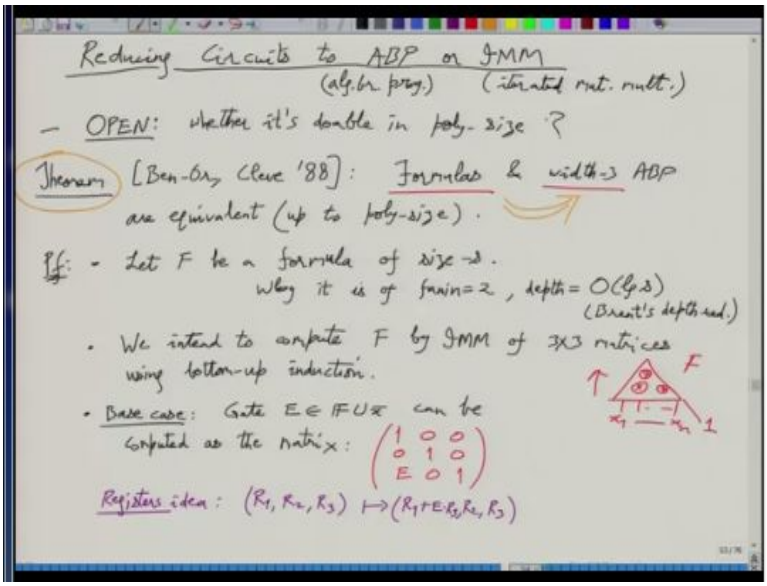
Arithmetic Circuit Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

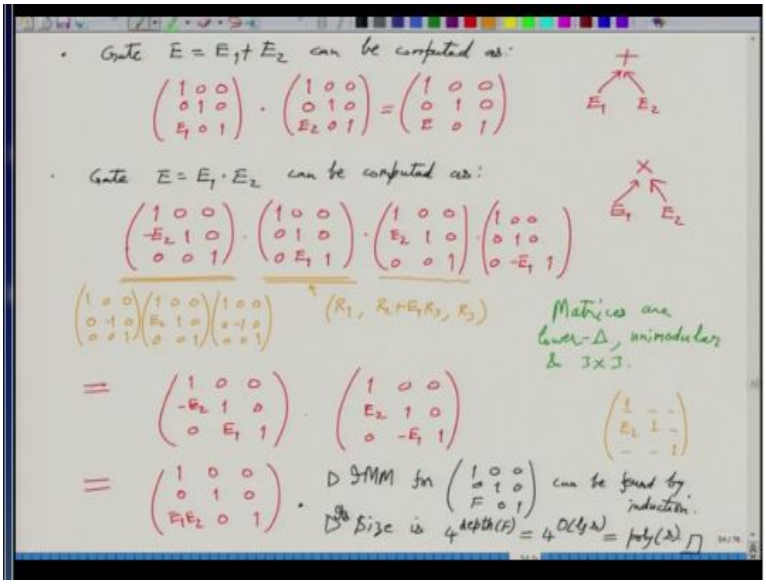
Lecture-12
Width-2ABP Chasm

(Refer Slide Time: 00:17)



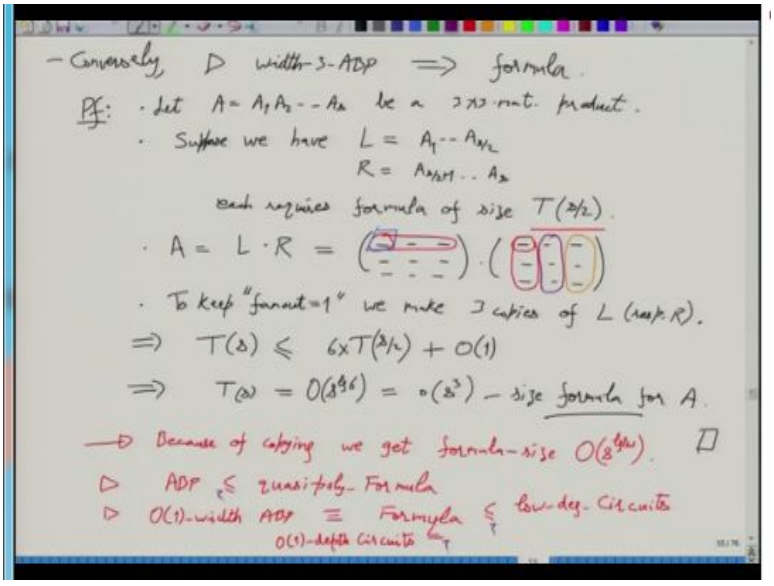
Ok, so last time we proved this theorem by Ben-Or Cleve which we actually showed one type which reduced formulas to be width-2 ABP, right. We have shown this direction that if you are given a formula of sizes. Then you can whatever that polynomial suppose the formula computes polynomial f, then f can as well be written as a matrix products, where is the matrices are only 3×3 where entries are very simple, they are just variables or constants and the number of matrices only poly s.

(Refer Slide Time: 01:12)



So this we did by using some matrix identities 3×3 matrix identities and guiding principle was this interpretation via registers that there are 3 registers and they store polynomials and we work with them. So we apply some simple transformation on one of the registers and then maybe add it to the first register. So those kinds of transformations ultimately give you the full polynomial f in the first register.

(Refer Slide Time: 01:55)



So the other part is much easier, so the converse is, you want to show that width 3 ABP can be converted into a formula, how will you do that? So this is just divide in conquer, so you have let

us say s 3×3 matrices, in fact for this argument even 3 would not be important. So you have some constant dimensional matrices, s of them, you want to multiply them. So you actually multiply the first half, the second half and then merge right.

So it is just like merge sort but it is not as fast as merge sort, so it is basically this divide and conquer argument. So instead of doing the multiplication one by one from left to right you just do it for each of the halves and then the two results you multiply. So let the matrix product be $A_1 A_2 \cdots A_s$ be a 3×3 matrix product. So suppose we have the left product which is $A_1 A_2 \cdots A_{s/2}$ and the right product which is $A_{s/2+1} \cdots A_s$.

So you have these two halves already recursively solved and you have L R, each requiring formula size it is so recursively $T(s/2)$. So we are defining this size function to multiply s 3×3 matrices it the formula sizes, let us say $T(s)$, so for $s/2$ you have $T(s/2)$. But now what you have to be careful about is, how will you multiply L and R right, so $A = L \cdot R$. But how will you do this multiplication because you have these 9 things here and 9 things here for each of them so there are 18 polynomials.

And for each of these polynomials, you have a matrix product, in fact for the L matrix you have a matrix product. So which also gives you these 9 polynomials and analogues for R. So if you do this multiplication in the naive way, so say you multiply the whole row with the first column. But then that row you have to also multiply with the second column and then also the third column.

So in particular this first entry this entry, right this you have to use 3 times. So you have computed that entry but to get an A you actually have to use a fan out of 3 right that is not allowed. So how do you save this, I mean how do you make a fan out 1 exactly you have to make copies of this, so that will be a slight blow up. So to avoid or directly speaking to get fan out 1 to keep fan out 1 we make 3 copies of L respectively R.

So although L we have computed as a single matrix product, we will make copies of it. We will have the first copy of L, second copy of L, third copy of L. And whenever I want, let us say this first entry of L, I will use it first time from the first copy, second time from the second copy and third time from the third copy. So that the fan out everywhere is 1, it is not being reused. So essentially every time it is being the computation is being redone.

We are not using computation, we are actually repeating it, but it is only 3 times, so you have 6 of these, so this claims there following recurrence right. So $T(s)$ is at most, so how many times $T(s/2)$, 6 times and how much of size are you investing right now, assuming that L and R are given in the copies have been made. So how much extra gates are you adding in your formula to compute these row column inner products, constant it is just a function of 3 right.

So what is the solution of this recurrence, so it is worse than merge sort, it is not $s \log s$ it is?. So this is $O(s^{\log 6})$ which is certainly sub cubic, but that is not very important. So $T(s)$ we have gotten to be $poly(s)$, $T(s)$ sized formula for A. So the formula you get is just by divide and conquer and do it with some care, so that the fan out remains 1. So the circuit was direct but formula needs this trick of copying.

And where are you using the fact that it is constant width, so the copying is the only place where you need constant width, that is the important point. So because of copying we get formula size, in general you will get, when you are using $w \times w$ matrices, then this will give you $O(s^{\log_2 w})$. So up to constant width this is fine but when the width is non constant then it gives you a quasi polynomial sized formula right.

So this at the same time we have shown that the ABP model introduces or has a quasi poly formula okay that is one thing we have shown the other thing we have shown is that the model formula and constant width. So formula whatever a formula can compute constant width ABP also can compute. It is exactly these that are exactly equivalent up to poly size blow up.

So this is an interesting fact because this ABP model and formula model where defined in a very different way. In fact they had no relationship whatsoever, in ABP inherently you are reusing computation a lot and in every level actually you the fan out is equal to width and then the length can be arbitrary. So the reuse is happening in an arbitrary way, but still you can get a formula out of this. And what is the relationship between formulas and circuits, so formula seems to be well it is in fact no well.

If you look at low degree polynomials, then it is not clear whether formulas and circuits are different. But intuitively they will seem much weaker than circuits, low degree circuits but we do not know whether they are separate. So just circuits is strictly more than formula because well the degree can go to exponential. But if you restrict to low degree circuits like VP, then we do not have a good understanding of VP compared with VS, how smaller or whether they are equal.

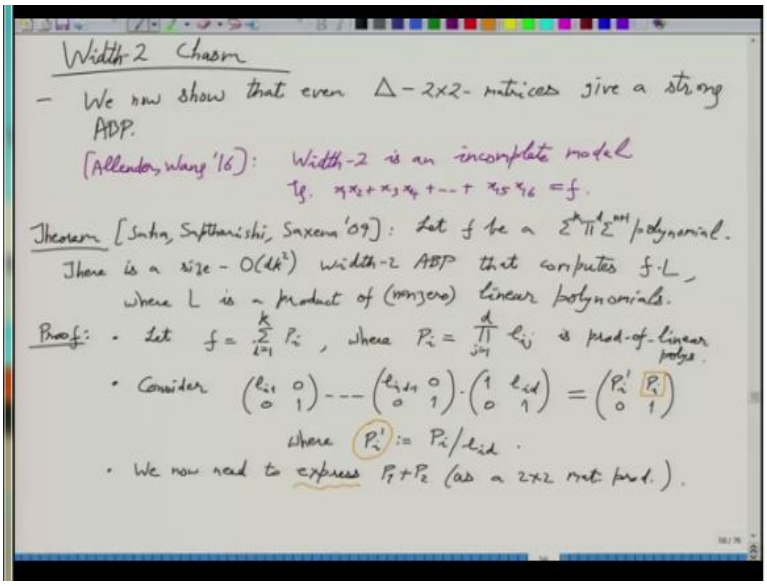
And one type of circuits are actually subsumed by formulas which are constant depth circuits. So constant depth circuits are actually subsumed by formulas. Why is that? So just look at depth 3 for example, right, depth 3 is a powerful module depth 3 circuits. But it is also a formula because nowhere do you need fan out more than 1, I mean these product gates and you can just make copies of everything.

So you do not really need fan out more than 1. So depth 3 circuit is naturally a formula, I mean also when you write it down on paper, you are actually writing it in a single line. So nowhere are you using previous things multiple times previous computation. So it is, so in the same spirit constant depth is also you can implement it as a formula. But whether you can implement formulas in constant depth, that is obviously too much to ask for.

We only know that formulas can be reduced to log depth, we do not know sub log. But it should be easy to rule that out just by degree, just look at the degree no but the fan out then, the fan in can be more than 2. So a product gate can increase degree by taking lots of inputs, so this is also an open question. So these are all open, their strictness is an open question right.

So now we have studied several modules and correspondingly we have several complexity classes and we have some highly non trivial relationships okay. It needs a lot of algebraic and algorithmic ideas. So that was width 3, then we will move to width 2.

(Refer Slide Time: 15:53)



What is width 2 good for? So that is also a chasm okay just like depth 3 circuits width 3 circuits width 2 is also a chasm. So although the width has been minimized still the module remains quite potent okay, you can do powerful things in this module. So we will now show that even triangular 2×2 matrices give a strong ABP. By the way, where did that reduction use width 3, why was 3×3 matrices important, can you do it by just 2×2 matrices.

So sum actually will work, sum is only using this much of the matrix, this part of the matrix. So sum works just like this but the product is more complicated . In fact, there is a result by Allender and Wang that this model is incomplete. So even existentially in this model you cannot compute certain polynomials, no matter how long it gives ABP. So there is a result by Allender and Wang, width 2 is an incomplete model.

So for example, so can you think of a polynomial that is a good candidate, impossible to compute here. So since addition can be done, so somewhere you have to also multiply things to

make it difficult to make it hard. So the degree actually so it will be quadratic, there is a quadratic polynomial that this cannot compute. So like $x_1x_2 + x_3x_4$ is what you can try first but this probably has a width 2 ABP.

So keep going, go up to 16 variables and then you get Allender Wang's proof. So these it is a 16 variate and 8 monomials, they are all disjoint monomials and width 2 model is just cannot compute this it is impossible. I mean, guessing a counter example is not so hard. But you have to prove that this does not have a width 2 ABP with unlimited resources, unlimited length.

I do not think it is that hard actually, you just have to go this model go modulo x_1, modulo x_2 and so on, because in a width 2 ABP your matrices have just linear polynomials right. So actually you can go model those linear polynomials, so that your 2×2 matrix becomes triangular and then show that this polynomial on the left hand side this polynomial $f = x_1x_2 + \cdots + x_{15}x_{16}$ under the modulus you get some contradiction, because this is also absolutely irreducible right.

This polynomial is clearly irreducible and try to use some modulus operations to get that contradiction. So this I will not do, I will see how difficult it is and then it can become an assignment. I do not think it is that difficult. So let me get back to well so width 2 by itself is in I mean generally speaking it is incomplete, but that does not mean that it cannot compute hard polynomials.

So maybe it can still compute some hard polynomials and that is what we will show now. There is a result by Saha, Saptharishi and me which shows that in a way you can convert depth 3 circuits to width 2 ABP, let f be a $\Sigma\Pi\Sigma$ polynomial n variate. So let us also add that here n + 1 is the bottom Σ fan, k is the top fan in and d is the formal degree of your $\Sigma\Pi\Sigma$ circuit right, so then there is a size not much, much bigger than this.

So just size (dk^2) width 2 ABP that computes, well can we say that it computes f? We cannot say that because $\Sigma\Pi\Sigma$ is a complete model. And just the previous counter example you cannot I mean it has a $\Sigma\Pi\Sigma$ polynomial circuit but it cannot have a width 2 ABP. So what we will say is, some multiple of f is computable, $f \cdot L$ is computable, where L is not too difficult. So where L is just a product of non zero linear polynomials and even these linear polynomials will not be magical.

They will just be these linear polynomials that you see in the $\Sigma\Pi\Sigma$ representation. The $\Sigma\Pi\Sigma$ representation is basically a sum of products of linear polynomials. So the set of these linear polynomials you take a suitable product of them, that is L. So $f \cdot L$ is something that has a very small width 2 ABP. Although f may not have a width 2 ABP, in fact these 2×2 matrices will also be upper triangular.

So it will be a very simple representation, the simplest possible that you can think of, you cannot go below width 2. Because below width 2 you are just multiplying linear polynomials even in width 2 we are only using triangular matrices. There are 4 places we are only using 3 which means that this is the minimum kind of the minimum matrix product, where you have non commutativity. If you just use the diagonal entries, then it would be a commutative product.

So this is the first place, where you actually have a non commutative product that is enough to compute kind of depth 3. So let us prove this, so let $f = \sum_{i=1}^{k} P_i$ where P_i is a product of linear polynomials d many, so I mean unsurprisingly this will be an inductive proof. And surprisingly it will also be a divide and conquer proof, so we will somehow use divide and conquer here to be precise, we will use it on these additions.

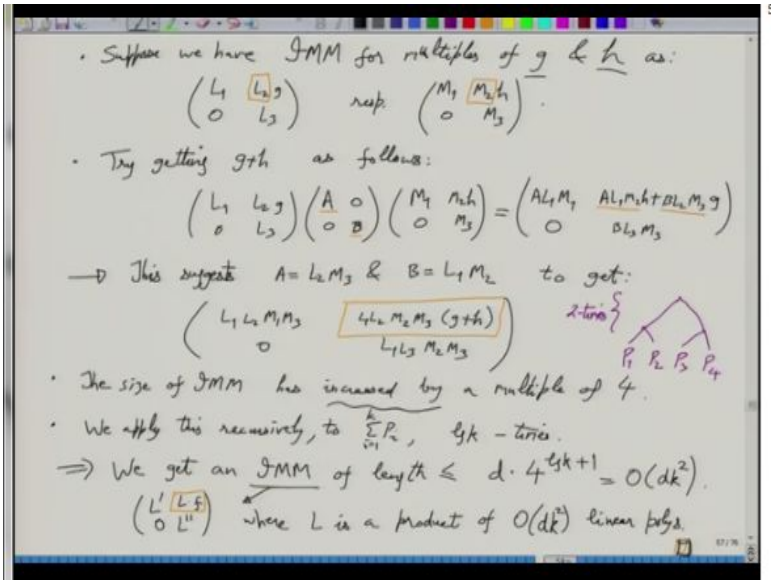
There are k product gates, so we will compute the first $k/2$ as width 2 and the second $k/2$ as width 2 and combine . So base cases, P_i, how do I express P_i as a product of 2×2 matrices. So for P_i I will do this kind of the obvious thing that your $l_{i1} \cdots l_{id-1}$. So in the top left entry

you have computed almost the product P_i . But in the end I will add something different which is okay.

So what is this product, so I get some P_i`, $P_i, 0, 1$ where $P_i` = P_i / l_{id}$. So I have located P_i here, I mean we could have taken l_{id} also the same way as we have taken l_{id-1} and we would have gotten the product P_i, but that representation will not help you in doing addition. Because remember you have to add P_1 with P_2 using matrix products, so for that you have to do something more complicated.

So let us see, how will this representation help, we now need to express $P_1 + P_2$ as a matrix product. So what do you do, so well we will not really express $P_1 + P_2$, but we will get some multiple of $P_1 + P_2$ as a product of 2×2 matrices, assuming that we have a representation for product of P_1 and representation for product of or multiple of P_2.

(Refer Slide Time: 28:56)



So let us go to the next suppose we have iterated matrix multiplication expression for multiples of 2 polynomials g and h. These are some arbitrary polynomials and the representation we have is suppose this okay. So the multiple is by L_2 here and by M_2 here okay, so instead of

computing g and h exactly we have completed them up to these multiples, and L_1, L_3, M_1, M_3 are garbage polynomials we do not care about them.

So now using these 2 matrices can you get by doing matrix product, can you get multiple of g + h, so try getting g + h as follows right. So what are the things that you can do well, you can only multiply matrices, so you have these 2 matrices. Let us write them and maybe you can introduce a third matrix in the middle which is yet to be fixed. So let us say this . So basically you have $L_2 g$ and you have $M_2 h$, a useful combination of this would be $M_2 L_2 g + L_2 M_2 h$.

Because from that $L_2 M_2$ will come out , so that is exactly what you want to do, so for that you have to scale these matrices simultaneously the scaling should happen So for that I have introduced A and B, which is yet to be fixed. So let us look at this product, what is this product, so let me directly write, it is just $L_1 M_1$ gets multiplied with scaling the 0 remains, well because you are only multiplying kind of upper triangular 3 matrices.

So you will get an upper triangular product in $L_3 M_3$ get scaled, so this is the garbage part, the interesting part is what you get here. So that will give you an idea for A and B, so what should you pick A and B as making these equal, if you make them equal then it is a multiple of g + h.

[student] This middle matrix will have elements only in the diagonal, like before even proceeding what was the intuition behind multiplying into the matrix with just diagonal entries.
[Professor] B is scaling $L_2 g$ scaled by B. And simultaneously A is scaling $M_2 h$ and you want things to be as simple as possible. So we actually take a scalar matrix, it is a scalar. So this suggests that take $A = L_2 M_3$ and $B = L_1 M_2$ and that will give you this. So we are only interested in top right part but we also have to see whether for this middle matrix, we have a matrix product right.

So why do we have that for this A B matrix, do you have this width 2 representation. So, answer will be very simple, so the answer for this will be very simple actually, this $L_i M_i$ will all be just

products of linear polynomials. So this middle scalar matrix is actually just naturally it factors. You do not need to design a width 2 ABP it is a scalar matrix and it will completely factor into matrices with which will also be scalar of course, and they will have linear entries.

So essentially it finishes the proof. We just have to calculate the blow ups that the blow ups are under control that will happen by divide in conquer. So now at this point you should observe that to compute g + h. So adding 2 things will be kind of doubling the size because you are multiplying so many things L_1, L_2, M_2, M_3. So in fact, kind of 4 times there is a blow up of up to 4 times and so you cannot afford this blow up more than log many times.

But which is fine because you can add things always, this 2 at a time. So when you have P_1 to P_k you add recursively the first $k/2$ things the next $k/2$ things and then add these 2. So you will again get a recurrence with the halving, so all the things will be fine. So the size is of IMM. Sorry, no the exact expression we will see what is your general question? No, no these recursive steps will only be log k many times.

First $k/2$ next $k/2$ and then merge,

[student] So that is the case $2T(s/2) + O(1)$, $O(1)$ for adding, but if you have $2T(s/2)$ that means 2 to the log many times, that is like s but you want log s right?

[Professor] No, no the T is the size, if your size if your size is coming out to be poly then you are fine.

By a multiple of 4. So max increases is 4 times in one step of this. So we apply this recursively to $\sum_{i=1}^{k} P_i$ log k many times. So in log k additions you can get to k additions it will not log k additions. So in log k steps you can get you can add k things or log k is the depth of your recursion tree right.

We get an IMM of length? So initially we had multiplied d things to compute P_1. So that is length d already consumed and after that you are doing it a blow up every time of 4 maybe up to

$\log k + 1$. So this gives you $O(dk^2)$ that is the length, so you have $O(dk^2)$ many matrices 2×2 . Each matrix has linear entries and it is upper triangular.

This IMM looks like $L`, L``, Lf, 0$, where L is a product of some number many linear polynomials that is an invariant. So how many linear polynomials do you multiply in the end in L, I want to say dk, dk^2 , so maybe dk is too less $O(dk^2)$ many. Because $L_2 M_2$ we have blown up, I mean L_2 we have blown up 4 times kind of, so fine.

So dk^2 is certainly an upper bound, so that many linear polynomials we have multiplied that forms this co-factor L and L times f is what you can now compute as a width 2 ABP. And what are these linear polynomials. So that you can see in the recursive in the induction step, or recursion step, so these are the I mean if you look at P 1 + P 2 computation. So P 1 has these so P what is P i prime right.

So $P_i`$ has these linear factors which appear in the $P_i`$ and P_i are these linear factors which appear in the $\Sigma\Pi\Sigma$ representation. So these are the things which you are multiplying them in a systematic way, so that gives you the multiple right. So that is, so this is the that is the full proof, yes any questions? It is not log k additions it is a recursive algorithm with log k recursion depth, not addition, it is done recursively and if you look at the recursion tree, the depth is log k.

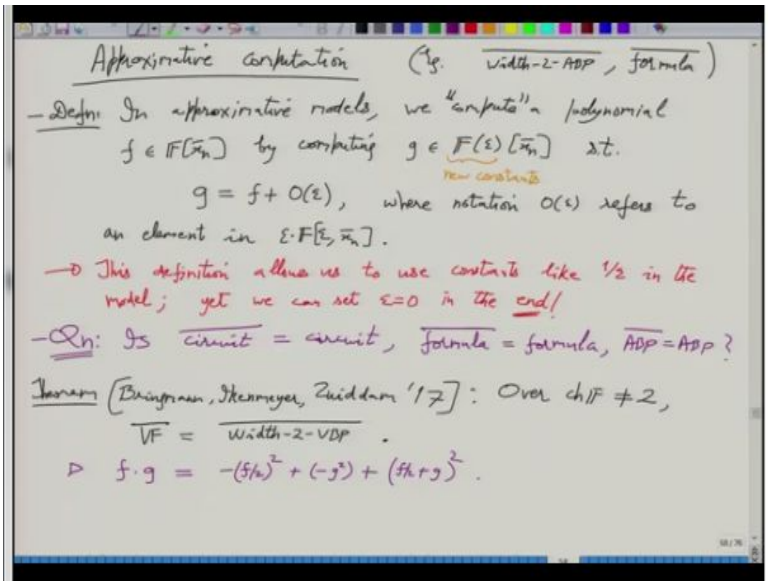
So what if a tree has depth l then it has 2^l leaves and these leaves are P_1 to P_k. So it is if you have P_1, P_2, P_3, P_4 then we will add P_1, P_2 and P_3, P_4 and then we will add these 2 things. And as you can see this can be done beautifully in 2 these are the levels, I am calling it 2 times but it is not obviously 2 additions. The additions are more, so you are adding $P_1 + P_2$, $P_3 + P_4$ and those 2 things actually k - 1, any other question.

Since I do not want to start a completely different concept or I mean which will be actually lower bound. So the next topic will be lower bounds, lower bound results this is structural results we have done, I think almost, whatever there is to do. So the next thing will be the next family of

results are either lower bound proofs. So you take a model and you show that a natural polynomial will require a very large size, those kinds of theorems.

The other interesting results in recent years are these circuit related algorithms. So some of the things which we may cover here are PIT algorithms. So identity testing algorithms that will probably see towards the end of the course. But I do not want to start it now, so I will just continue with a recent result of width 2. But the model will be slightly different then what we have been doing till now in the course okay, so it will be an approximative computation.

(Refer Slide Time: 44:41)



So approximative Computation and as an example case we will prove a result about width 2 ABP in the approximative sense and that is written by drawing a bar. So in fact, I can also do the same for formulas, so formula bar, so this bar will actually denote some kind of a closure of your model. But the closure is, I mean at least the analytic intuition is quite complex. So instead of giving that intuition, I will just give the explicit definition.

So what is meant by approximative, as the word suggests you do not want to compute the polynomial exactly. You want some kind of an approximation of it, so what should be the correct notion of approximation? What do you suggest is the limit of what?

But if you are working in let us say finite field with size 2 elements, what is the meaning of limit? So there is definitely this analytic intuition, which is being referred to but since I want to not depend on the field, I want the notion to be independent of the field. In fact even the ring base ring, so let me just give that definition directly, so in approximative models of computation, what we want is to compute polynomials.

So compute will now be in quotes because we will actually not compute polynomial f, we will compute something else related to f. So we will quote and quote compute f by computing something else which will be in one more variable. So we just add another variable which is ε and if you have taken basic math courses, ε has a special character right. It always is something very close to 0, so that we want to keep that allegiance with ε.

So here also somehow ε denotes something very, very small. But formally speaking it is just a new variable and what is the meaning of this bracket. So this is the function field, so it is basically a ratio of 2 polynomials in ε. So you have now $1/\varepsilon$ or you have $(\varepsilon + 1)/(\varepsilon + 2)$, so you have those expressions okay rational functions are there in ε and then the relationship with the variables x variables is the same as before.

Basically constants, these are the new constants, are now more complicated. And how should g behave, how will g know f. So g should be equal to $f + O(\varepsilon)$, what is $O(\varepsilon)$? So we are actually abusing this notation $O(\varepsilon)$ to an element in it is a polynomial in ε and all the variables. Moreover, it should be a multiple of epsilon, so in algebraic terms $O(\varepsilon)$ is just an element in the ideal generated by ε.

So basically g = f modulo order of the ideal epsilon, but we will continue with this notation $g = f + O(\varepsilon)$ because it kind of means it carries itself the intuition that there are these lower order terms. So g = f, but there are also these lower order terms which are kind of bounded by epsilon. So if you make ε too small I mean analytically or real analytically if you make it very small like 0 then g is f.

So this is the meaning of approximative computation or approximative model. So you can take any model let us say you take a circuit you have written down a circuit that is computing g right which is equal to f up to higher or whatever lower order terms in epsilon. So why does not that mean that you also have a circuit for f, why do not you just set epsilon to 0 and get a circuit for f free of epsilon?

So the only obstruction is or only obstacle is that your circuit will be using $1/\varepsilon$ constants. And then those things will become undefined, the circuit will become undefined. So that is the only catch here, everything else is a triviality. So this definition allows us to use constants like $1/\varepsilon$ in the model, yet we can set $\varepsilon = 0$ in the end. So in the intermediate computation, we are using a $1/\varepsilon$.

So you cannot set ε into 0 because those things will become undefined, but in the end you get $f + O(\varepsilon)$ and there you can set a ε into 0 and recover f. But that does not give you an actual circuit, because intermediate things are all undefined.

[Student] So this is the polynomial for which unless you use $1/\varepsilon$ the circuit size will blow up to

[Professor] Right. So whatever interesting question you will ask, we will call it an open question.

[Student] There are some I think elementary symmetric polynomials or something that are good candidates right?

[Professor] So is circuit closure equal to circuit? Or formula closure equal to formula? With the appropriate meaning of equality. We mean that if you have an approximative circuit then there is also a small circuit exact circuit.

If you have an approximative formula then there is also an exact formula of size up to a poly blow up right. So are these things different or are they equal, so these are all open questions, you had some solution to solve this okay. Elementary symmetric polynomials have a depth 3 circuit very simple. No, no it is there is a small $\Sigma\Pi\Sigma$ circuit by interpolation.

You just look at the auxiliary polynomial interpolate, so you have to , so it is not clear. I mean there is another ABP is also there that everything is open $\overline{ABP}$ whether it is equal to a ABP although these general questions are open, but this approximative, this concept of approximation has been used to design newer and newer matrix multiplication algorithms. So the fast matrix multiplication algorithms in the last 3 decades are all based on this approximative idea.

So algorithmically they have been used, but it does not tell you anything about these fundamental questions. Because I mean matrix multiplication is theoretically a trivial problem, it is solvable in cubic time. So to improve it to quadratic, approximative ideas have been used, but it does not tell you anything about the fundamental nature of approximation. So can circuit bar compute something that is much, much harder than circuits that we have no intuition.

So this is a different dimension to algebraic computation and in this there is a recent theorem, so Bringmann, Ikenmeyer and Zuiddam around 2 years old so it says that formulas. So the complexity class $\overline{VF}$ right, so formula approximative formulas can be written as width 2 ABP. So any polynomial for which you have an approximative formula you also have an approximative width 2 ABP with one mild assumption that you want the characteristic of the field to be not 2 okay.

So characteristic odd are 0 is fine over GF(2) this question is still open okay, but I have no idea why it should I mean why should it not be true? It should be true for all fields irrespective of the characteristic. But the proof which they have given fails for characteristic 2 which is the statement clear, how will you even start to prove such a thing. So you have to convert the formula to well so by induction ultimately it will boil down to converting a product gate into width 2, 2×2 matrix multiplication.

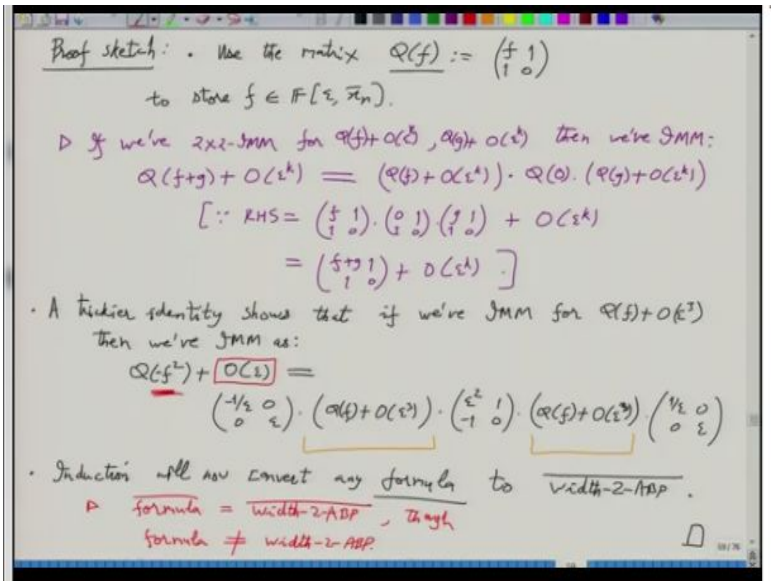
So that seems to be the only thing we have to express using approximation. So I mean what is the benefit of approximation? It is that you can divide by epsilon. So we will actually use matrices,

where entries are dividing by ε. But that is our definition this $f + O(\varepsilon)$ is the definition.

So somebody gives you. So here is the basic question somebody gives you $f + O(\varepsilon)$ can you get from this $f^2 + O(\varepsilon)$. Though this is the only thing which will be solve. No, no it is a width 2 ABP. So somebody gives you a matrix product computing $f + O(\varepsilon)$ from that can you get a matrix product that gives you $f^2 + O(\varepsilon)$. So this is the only thing which will be solved here and it will imply the theorem okay.

We will just show squaring, as you should know multiplication reduces to squaring right, why is that. No, no I am talking about polynomials there is no matrix. So if you 2 polynomials f and g then the product fg can be written as $fg = -(f/2)^2 + (-g^2) + (f/2 + g)^2$. So unless you are unfortunate and characteristic is 2, this identity is holds. So to product actually reduces to sum of squares and sum nobody here doubts that sum can be expressed as width 2 ABP. So assuming that sum is easy, all we have to do is learn how to square right, so that is all where we will use $1/\varepsilon$.

(Refer Slide Time: 1:00:57)



So let us for proof sketch, so use the matrix Q(f). So when we think of the polynomial f the matrix version will be this matrix, it is not really a triangular matrix, but still it has only 3 entries

okay this there is 1 0 present. So this is the matrix that will store f which is a polynomial in all the variables. So $F[\varepsilon, \overline{x_n}]$, so if we have width 2 I mean 2×2 IMM so 2×2 iterated matrix multiplication expression for you have it for $f + O(\varepsilon)$ and $g + O(\varepsilon)$.

In fact it may be even arbitrary precision ε^k, it just means that the remaining part is divisible by ε^k and not just epsilon. So some precision ε^k if you have these 2 expressions then we have IMM to add. So we can add f and g up to the same precision. This is easy to guess you will kind of just multiply the 2 expressions. Sorry I should call it Q(f), now we are into matrices not polynomials.

So I mean Q(f) is a matrix + some matrix which is a multiple of ε^k, every entry there is a multiple of ε^k. So you have a representation IMM for Q(f) matrix with this error and for Q(g) matrix with this error you have these 2 products. So if you multiply these 2 products then you expect to get f + g up to that precision right. So that is all with some calculation you have to show this, so there is a Q(0) sitting in between okay.

So this is just check that in the right hand side you look at Q(f) which is f, 1,1,0, Q(0) is 0,1,1,0, Q(g) plus the error term and what is the part of these 3 matrices it is f + g, 1,1,0. So that is the proof of this identity, so the 2 representations you essentially multiply with something sticking in between this 0,1,1,0 matrix. And that will give you the sum, so that was not very unexpected, what will be more shocking is f^2, how do you get $Q(f^2)$.

So what trickier identity shows that if we have IMM for Q(f) with precision ε^3, I mean the error terms are, they are divisible by ε. If we have an IMM with this much precision for f, then we have an IMM for $Q(f^2)$ in fact $Q(-f^2) + O(\varepsilon)$. So the precision reduces the error term is now nearly a multiple of ε and not ε^3 and it is a product of lots of matrices, where a lot of division is happening, $1/\varepsilon$.

So that expression I will just pull out of thin air right, so you have to bear with me. Sso you want to compute f^1 out of this expression $Q(f^2) + O(\varepsilon^3)$. So we will be multiplying these 2, but we

have to use 3 other matrices which are some entries they will multiply by ε and some entries will scale down by $1/\varepsilon$. So the scaling up and scaling down with the appropriate in the appropriate places will lead to $Q(-f^2)$ with error ε.

So this can again be checked you just have to multiply the first matrix with Q f, then the third matrix with Q f and then the fifth matrix. So just look at this matrix product, this will give you Q of - f square and the remaining things are error terms, you can check there it is they are all multiple of epsilon not epsilon square and not epsilon cube okay. So this precision kind of reduces, so you have to start with a higher precision expression of f.

And by this, I mean that you have to do it because you are going to divide by ε. So if you started with the only precision ε when you divide by ε you get a whole lot of garbage which you cannot deal with. So, you have to start with ε^3 and then it will kind of get divided by ε^2 because you have these two $1/\varepsilon$. So although there is this width 2, 2×2 IMM expression, this does not exist for $\varepsilon = 0$.

So that is and you know that provably does not exist because there is a polynomial which cannot be written as a width 2 ABP which is a simple formula, but it cannot be expressed as width 2 ABP . On the other hand, it has an expression approximatively, this expression. So do induction on this and so induction will now convert any formula to width 2 ABP approximatively, so that is it.

So what you now know is to reiterate $\overline{formula} = \overline{width - 2\ ABP}$, though formula is not , so if this makes you happy, here is an example where approximative computation is more powerful. Although it does not make me happy because it is a very weak model, this width 2 ABP is a very weak model it was actually to begin with it was incomplete, so approximative on top of this just makes it complete.

But it does not really tell you whether at the level of circuits or formulas, there are exponential gaps. So can you do much more with approximative computation than you could do with normal algebraic computation.