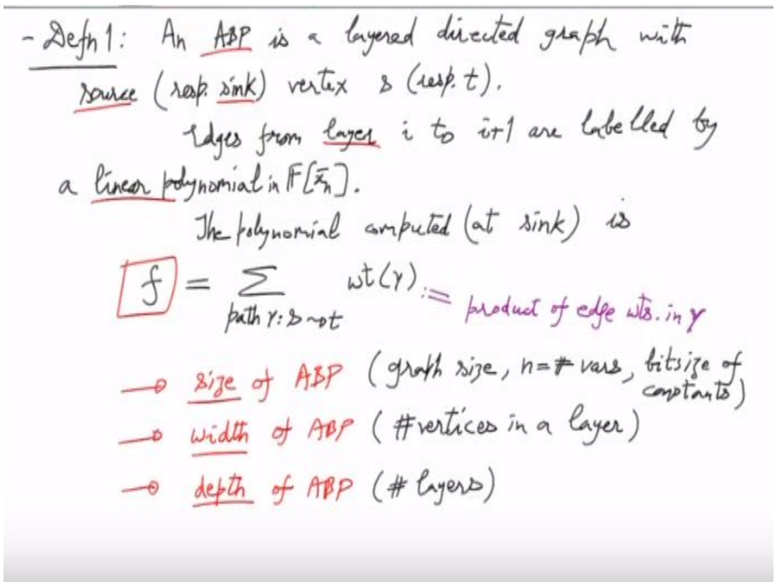
Arithmetic Circuit Complexity

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

Lecture - 04

Fine then we will continue with the definition of arithmetic branching programs. So we have identified the size of ABP.

(Refer Slide Time: 00:17)



- Defn 1: An ABP is a layered directed graph with source (resp. sink) vertex s (resp. t).
Edges from layer i to $i+1$ are labelled by a linear polynomial in $\mathbb{F}[\bar{x}_n]$.
The polynomial computed (at sink) is

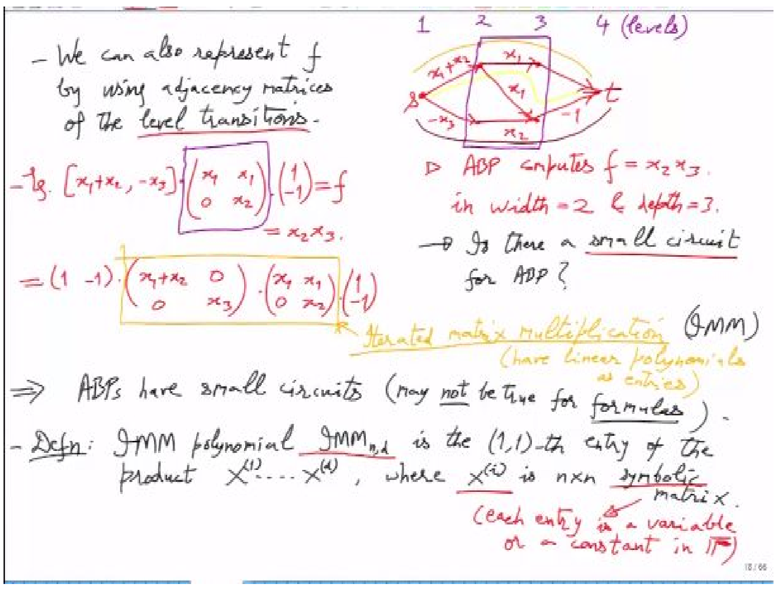
$$\boxed{f} = \sum_{\text{path } \gamma : s \rightsquigarrow t} wt(\gamma) := \text{product of edge wts. in } \gamma$$

→ size of ABP (graph size, $n = \#$ vars, bitsize of constants)
→ width of ABP (# vertices in a layer)
→ depth of ABP (# layers)

So this is just the size of the graph and number of variables you have and it may depends on what you are interested in but if you want you can also include the constant bit size. Since linear polynomials on the edges from layer i to layer i + 1 they have constants. So how big these constants are. If you want to measure, if you want to take that into account then you can also include that in the size.

Otherwise usually it is only the graph size and includes a number of variables. Width is then the number of vertices in a layer and depth is the number of layers. And you can think of it as going from left to right. So let us see an example.

(Refer Slide Time: 01:45)



- We can also represent f by using adjacency matrices of the level transitions.

- Eg. $[x_1+x_2, -x_3] \cdot \boxed{\begin{pmatrix} x_1 & x_1 \\ 0 & x_2 \end{pmatrix}} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix} = f$
$= x_2x_3.$

$= (1\ \ -1) \cdot \boxed{\begin{pmatrix} x_1+x_2 & 0 \\ 0 & x_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 & x_1 \\ 0 & x_2 \end{pmatrix}} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

▷ ABP computes $f = x_2x_3$. in width $= 2$ & depth $= 3$.

→ Is there a small circuit for ABP?

Iterated matrix multiplication (IMM)
(have linear polynomials as entries)

⇒ ABPs have small circuits (may not be true for formulas).

- Defn: IMM polynomial $IMM_{n,d}$ is the $(1,1)$-th entry of the product $X^{(1)} \cdots X^{(d)}$, where $X^{(i)}$ is $n \times n$ symbolic matrix.
(each entry is a variable or a constant in $\mathbb{F}$)

18 / 66

So you have a source here. So this will be width 2, s to t. So the direction is left to right and on this you have $x_1 + x_2$. Here you have $-x_3, x_2, -1, x_1, x_1$ and when I do not see anything then it is assumed to be 1. So what is, so I will not draw the directions always. This is just going like left to right. So at t what is the polynomial that you have computed? Just x_2x_3.

So you have this ABP computes the monomial x_2x_3. So the reason is that there are basically 3 parts from s to t. The first two, whatever they compute, it is negative with each other so in the sum it will not contribute. So the only thing that contributes is $-x_3 \times x_2 \times -1 = x_2x_3$.

So that is the model. This has width 2 and depth is, so the longest path is length 3. So depth is 3. Size you can see, just count the number of edges, vertices and how many variables there are. So the sum will give you the size. Now this pictorial representation is fine, but to analyze it, we also need a more algebraic representation and that ABP has. For circuits, we do not have an alternative representation.

But for ABP we actually have. Can you see that this ABP is also a circuit? It is not so easy because the number of paths can be exponential. So how will you implement that in a circuit? So I should leave it as an open question at this point. Is this a circuit? Is there a small circuit for this? No, it is not obvious because the sum is big.

The sum is over all the paths. So at this point actually, it is not even obvious whether circuits can be implemented by a small circuit. So for all you know ABP may be an incomparable model, but it will not be because we now analyze it algebraically and we will give an alternate representation. So we can also represent this f by using adjacency matrices for the levels, say of the level transitions, okay.

So from level i to i + 1 look at this as a bipartite graph, there is an adjacency matrix. So for a level i there is an adjacency matrix and what is happening, what is the operation on these adjacency matrices that could correspond to the computation ABP computation. Can you guess it? So it is matrix multiplication. So from level 0 to level 1, you have a matrix. Yeah, so let us say 010 or 1112. So that is a row.

Then level next level, let us say 2 to 3. So maybe I will label them. So level 1, 2, 3, 4. So now 2 to 3 the matrix is a square matrix. So you get x_1, x_2 in first row and $0, x_2$ in the second row. And finally, there is a column vector 1, -1. And you can see that this evaluates to the same f. You get $x_2 x_3$. So this is happening because summing over all the paths from s to some vertex that is being simulated by iterated matrix multiplication.

And the first row looks like a polynomial, I mean it has polynomial entries, but we can further simplify it like this. Okay, so now this is the part where you are multiplying matrices that have polynomial entries and on the left you have a constant row vector, on the right you have a constant column vector. So ultimately it will be computing a polynomial over the field.

So this part, the central part is called Integrated Matrix Multiplication. And these matrices have linear polynomials as entries.You will be looking at only this sub graph. So inside this subgraph you see a bipartite graph and the adjacency matrix. That is the x_1, x_1, 0, x_2. So you do this for every level and then multiply them. So that exactly simulates paths.

And then, when you multiply by these row and column vectors, you actually get a polynomial. Intermediate computations are actually matrices having polynomial entries. And the matrices have dimension? Same as width. So these are width by width matrices and intermediate will be width by width matrix with polynomial entries, but in the end you will get a single polynomial.

But this iterated matrix multiplication is a problem in itself. So this is why you should focus more on this. This constant column and row vectors are not very important. So now can you see that this can be subsumed or this can be implemented on a small circuit, because we simply have to multiply matrices and matrix multiplication you can do by addition, multiplication gates.

So at any point of time there will be a $w \times w$ matrix whose circuits you would know. Say you have two matrices, each of $w \times w$ entries whose circuits you know and when you want to compute the matrix product that is given by the obvious formula. So that will again be as a circuit and the growth in the circuit for this multiplication is only additive. And is this a formula?

It isn't a formula right because you are using an entry many times. You are using an entry around w^2 many times. And if you do not use that many times, then you will have to recompute that. So you will have to make copies of that so that in every multiplication it will blow up the formula size. So it is not a formula. It is not a small formula, but it is a small circuit.

So that is a non-trivial thing. It was not obvious from the picture, but this representation immediately gives you that. So ABPs have small circuits but may not have formulas, may not be true for formulas. **"Professor - student conversation starts"** So this width may change level. Yes, the width from i to i + 1 may be different. So then you do not get a square matrix. That is fine. **"Professor - student conversation ends".**

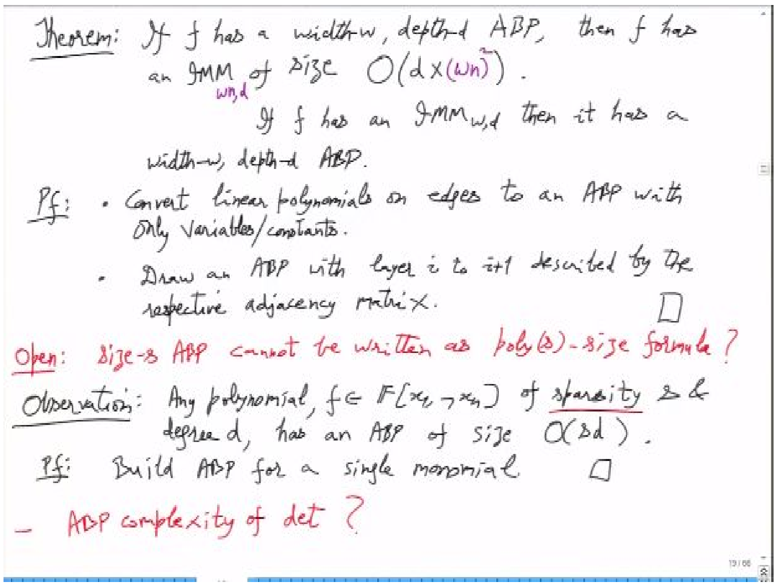
This path interpretation only needs matrix multiplication to be defined, which it will be defined because, when you look at i, you are looking at i, i + 1. And when you look at i + 1you are looking at i + 1 cross i + 2. So the matrix multiplication is always different. It is unconditional on the dimensions. And the biggest matrix that you will need is the width. So that is it. Or the biggest dimension that you will need for any matrix.

So let us define this second problem or second model. This I will shorten to IMM, Iterated Matrix Multiplication. IMM polynomial we will call it, we will denote it by $\text{IMM}_{n,d}$. So this is the let us just look at one entry of the matrix products that is the top left corner entry $(1,1)^{th}$ of the product of d matrices, where these are symbolic matrices, $n \times n$ let us say symbolic matrices.

Okay, so we have defined the polynomial $\text{IMM}_{n,d}$ and it is just a product of the $dn \times n$ totally symbolic matrices which means that every entry is a distinct variable, a fresh variable. So how many variables are there? Well dn^2. So there are dn^2 variables in this polynomial and it is a huge polynomial. The polynomial actually has a lot of monomials exponentially many.

But it has a very small ABP and it has a very small circuit. Okay, so it is an important polynomial. Symbolic would mean that each entry is constant or a variable. Variables may also overlap. Each entry is a variable or a constant in the base field F.

(Refer Slide Time: 17:04)



Theorem: If f has a width-w, depth-d ABP, then f has an $\text{IMM}_{wn,d}$ of size $O(d\times(wn)^2)$.

If f has an $\text{IMM}_{w,d}$ then it has a width-w, depth-d ABP.

Pf:
- Convert linear polynomials on edges to an ABP with only variables/constants.
- Draw an ABP with layer i to $i+1$ described by the respective adjacency matrix. $\square$

Open: size-s ABP cannot be written as poly(s)-size formula?

Observation: Any polynomial, $f\in \mathbb{F}[x_1,\dots,x_n]$ of sparsity s & degree d, has an ABP of size $O(sd)$.

Pf: Build ABP for a single monomial. $\square$

– ABP complexity of det?

So then we can write down this theorem, based on the observations that if f has a width w, depth d ABP then f has an iterated matrix multiplication representation of size, so size of IMM is just the entries you see, the space required to describe these entries. And this will be yeah, so what? So you have d many layers so it is d times something. So what do you want to put here, d times what?

So in a single matrix, there are w^2 entries. But, remember in IMM we have said that each entry should be a variable, it cannot be $x_1 + x_2$. So what do you do with $x_1 + x_2$ for example, which was here. What do you do with this? Or which is actually the source of this is this, you have this $x_1 + x_2$, right.

So how do you implement this or replace this ABP, transform this ABP so that such things are not present, only variables are present or constants are present. You just, t increase the size of it by where you compute $x_1 + x_2$ by a small ABP. You add a layer here. So one path will, one sub path will compute x_1 another sub path will compute x_2. So you will add these edges.

And basically this particular edge where $x_1 + x_2$ is you will actually duplicate it, one with x_1, the other with x_2. And that will grow the width by 1. So you will have width 3 now. But the advantage is that now in the matrices you will have only variables or constants. So that will be an IMM representation. So this width basically grows by multiple of n, right so wn and then you are squaring it. So you get wn^2.

So it is just this simple transformation. So any width w, depth d, ABP also has an IMM representation. So matrix product matrix multiplication of size $d \times (wn)^2$. The matrices are $wn \times wn$. So that is it. And width is wn, depth is d. So this is $\mathrm{IMM}_{wn,\, d}$ representation.

So ABP can be converted to an IMM and IMM can be converted to an ABP. So if f has an IMM width w, depth d then it has a ABP will have width w and depth d ABP. Right, this is even simpler because just from the matrices you can read off the graph, bipartite graph that it represents. So the first thing is just to convert linear polynomials on edges to an ABP with only variables or constants. That will give you an ABP of suitable dimensions.

And second is that IMM to ABP you just drawn an ABP with layer i to i + 1 described by the adjacency matrix, by the respective adjacency matrix. So you can see that ABP and IMM are equivalent models, okay and both of them can be so since ABP and IMM are equivalent and IMM can be implemented as a circuit, ABP is also implementable as a circuit.

But they seem stronger than formulas. So that we can leave as an open question. So what is open is ABP cannot be written as poly s size formula. So the conjecture is that ABPs are stronger than formulas and so ABPs polynomials that are comparable by ABPs in a small way, they may not be representable as small formulas. Formula complexity can be much high, much higher than poly.

Okay, this is an open question. What else can you compute with an ABP? So you have seen that you can compute matrix products, but what are the other examples. Yes, determinant will be a big example. But there is an even simpler example.Consider a sparse polynomial, a polynomial that has few monomials. So any polynomial f in n variables, constants in the field F that has sparsity s, what is sparsity?

Number of monomials on which the polynomial is supported or an upper bound on that. So if the sparsity is less than equal to s, if there are less than equal to s monomials in f and say it has degree d. So any polynomial of this type has a small ABP. Has an ABP of size what do you think in terms of s and d and n? So size is sd.

Yeah so you prove this monomial by monomial. For a single monomial when sparsity is 1, then you just have to draw a line, a path which is a line and which is just multiplying the respective variables. So that has size d and then for s monomials you just have to repeat these many lines and the sum of this is your polynomial okay. So sparsity gives you the size.

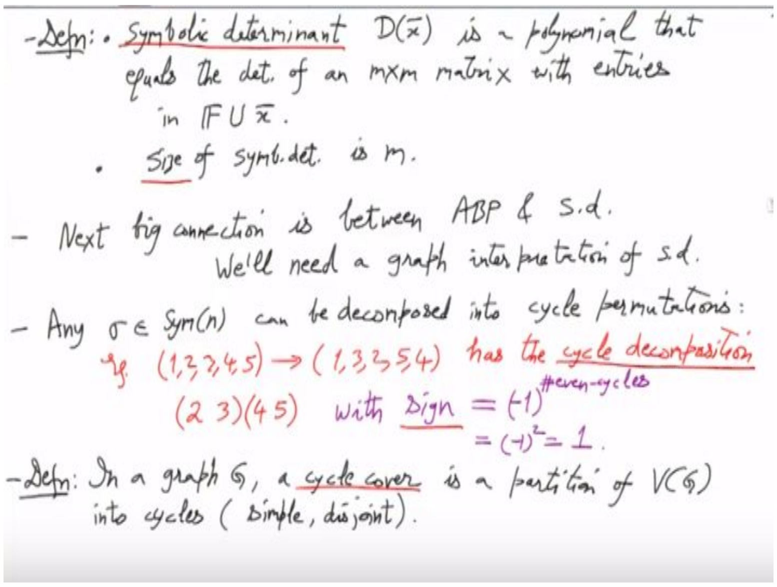
That was simple ABP construction. Proof is just by building the ABP for each monomial. Well for s equal to 1 it is tight. Okay. So it is an optimal bound. **"Professor - student conversation starts"** If you solve many variables you can make it disjoint, let us say d is n by 2 and then this will be tight. Yeah so maybe I should not clean anything about that because those things will need more proofs. **"Professor - student conversation ends".**

But you can, the feeling is that this will be tight and for s is equal to 1 it is. If you have several monomials then it will need a proof. Sure. If you have variable disjoint, yes. Any other questions? So the thing that we are now interested in is what about determinants? ABP complexity of determinant.

So we know that determinant has a small circuit but that does not mean that this has a small ABP because ABP is a more restricted model. So what we will prove at the end of this chapter is that not only are determinants computable as ABPs, ABPs are also computable by determinants. Okay, so this will be, these two will be equivalent models, which is a very powerful connection.

And this again is hard to conjecture. And more importantly it will tell you something about something as fundamental in theory and practice as determinant. Okay, so we will see that actually this connection will improve our results of determinant that we saw in the last class. So we got some parameters. Those parameters will get optimized with this connection.

(Refer Slide Time: 30:06)



- Defn: • Symbolic determinant $D(\bar{x})$ is a polynomial that equals the det. of an $m \times m$ matrix with entries in $\mathbb{F} \cup \bar{x}$.
  • Size of symb. det. is m.
- Next big connection is between ABP & s.d.
  We'll need a graph interpretation of s.d.
- Any $\sigma \in Sym(n)$ can be decomposed into cycle permutations:
  eg. $(1,2,3,4,5) \to (1,3,2,5,4)$ has the cycle decomposition $(2\ 3)(4\ 5)$ with sign $= (-1)^{\#\text{even-cycles}} = (-1)^2 = 1$.
- Defn: In a graph G, a cycle cover is a partition of $V(G)$ into cycles (simple, disjoint).

So let us define symbolic determinants. Symbolic determinant D on some variables is a polynomial that equals determinant of an $m \times m$ matrix with entries either they are field constants or they are variables. So that is a symbolic determinant and the size of symbolic determinant is m. So yeah, this is nothing surprising or important. It is just, we are just formalizing the determinant as a computational model.

So we will say that a polynomial f has determinant complexity or determinant size m, if that polynomial can be written as the determinant of a matrix and by a matrix with entries literals which are variables or field constants. So just like we have circuit complexity, ABP complexity, there is also a determinant complexity of a polynomial. And symbolic determinant is exactly this where the entries are literals.

So the determinant is a polynomial. So the connection is with this model. These two models are equivalent. So the next big connection is between the computational model, algebraic computational model ABP and symbolic determinant. So we will show that they are equivalent up to polynomial size. **"Professor - student conversation starts"** So when we are trying to compute between ABP and symbolic determinant so can't we do that inductively so at each layer i you compute all the, that is a matrix. There is no polynomial.

There is no single polynomial. Yeah but at each layer as each layer if you then made a sink at that layer then what you would have computed would be that. Right, so you will get a matrix of determinant. **"Professor - student conversation ends".** And then you have to show that this matrix of determinant and another matrix of determinants when you multiply them that can be expressed as a single determinant without size blow up.

So it is a pretty non trivial connection. It will need combinatorial and especially graph theoretic interpretation of determinant. So we will need, because ABP I mean you can guess this because ABP was defined as a graph. And so, so we will have to see determinant in graph form, and then connect the two, connect with ABP. So we will need a graph interpretation of symbolic determinant. So have you seen this?

How do you view determinant polynomial or determinant function as, so determinant of a matrix we want to compute. That matrix defines a graph whose adjacency matrix it is. So what is the connection between determinant and that graph? Matching is more to do with the permanent. Thankfully, it has nothing to do with determinant. Otherwise determinant would also been hard.

In the definition of determinant every monomial corresponds to a permutation. So we will actually think of the permutation in terms of cycles in the graph. And we will call it a cycle cover. And then we will say that the determinant is computing sign cycle covers. So basically, it is a sum of the weights of cycle covers over all cycle covers.

So we will, do it slowly. So let us first make the permutation graph friendly. So any permutation on elements 1 to n can be decomposed into cycles, cyclic permutations right. So instead of showing this for a general permutation, let us just see an example because that is enough to understand. So suppose you send 1, 2, 3, 4, 5 to, so this is let us say ordered. So this you are sending, okay let me make it ordered, to 1, 3, 2, 5, 4.

Okay so 1 is mapped to 1, 2 to 3, 3 to 2 and so on. So what is the cyclic decomposition of this permutation? So one is fixed, right? Right so 2, 3, 3, 2 that is a cycle. We will write it compactly this way and the other cycle is 4, 5. So these are cycles of length 2 and this is this has I mean we are not writing 1 here. So 1 also you can see as a cycle which is basically a self-loop. So there are three cycles.

Okay, there is one cycle of length 1 and two cycles of length 2. So that is one thing. The other thing is we can associate a sign with this. So with sign we want to associate with every permutation plus or minus 1. So what do you think it will be? Yeah, so flip is done by this kind of a even cycle. Flip as in swap, for example, it is turned by 2, 3. And that in general if it is an even cycle we will call even cycle we will give it sign -1.

And odd cycles we will give sign +1 and then we will just multiply the signs. So here it, so basically it will be -1 raised to the number of even cycles. So which in this case would come out to be $(-1)^2 = 1$. So we have defined two things: cycle decomposition and the sign of a permutation. So to compute the sign of a permutation, you compute the cycle decomposition. Now convince yourself that it is unique.

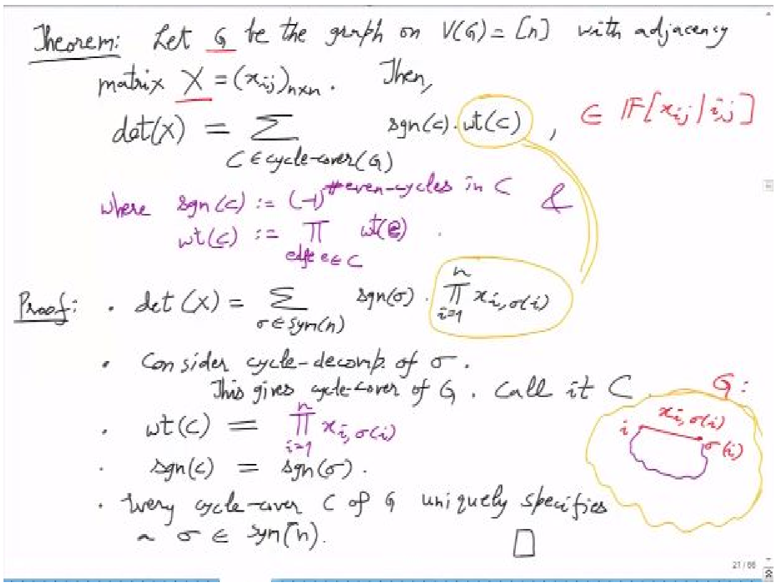
There is only one cycle decomposition and well it is unique mainly because of the disjoint nature. 2, 3 and 4, 5 are disjoint right. So you can easily show that the cycle decomposition is actually unique and then look at the number of even component cycles and parity of parity of that. So you may have written these programs in ESC101 for example, where you compute the number of inversions and so that is related to this.

This even permutation odd permutation, I think has multiple definitions. But they are all equivalent to this, what we just did and so this will now be the also the sign that appears in determinant. So another based on this we will define in a graph G a cycle cover. So this is a partition of vertices VG into cycles where we are we will be assuming that the cycles are disjoint and simple.

So cycle cover of a graph is basically what it says that you try to cover your graph by cycles that are disjoint and within a cycle a vertex does not repeat. Okay, so it is a simple cycle. Every cycle is simple and then they are disjoint and they cover all the vertices. Okay, that is a cycle cover in a graph. And so by the above connection, you can see that cycle cover is basically a permutation as well on the vertices.

It can be thought of like that, and it has an associated sign. So this is now the place where you can see the connection between determinant and the underlying graph. So what a determinant computes of a graph? Any questions about the definition? Okay, so then let me just write down the connection as a theorem statement.

(Refer Slide Time: 42:12)



Theorem: Let G be the graph on $V(G) = [n]$ with adjacency matrix $X = (x_{ij})_{n \times n}$. Then,

$$\det(X) = \sum_{C \in \text{cycle-cover}(G)} \text{sgn}(C) \cdot wt(C) \quad , \in \mathbb{F}[x_{ij} \mid ij]$$

where $\text{sgn}(C) := (-1)^{\#\text{even-cycles in } C}$ &

$wt(C) := \prod_{\text{edge } e \in C} wt(e)$.

Proof:
- $\det(X) = \sum_{\sigma \in \text{Sym}(n)} \text{sgn}(\sigma) \cdot \prod_{i=1}^{n} x_{i,\sigma(i)}$
- Consider cycle-decomp. of σ. This gives cycle-cover of G. Call it C
- $wt(C) = \prod_{i=1}^{n} x_{i,\sigma(i)}$
- $\text{sgn}(C) = \text{sgn}(\sigma)$.
- Every cycle-cover C of G uniquely specifies a $\sigma \in \text{Sym}(n)$. $\square$

So here will say that if G is a graph on vertices, let us say 1 to n, set 1 to n with adjacency matrix given to you $n \times n$ matrix with entries $(x_{ij})_{n \times n}$. So that is the adjacency matrix and this is the underlying graph. So then what is the determinant of x? So determinant of x equals go over all the cycle covers of G and for each cycle cover, take the sign and the weight and sum it up. That will be determinant.

So where sign of C is defined as before and weight of C is just the, what is weight of a cycle cover, what is weight of a cycle first? So the weight of a cycle is a product of the edge labels or weights and then the weight of a cycle cover is all the cycles multiplied by the weights. So whatever edge e you see in the cycle covers the weight of that product. So maybe wt here. So weight of e. okay.

So what is the proof of this? I Thing should be clear by the previous example of permutation decomposition into permutation decomposed into cycles. By that example, you just have to use that example on the definition of determinant. So determinant by definition is going over all the permutations. Now this thing is so I want to write this as weight of some cyclical cover.

Want to associate those two to consider the cycle decomposition of σ. So cycle decomposition of σ gives a cycle cover of the graph. So call it C. When you decompose σ, σ basically will decompose into these cycles as you got before 2, 3, and 4, 5. So you get and 1 is the self-loop. So you get these three cycles.

So similarly, this C, any C would give you a bunch of disjoint simple cycles in the graph G. And now you will look at the weight of that. So weight of the C is what? So the weight on i, $\sigma(i)$ is $x_{i,\sigma(i)}$. This is what is happening inside the graph. So graph is **is** G and inside this what is happening is if i, if σ maps i to $\sigma(i)$ then the weight is this.

So when you look at a cycle, when you look at a cycle inside C, then you can see that weight of this cycle is exactly the product of this $x_{i,\sigma(i)}$ for i appearing in the cycle and then hence also across the other disjoint cycles. So weight of C is exactly equal to the product $x_{i,\sigma(i)}$. And since the product is obviously commutative, so commutativity of the product is used. You will get all the things from i_1 to i_n.

Okay, you can write in this order. But they will be clustered according to cycles in this proof. And then you reorder them because multiplication commutes obviously.

Yeah, so this, maybe, it is important to state here that this determinant is the usual determinant. So it is in this polynomial ring. Okay, so this polynomial ring it is by definition commutative. So you will get this property immediately.

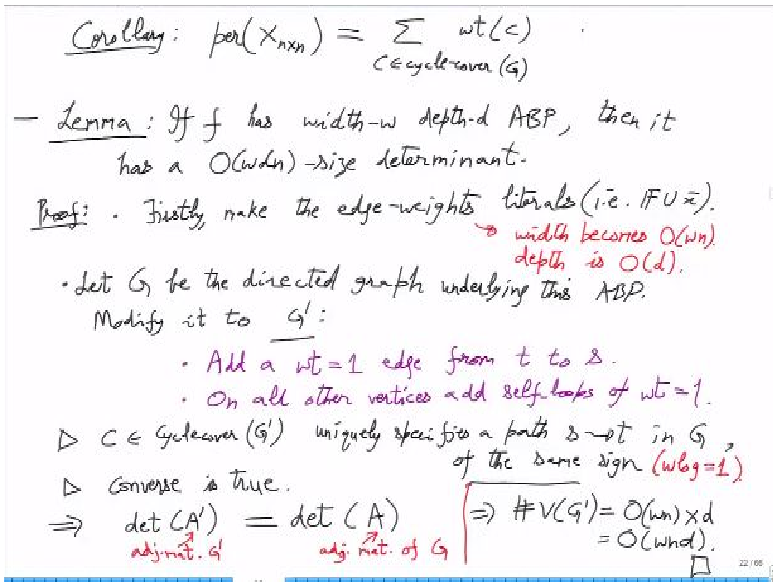
And second thing is sign of C. So if you recall the definition of sign of σ, you will see that is the same. There is really only one definition for the sign of permutations. So with this the proof is complete. Okay, maybe one more thing. So for every σ there is a cycle cover that is clear. What about the converse? For every cycle cover is there a permutation?

So every cycle cover C of G uniquely specifies a permutation on n elements. Right, this is again this follows from the disjointedness of these cycles and their simplicity also. The ones that are self-loops, they will be dropped because they are not being transformed, they are not being moved. The ones that are being moved they are within the site, within a cycle but the cycles across are disjoint. So this is clearly a permutation.

So that finishes the proof. So we get, I mean this sum over cycle covers gives you all the permutations. Basically there is a bijection between cycle covers and permutations. There is a bijection and under this bijection the weight of C and weight of sigma or weight of C and the monomial, they are the same, okay. So that means that the determinant is exactly as stated in theorem, that the two sums are the same.

Okay, yeah but it does not seem that we have done something great because there is a bijection. So either of these things are hard or easy for the same reason. It does not tell us anything particularly new. But at some point it will. At some point it will give you a big result. We will use this.

(Refer Slide Time: 51:43)


<u>Corollary</u>: $per(X_{n\times n}) = \sum_{C \in \text{cycle-cover}(G)} wt(C)$.

— <u>Lemma</u>: If f has width-w depth-d ABP, then it has a $O(wdn)$-size determinant.

<u>Proof</u>: • Firstly make the edge-weights literals (i.e. $\mathbb{F} \cup \bar{x}$).
$\rightarrow$ width becomes $O(wn)$. depth is $O(d)$.

• Let G be the directed graph underlying this ABP.
Modify it to $\underline{G'}$:
• Add a $wt=1$ edge from t to s.
• On all other vertices add self-loops of $wt=1$.

▷ $C \in$ Cyclecover(G') uniquely specifies a path $s \rightsquigarrow t$ in G, of the same sign (wlog $=1$)

▷ Converse is true.

$\Rightarrow$ $\det(\overset{\nearrow}{A'}) = \det(\overset{\nearrow}{A})$
adj. mat. G' adj. mat. of G

$\Rightarrow \#V(G') = O(wn) \times d = O(wnd)$. $\square$

22 / 66

Okay, before that, as a corollary you can also say something about the permanent. So permanent of $n \times n$ matrices also is related to the cycle cover. So drop the sign. So if you just took the sum over weights of all cycle covers in the graph underlying graph, then you will get permanent. This if you specialize to a bipartite graph, then you will get matchings.

"Professor - student conversation starts" It is slightly different right because in a bipartite graph you take the representation of different kind. The representation is different,yes. No it is not. It is the matrix. **"Professor - student conversation ends".** It is a part, yeah right. So yeah you have $2n$ yes you have $2n$ vertices in a bipartite graph assuming here n and m but you do not look at $2n \times 2n$ matrix. You just look at $n \times n$. So you just look at the part that already contains all the information for that, yes exactly. So for specializing x to that you will get the correspondence with matchings which is also naturally related to cycle covers, okay. So now let us do one part of the model equivalents which is, so what do you think will be easy reducing ABP to determinant or determinant to ABP.

So determinant to ABP will actually be the hard part. That came out as a surprise in the 80s. It is a result by Mahajan, Vinay. So that we will do later. If you are interested in practical algorithms it is a new thing because if you recall Gaussian elimination will require division.

So say you want to compute determinant in a ring, entries are in a ring, which does not have division. So it is not a field for example. So in such a ring since you cannot divide you cannot use Gaussian elimination and then you cannot even compute determinant in practice okay and there are many conceivable simple rings where this problem will appear.

And then there is actually no algorithm except the trivial exponential time algorithm to be implemented in practice. So Mahajan, Vinay's result will actually give an ABP for determinant which in particular would mean that no division is required. Determinant is a polynomial, why should you require division?

So ABP representation once you get you will just do addition, multiplication, there is no division and can be efficiently implemented okay. So that will be a really new thing about or even for the practical implementation of determinant. So that is the harder thing. Easier thing now would be with the cycle cover interpretation. Easier thing would be to actually see ABP as a determinant.

Because ABP is defined as a graph with sum over paths. So if you can make those paths into a cycle then it will be like summing over all the cycles and that can be written as a determinant. Also the science would also match. So you want to convert ABP picture, given a ABP picture, you want to convert it into a small modification, it is not hard.

You simply convert it into another graph where you have added some reverse edges so that these paths make a cycle. Say all the cycles have the same length. So the science will be the same. And then the adjacency matrix of this graph will have determinant equal to ABP. Okay, so it will be as simple as that. It is just working with the graph representation. So let us do that.

"Professor - student conversation starts" If we add the reverse edges are we keeping true to the definition of the ABP. No, not certainly. No, we are keeping true with the definition of the determinant. ABP remains what it was. Determinant will be of something else. No. That is not the question. The question is to write a ABP as a determinant.

Can we follow a similar approach to get towards permanent as well, what would be the difficulty we will be facing? Yes. Let us see what the difficulty is. **"Professor - student conversation ends".** No, but so then what would you show. You showed that ABP reduces to permanent. Nobody is surprised with that. It is trivial. The surprising fact would be to convert ABP to permanent. Exactly.

So again, the hard part whether permanent can be written as an ABP. That will shock many people to death. But this one will not, because the permanent is actually complete for VNP. So permanent can compute everything, more on this later. So if f has width w depth d ABP then there is a determinant. Then it has wdn size determinant. Yeah, I am not sure why we are multiplying everything, but we will see in the proof.

So the graph we will get, we will probably have wdn vertices. So you first look at the IMM where the width becomes wn and so there are d layers each with the wn vertices. So you get the product of everything. That is the number of vertices in your graph. So the adjacency matrix will have that as dimension. That is your determinantal complexity. We will prove this is the last lemma today.

So firstly make the ABP edge-weights literal or literals that is in $F \cup \bar{x}$. So that makes the width w n. So that is not very important. So once you are in this situation, width grows, but the depth does not grow. So depth remains the same as before. This is important here. So width changes but depth does not change and so now look at this picture, this graph and let us add the edges.

So let G be the directed underlying this ABP. So modify it to a graph $G`$ as follows. So what we will do is add a unit width add a which is weight equal to 1. Let me write weight equal to 1. So add a weight equal to 1 edge from t to s. Okay, so sink to source we had a reverse edge. So a path becomes now a cycle, s to t and back. But you would be interested in a cycle cover right.

So what will you do with the vertices that have not been used? On all other vertices currently we have only added one reverse edge from t to s and all the other vertices, internal vertices, add self-loops of again weight 1. So this now it should be now obvious that in the $G`$, whenever you look at a cycle cover, or okay let us start with the easy case.

So in G if you look at a path from s to t that corresponds to a cycle cover in $G`$. Because take that path to s to t come back and on the other vertices that have not yet been used, use the self-loops. So that is a cycle cover of $G`$. Now conversely what is a cycle cover of $G`$? Right, so other than the self-loops the only way to get a cycle is to use this newly added reverse edge.

But if you as soon as you use the reverse edge, it means that some path from s to t is being utilized. So every cycle cover in $G`$ corresponds to a path in G and converse. So it is a bijection. So C in cycle cover of $G`$ uniquely specifies a path s to t in G. Moreover, okay let us talk about the signs right.

Also weight is clearly not changed because we have only added unit, new weights are unit weights. So thing to observe is that what is the length of a path from s to t? Does it change over paths? It is the same right, it is equal to depth.

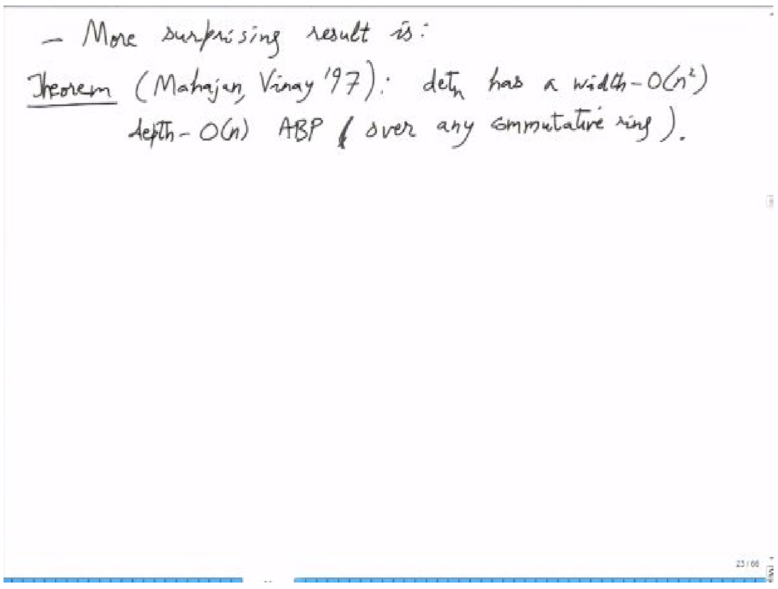
So all the paths are of the same length, which means once you look at the cycle given by this path, the remaining vertices are always the same. So all the cycles have the same, cycle covers have the same sign of the same sign. That is it. The signs are the same, you can assume it to be +1 because you can just the reverse edge you can choose the appropriate thing $+1$ or -1.

So you can assume that it is 1 without loss of generality. Is that clear? And the converse is also true. Every path in G uniquely specifies a cycle cover of the same sign in $G`$. So this sets up the bijection and this then the two things imply that determinant of A prime which is the adjacency matrix of $G`$. This is the same as the determinant of A. So the adjacency matrix of G, right? And finally, can do the size calculation.

So you also get that $A`$, number of vertices in G prime is just $wn \times d$. So that is the determinantal complexity of the original ABP. That finishes the proof. Is this clear? Okay, so that was the easy part, although it is still surprising, because syntactically it does not make any sense. So you were looking at matrix multiplication, iterated matrix multiplication. And now we have written it as a single determinant, right.

So syntactically, it seemed impossible actually, but semantically there is this graph connection interpretation that makes them the same. So iterated matrix multiplication and determinant are essentially the same up to this poly growth in size.

(Refer Slide Time: 1:08:23)



– More surprising result is:

<u>Theorem</u> (Mahajan, Vinay '97): $\det_n$ has a width-$O(n^2)$ depth-$O(n)$ ABP (over any commutative ring).

So more surprising property would be the converse. So this is, a result by Mahajan and Vinay well not 80's, it is actually 97. That determinant on $n \times n$ matrices have an ABP, which is pretty small. So the width is only n square and depth is only n over any commutative ring.

So in particular over fields it is true and the new thing that happened even in terms of even as an algorithmic phenomena the new thing is that when you are working over a commutative ring where you cannot divide, still so you cannot use Gaussian elimination which was the only known way to compute determinant because the previous algorithm which you saw this circuit based thing, right.

Will that work over non fields? **"Professor - student conversation starts"** The multiple of the poly is equal to I mean in retrospection you are scaling up the poly somewhere. So in the end you will have to divide by reverse 1, 2 already. Right **"Professor - student conversation ends".** Yeah, but that is still manageable. No, it is division, but then that is only a question about the characteristic of the ring.

So if the characteristic is large, then you could have done that. But assume that the characteristic is zero for example. It is a commutative ring with characteristics zero. And it has zero divisor. Right, so I guess that is another way. So then this is more I mean this optimizes even that because the width here is only n square and depth is exactly is order n.

So you have around n cube sized ABP representation. So when you convert it into a circuit you will actually get a lower depth. We will see that. So yeah maybe we will do it next time. So we will prove this. Any questions? Okay, so then we will meet next week, Tuesday. I will also release the first assignment this week. Every month we will have one assignment I think, okay.