots Q_n x_n \phi(\overline{x})$ this converts to $P_\psi$which will be sum product so$\overline{Q_1}\,\overline{Q}_2 \ldots\ldots \overline{Q_n}$and the arithmetized version of ϕ So where $\overline{Q_i}\epsilon\{\sum_{x_i}, \prod_{x_i}\}$ so $\overline{Q_1}$you can think of $\sum_{x_1\epsilon\{0,1\}}$ and $\overline{Q_2}$ you can think of as $\prod_{x_2\epsilon\{0,1\}}$ so on.

So that ultimately is not even a polynomial it is a constantP_ϕ actually is a constant because every variable you are putting both the values 0, 1. And also what is true is ϕ = true. So$\phi$is either true or false because all the variables are quantified. So $\phi$is true if and only if P_ϕ is not 0, P_ϕ is a constant in fact it is an integer and when this integer is non-0 you can deduce thatϕ is true if this integer is 0 you can reduce thatϕ is false.

So remember this P_ϕ definition. So where are we in the proof. So QBF instance isϕ it is a quantified boolean formula we have converted into a polynomial in fact this integer P_ϕ. Now Arthur actually wants to check whether P_ϕ is a non-zero integer and Arthur will take help from Merlin. So what is this interactive protocol that is the main part of the proof giving this interactive protocol or designing this protocol for Arthur.

(Refer Slide Time: 37:13)



So how could Merlin convince Arthur that this P_ϕ is non-zero. So the idea is that Merlin tries to prove to Arthur that P_ϕ = k in n rounds of interaction n is the number of variables where k is a non zero integer n- bits. This is the plan so Merlin will basically answer or respond to Arthur's challenges and ultimately will be successful in showing to Arthur that the value ofP_ϕ is exactly k which is a non-zero integer.

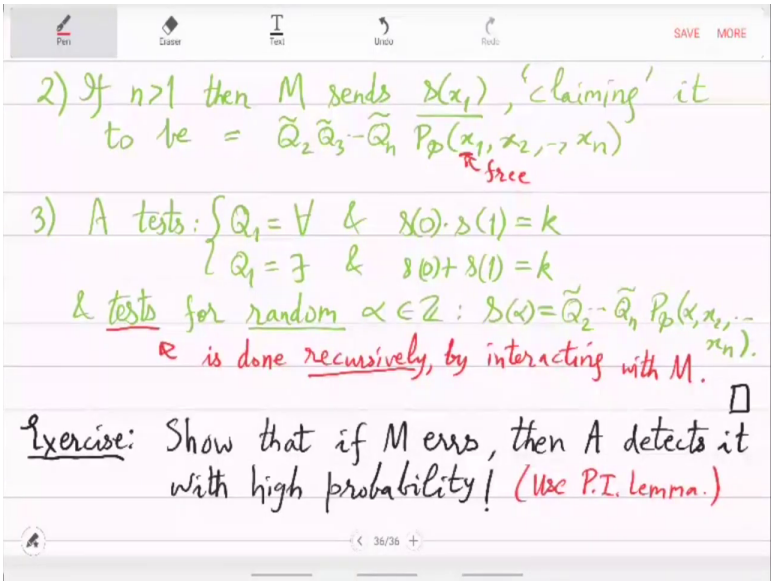
So in the i-th round Merlin fixes the variables and sends some partial polynomial to Arthur and Arthur does PIT. So these are the three key ideas in the protocol. Let us go into a bit more detail. So Merlin sends a k claiming that value of P_ϕ is k that is the first round of interaction or you can think of it as the 0th round of interaction the first round will be either the number of variables is 1. n was 1.

Then Arthur accepts so what will Arthur do? So Arthur will accept if and only if this quantifier on x_1 if this is for all then what should be checked that $P_\phi(0) \cdot P_\phi(1)$=k . On the other hand if the quantifier was there exist then take the sum that should be k. So this is the you can think of this as the base case if n = 1 then it will be easy for Arthur to verify once marlin sends the certificate k because all Arthur has to do is either compute the product or compute the sum of P_ϕ

.

P 5 also is easy to compute for Arthur. So Arthur can do these computations and be happy that is kind of trivial base case. What happens if there are more variables that is the interesting part.

(Refer Slide Time: 42:15)



So if n. 1 then Merlin will send not; Merlin has already sent the value k but now Merlin will send a polynomial in x_1 claiming it to be equal to be= $\overline{Q_2\, Q_3}$..... $\overline{Q_n}\, P_\phi\,(x_1,x_2,\ldots.. x_n)$ So, all that is free here in this list is $x_n$this is free $x_1$is free so this is actually a polynomial in $x_1$and Merlin sends this polynomial as a response to Arthur.

And this is something that Arthur has to now check. So Arthur tests whether what Merlin has just said is true or not. So let me correct the proof idea in ith round Merlin unfixes variables not fixes this is actually unfixing$x_1$will be made free then x_2 and so on. And this partial polynomial which

has been sent that Arthur will now verify so Merlin sent the response as a univariate polynomial in x_1 which we are calling sx1 and Arthur has to test it.

So how does Arthur test? So Arthur in case $Q_1$was for all then Arthur has to check whether $s(0).s(1)$= k if Q_1 is there exist then the sum should be indeed k that is the test which is the same as you did in base case. But this is not enough to check whether the whether s is the univariate polynomial. To check that what Arthur will do is? Tests for a random value αwhether $s(\alpha) = \overline{Q_2\, Q_3}\ldots\ldots \overline{Q_n}\, P_\phi(x_1, x_2, \ldots\ldots x_n)$

So this is like identity testing where left hand side is a polynomial s x hand side is a polynomial in x_1. So these 2 polynomials in x_1 you want to check whether they are equal. So you pick a random α and check the values. So if these 2 values are equal then with high probability Arthur will be convinced that s (x_1) was the correct polynomial sent by Merlin and Merlin was not trying to fool Arthur.

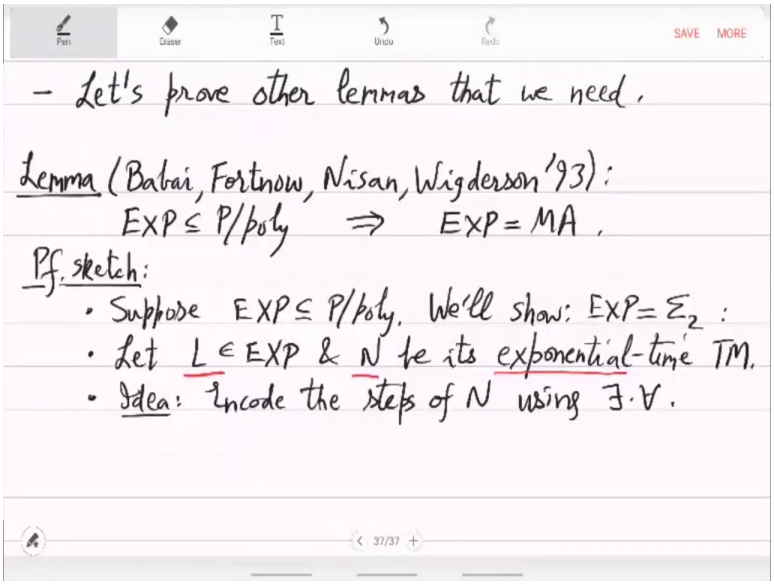
So do these tests but how will this test be done? This is actually the hand polynomial or the hand expression is actually again arithmetized QBF with 1 less variable this is arithmetized for x_2 to x_n variables but this is sigma product sigma product it is a big expression. So Arthur cannot really compute this what Arthur will do is Arthur will actually test this recursively by asking Merlin.

So this test is done recursively by interacting with Merlin. So suppose n was 2 then this test requiresx_2 this computing the expression inx_2. So basically Arthur can again challenge Merlin to send some other polynomial in x_2 which is supposed to be this $\overline{Q_2}\, P_\phi(\alpha, x_2)$ and then this will be the response of Merlin and Arthur has to test it but that will be the base case because both the variables $x_1\, x_2$ will be covered.

So this is a recursive protocol definition. So I have given you the interactive protocol recursive definition. So this proof sketch I hope is clear it is a beautiful protocol. It shows how something as hard as QBF can be solved by interaction. So it was a very brief and very quick proof or proof sketch. The details you can fill or you can read in the textbook. So show that if Merlin makes an error or is trying to fool Arthur by sending wrong polynomial then Arthur will detect it with high probability.

And a hint is use the polynomial identity lemma for this. So you will have to use the polynomial identity lemma for every variable x_1 ,x_2….x_n n times which corresponds to the number of rounds roughly. So we have shown this theorem of Shamir that IP is equal to P space P exactly equals P space and using this we can now prove more interesting lemmas.

(Refer Slide Time: 50:44)



So the next lemma is due to Babai, Fortnow, Nisan and Wigderson and this will be of the type you saw before that assumes something which you do not believe like the class exponential time having polynomial size circuits. We do not believe this because exponential time has very difficult problems. Problems that are even harder than SAT and we certainly do not believe that they can have polynomial size circuits.

If they cannot have polynomial time turing machines then we there is no reason why natural problems should have polynomial size circuits. You assume this and this will give you something

equally hard to believe equality or containment which is x = MA. So we do not believe any of these containments but the point of the lemma is that there is a connection. So if you assume EXP ⊆ P /poly that is expressed small circuits.

Then in fact x has a protocol it has a 2 round Merlin Arthur protocol contra positive of this is that if you can show that EXP is not in MA that x does not have a protocol then you would have shown thatxp is not in P / poly it is a circuit lower bound boolean circuit lower bound that you will get. So we will prove this it will not be a difficult proof we just have to go through some of the basic theorems using the complexity classes that we have defined till now.

So supposeEXP⊆ P /poly which basically gives you small circuits for problems in x. So once you have a small circuit once you know that small circuits exist you can actually try to design an interactive protocol where you challenge Merlin to give that circuit. Since it exists and since it is small Merlin can actually as a response send that tip certificate. Now Arthur has to verify it in the protocol that is the only thing.

So we will we will show that this puts EXP=Σ_2. So let us pick a problem in x let L ϵEXP and N be its exponential time turing machine. So n is the exponential time turing machine solving this problem L. We want to show that L is in Σ_2 which means that we have to write L as they exist for all P we have to express L in this 2 quantifiers there exists for all basically Merlin giving there exists ϕ_1.

So Merlin gives ϕ_1 and then L we cannot see it that way but yeah so 2 quantifiers there exists for all you want to express L s. So how will you do it? So in the next class what we will do is we will go through the steps of turing machine n they are exponentially many steps and we will try to simulate every step in this quantified boolean formula with their exist for all. So we will try to encode the steps of n or using there exist for all encode the steps of n using there exist for all that is the plan.