Recent Progress in Algebraic Circuit Complexity

Prerona Chatterjee

October 20, 2024

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many additions and multiplications does it take to compute f formally?

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many additions and multiplications does it take to compute f formally?

Why?

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many additions and multiplications does it take to compute f formally?

Why?

- Can one succinctly represent polynomials of interest?

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many additions and multiplications does it take to compute f formally?

Why?

- Can one succinctly represent polynomials of interest?
- All *explicit* polynomials are efficiently computable $\overset{\text{G.R.H.}}{\Longrightarrow}$ $\mathsf{P} = \mathsf{NP}$.

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many additions and multiplications does it take to compute f formally?

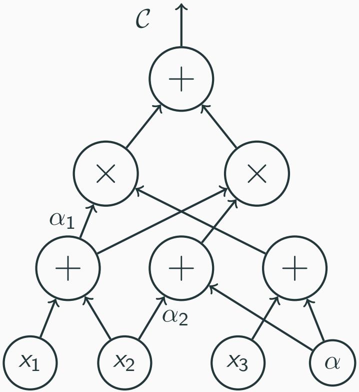
Why?

- Can one succinctly represent polynomials of interest?
- All *explicit* polynomials are efficiently computable $\overset{\text{G.R.H.}}{\Longrightarrow}$ $\mathrm{P} = \mathrm{NP}$.
- **[Shamir 79, Lipton 94]**: If $h(x) = \prod_{i=1}^{d}(x - i)$ can be computed using $\mathrm{poly}(\log d)$ additions and multiplications, then integer factoring is easy for boolean circuits.

Algebraic Models of Computation

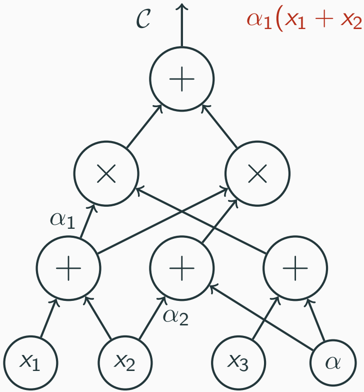
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?



Algebraic Models of Computation

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?

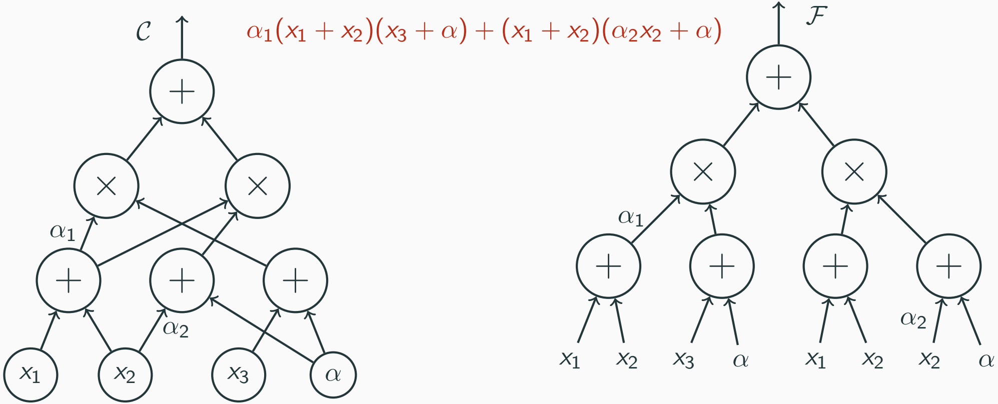


$$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$$

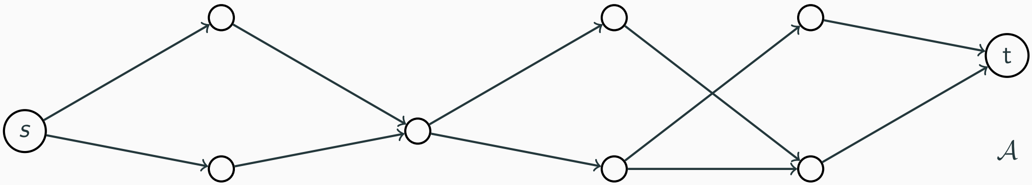
Algebraic Models of Computation

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?

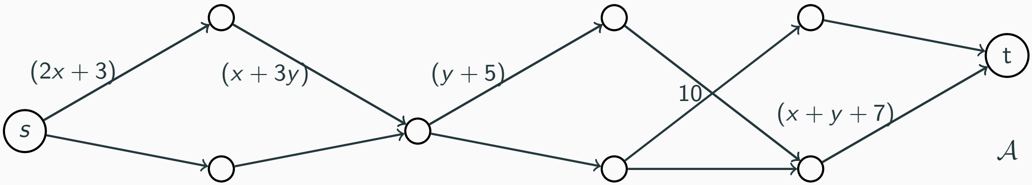
$$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$$



Algebraic Branching Programs

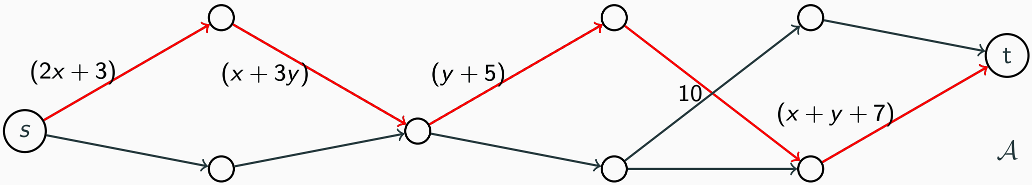


Algebraic Branching Programs



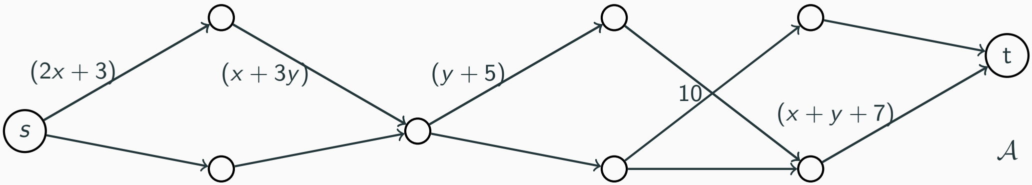
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$

Algebraic Branching Programs



- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \text{wt}(p)$: Product of the edge labels on p

Algebraic Branching Programs



- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on p
- Polynomial computed by the ABP: $f_{\mathcal{A}}(\mathbf{x}) = \sum_p \mathrm{wt}(p)$

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Determinant and Permanent Polynomials

Symbolic Matrix : $$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Computable efficiently by ABPs.

Determinant and Permanent Polynomials

Symbolic Matrix :
$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathsf{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Symbolic Permanent

$$\mathrm{Perm}_n = \sum_{\sigma \in S_n} \prod_{i \in [n]} x_{i\sigma(i)}$$

Computable efficiently by ABPs.

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathsf{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Computable efficiently by ABPs.

Symbolic Permanent

$$\mathrm{Perm}_n = \sum_{\sigma \in S_n} \prod_{i \in [n]} x_{i\sigma(i)}$$

Complete for the class of all explicit polynomials.

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathsf{coeff}_{t^d}\left(\prod_{i \in [n]} (1 + t x_i)\right)$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathsf{coeff}_{t^d}\left(\prod_{i \in [n]} (1 + tx_i)\right)$$

$$\begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^j & \cdots & \alpha_0^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_i^0 & \cdots & \alpha_i^j & \cdots & \alpha_i^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_d^0 & \cdots & \alpha_d^j & \cdots & \alpha_d^d \end{bmatrix} \times \begin{bmatrix} \mathsf{coeff}_{t^0}\left(\prod_{i \in [n]} (1 + tx_i)\right) \\ \vdots \\ \mathsf{coeff}_{t^j}\left(\prod_{i \in [n]} (1 + tx_i)\right) \\ \vdots \\ \mathsf{coeff}_{t^d}\left(\prod_{i \in [n]} (1 + tx_i)\right) \end{bmatrix} = \begin{bmatrix} \prod_{i \in [n]} (1 + \alpha_0 x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_i x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_d x_i) \end{bmatrix}$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathsf{coeff}_{t^d}\left(\prod_{i \in [n]} (1 + t x_i)\right)$$

$$\begin{bmatrix} \mathsf{coeff}_{t^0}\left(\prod_{i \in [n]} (1 + t x_i)\right) \\ \vdots \\ \mathsf{coeff}_{t^j}\left(\prod_{i \in [n]} (1 + t x_i)\right) \\ \vdots \\ \mathsf{coeff}_{t^d}\left(\prod_{i \in [n]} (1 + t x_i)\right) \end{bmatrix} = \begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^j & \cdots & \alpha_0^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_i^0 & \cdots & \alpha_i^j & \cdots & \alpha_i^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_d^0 & \cdots & \alpha_d^j & \cdots & \alpha_d^d \end{bmatrix}^{-1} \times \begin{bmatrix} \prod_{i \in [n]} (1 + \alpha_0 x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_i x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_d x_i) \end{bmatrix}$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S\subseteq[n]} \prod_{i\in S} x_i = \mathrm{coeff}_{t^d}\left(\prod_{i\in[n]}(1+tx_i)\right)$$ is efficiently computable by $\Sigma\Pi\Sigma$ formulas.

$$\begin{bmatrix} \mathrm{coeff}_{t^0}\left(\prod_{i\in[n]}(1+tx_i)\right) \\ \vdots \\ \mathrm{coeff}_{t^j}\left(\prod_{i\in[n]}(1+tx_i)\right) \\ \vdots \\ \mathrm{coeff}_{t^d}\left(\prod_{i\in[n]}(1+tx_i)\right) \end{bmatrix} = \begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^j & \cdots & \alpha_0^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_i^0 & \cdots & \alpha_i^j & \cdots & \alpha_i^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_d^0 & \cdots & \alpha_d^j & \cdots & \alpha_d^d \end{bmatrix}^{-1} \times \begin{bmatrix} \prod_{i\in[n]}(1+\alpha_0 x_i) \\ \vdots \\ \prod_{i\in[n]}(1+\alpha_i x_i) \\ \vdots \\ \prod_{i\in[n]}(1+\alpha_d x_i) \end{bmatrix}$$

Iterated Matrix Multiplication

$$\begin{bmatrix} 1 & \cdots & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_{11}^{(1)} & \cdots & x_{1j}^{(1)} & \cdots & x_{1n}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1}^{(1)} & \cdots & x_{ij}^{(1)} & \cdots & x_{in}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1}^{(1)} & \cdots & x_{nj}^{(1)} & \cdots & x_{nn}^{(1)} \end{bmatrix} \cdots \begin{bmatrix} x_{11}^{(d)} & \cdots & x_{1j}^{(d)} & \cdots & x_{1n}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1}^{(d)} & \cdots & x_{ij}^{(d)} & \cdots & x_{in}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1}^{(d)} & \cdots & x_{nj}^{(d)} & \cdots & x_{nn}^{(d)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Iterated Matrix Multiplication

$$\begin{bmatrix} 1 & \cdots & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_{11}^{(1)} & \cdots & x_{1j}^{(1)} & \cdots & x_{1n}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1}^{(1)} & \cdots & x_{ij}^{(1)} & \cdots & x_{in}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1}^{(1)} & \cdots & x_{nj}^{(1)} & \cdots & x_{nn}^{(1)} \end{bmatrix} \cdots \begin{bmatrix} x_{11}^{(d)} & \cdots & x_{1j}^{(d)} & \cdots & x_{1n}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1}^{(d)} & \cdots & x_{ij}^{(d)} & \cdots & x_{in}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1}^{(d)} & \cdots & x_{nj}^{(d)} & \cdots & x_{nn}^{(d)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Canonical example of a polynomial that is computable by an ABP of width n and length d.

Objects of Study: Polynomials over n variables of degree d.

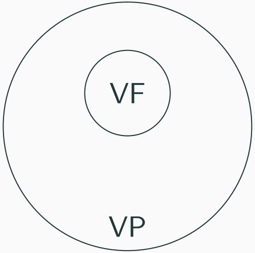
Objects of Study: Polynomials over n variables of degree d.

VP: Polynomials computable by circuits of size poly(n, d).

Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\mathrm{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\mathrm{poly}(n, d)$.



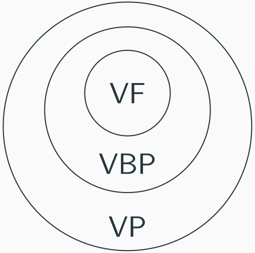
Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size poly(n, d).

VBP: Polynomials computable by ABPs of size poly(n, d).

VP: Polynomials computable by circuits of size poly(n, d).



Lower Bounds in Algebraic Circuit Complexity

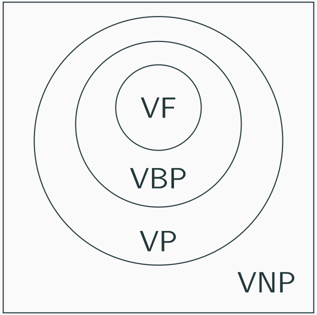
Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

VNP: Explicit Polynomials



Lower Bounds in Algebraic Circuit Complexity

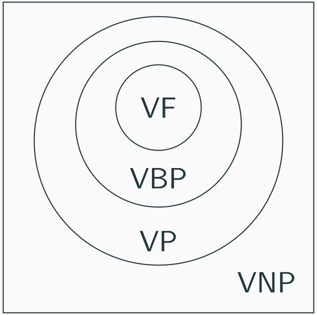
Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

VNP: Explicit Polynomials



Central Question: Find explicit polynomials that cannot be computed by efficient circuits.

Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

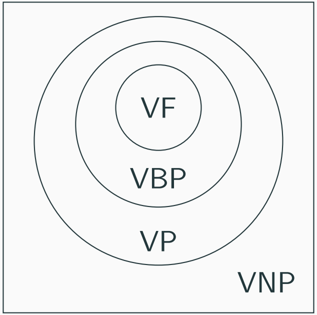
VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

VNP: Explicit Polynomials

$$\boxed{\text{VP} \stackrel{?}{=} \text{VNP}}$$



Central Question: Find explicit polynomials that cannot be computed by efficient circuits.

Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

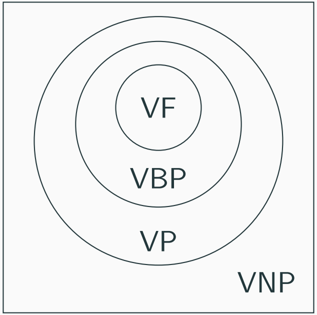
VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

VNP: Explicit Polynomials

$$\boxed{\text{VP} \stackrel{?}{=} \text{VNP}}$$



Central Question: Find explicit polynomials that cannot be computed by efficient circuits.

Other Motivating Questions: Are the other inclusions tight?

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

[Shpilka-Yehudayoff 10] (using Kalorkoti's method): There is an n-variate multilinear polynomial such that any formula computing it requires $\Omega(n^2/\log n)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

[Shpilka-Yehudayoff 10] (using Kalorkoti's method): There is an n-variate multilinear polynomial such that any formula computing it requires $\Omega(n^2/\log n)$ wires.

[C-Kumar-She-Volk 22]: Any formula computing $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$ requires $\Omega(n^2)$ vertices.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial

that is

computable by a general model of size s.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial
that is
computable by a general model of size s.

implies

it is also computed by a structured model of size func(s, n, d) for some function func.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial that is computable by a general model of size s.

implies

it is also computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Step 2: **Study Structured Models**

Prove strong lower bounds against structured models computing f.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial that is computable by a general model of size s.

implies

it is also computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Step 2: **Study Structured Models**

Prove strong lower bounds against structured models computing f.

Ultimate Goal: Prove better than $\text{func}(s, n, d)$ lower bounds.

Homogenity

A polynomial is said to be homogeneous of degree d if every monomial in it has degree d.

Homogenity

A polynomial is said to be homogeneous of degree d if every monomial in it has degree d.

Examples: $\mathrm{Det}_n, \mathrm{Perm}_n, \mathrm{ESYM}_{n,d}, \mathrm{IMM}_{n,d}$.

Homogenity

A polynomial is said to be homogeneous of degree d if every monomial in it has degree d.

Examples: $\mathrm{Det}_n, \mathrm{Perm}_n, \mathrm{ESYM}_{n,d}, \mathrm{IMM}_{n,d}$.

Circuits and ABPs: Can be homogenised with an $O(d)$ blow-up.

Homogenity

A polynomial is said to be homogeneous of degree d if every monomial in it has degree d.

Examples: $\mathrm{Det}_n, \mathrm{Perm}_n, \mathrm{ESYM}_{n,d}, \mathrm{IMM}_{n,d}$.

Circuits and ABPs: Can be homogenised with an $O(d)$ blow-up.

Formulas: Expected to require $s^{\Omega(\log d)}$ blow-up.

Homogenity

A polynomial is said to be homogeneous of degree d if every monomial in it has degree d.

Examples: $\mathrm{Det}_n, \mathrm{Perm}_n, \mathrm{ESYM}_{n,d}, \mathrm{IMM}_{n,d}$.

Circuits and ABPs: Can be homogenised with an $O(d)$ blow-up.

Formulas: Expected to require $s^{\Omega(\log d)}$ blow-up.

$\mathrm{ESYM}_{n,d}$ is computable by an $O(nd)$-sized non-homogeneous formula,
but is expected to require $n^{\Omega(\log d)}$-sized homogeneous formulas to compute.

Homogenity

A polynomial is said to be homogeneous of degree d if every monomial in it has degree d.

Examples: $\mathrm{Det}_n, \mathrm{Perm}_n, \mathrm{ESYM}_{n,d}, \mathrm{IMM}_{n,d}$.

Circuits and ABPs: Can be homogenised with an $O(d)$ blow-up.

Formulas: Expected to require $s^{\Omega(\log d)}$ blow-up.

$\mathrm{ESYM}_{n,d}$ is computable by an $O(nd)$-sized non-homogeneous formula,
but is expected to require $n^{\Omega(\log d)}$-sized homogeneous formulas to compute.

[Fournier-Limaye-Srinivasan-Tavenas 24]
Any homogeneous non-commutative formula computing $\mathrm{ESYM}_{n,n/2}$ requires size $n^{\Omega(\log \log n)}$.

Set-Multilinearity

The variable set is divided into buckets.

$$\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_d \quad \text{where} \quad \mathbf{x}_i = \{x_{i,1}, \ldots x_{i,n_i}\}.$$

Set-Multilinearity

The variable set is divided into buckets.

$$\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_d \quad \text{where} \quad \mathbf{x}_i = \{x_{i,1}, \ldots x_{i,n_i}\}.$$

f is set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

every monomial in f has exactly one variable from $\mathbf{x}_i$ for each $i \in [d]$.

Set-Multilinearity

The variable set is divided into buckets.

$$\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_d \quad \text{where} \quad \mathbf{x}_i = \{x_{i,1}, \ldots x_{i,n_i}\}.$$

f is set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

every monomial in f has exactly one variable from $\mathbf{x}_i$ for each $i \in [d]$.

Circuits and ABPs: Can be set-multilinearised with $2^{O(d)}$ blow-up.

Homogenisation and Set-Multilinearisation of Formulas

[Raz 10]: If a homogeneous polynomial has degree $d = O(\log n)$ and is computable by a formula of size poly(n), then it is also computable by a homogeneous formula of size poly(n).

Homogenisation and Set-Multilinearisation of Formulas

[Raz 10]: If a homogeneous polynomial has degree $d = O(\log n)$ and is computable by a formula of size poly(n), then it is also computable by a homogeneous formula of size poly(n).

[Raz 10]: If a set-multilinear polynomial has degree $d = O(\frac{\log n}{\log \log n})$ and is computable by a formula of size poly(n), then it is also computable by a set-multilinear formula of size poly(n).

Homogenisation and Set-Multilinearisation of Formulas

[Raz 10]: If a homogeneous polynomial has degree $d = O(\log n)$ and is computable by a formula of size poly(n), then it is also computable by a homogeneous formula of size poly(n).

[Raz 10]: If a set-multilinear polynomial has degree $d = O(\frac{\log n}{\log \log n})$ and is computable by a formula of size poly(n), then it is also computable by a set-multilinear formula of size poly(n).

Self-Reducibility of $\mathrm{IMM}_{n,n}$

If $\mathrm{IMM}_{n,D}$ is computable by a set-multilinear formula of size s, then $\mathrm{IMM}_{n,d}$ is computable by a set-multilinear formula of size $s^{O(\frac{\log d}{\log D})}$.

Towards Proving General Formula Lower Bounds

Ways of proving Super-Polynomial Lower Bounds against Formulas:

- Prove super-polynomial lower bounds against set-multilinear formulas for a polynomial of degree $O(\log n / \log \log n)$.

Towards Proving General Formula Lower Bounds

Ways of proving Super-Polynomial Lower Bounds against Formulas:

- Prove super-polynomial lower bounds against set-multilinear formulas for a polynomial of degree $O(\log n/\log\log n)$.
- Prove $n^{\Omega(\log d)}$ lower bound against set-multilinear formulas for $\mathrm{IMM}_{n,d}$ for any d.

Towards Proving General Formula Lower Bounds

Ways of proving Super-Polynomial Lower Bounds against Formulas:

- Prove super-polynomial lower bounds against set-multilinear formulas for a polynomial of degree $O(\log n / \log\log n)$.
- Prove $n^{\Omega(\log d)}$ lower bound against set-multilinear formulas for $\mathrm{IMM}_{n,d}$ for any d.

[Tavenas-Limaye-Srinivasan 22]

$n^{\Omega(\log\log n)}$ lower bound against set-multilinear formulas for $\mathrm{IMM}_{n,n}$.

Towards Proving General Formula Lower Bounds

Ways of proving Super-Polynomial Lower Bounds against Formulas:

- Prove super-polynomial lower bounds against set-multilinear formulas for a polynomial of degree $O(\log n/\log\log n)$.
- Prove $n^{\Omega(\log d)}$ lower bound against set-multilinear formulas for $\mathrm{IMM}_{n,d}$ for any d.

[Tavenas-Limaye-Srinivasan 22]

$n^{\Omega(\log\log n)}$ lower bound against set-multilinear formulas for $\mathrm{IMM}_{n,n}$.

[Kush-Saraf 23]

$n^{\Omega(\log n)}$ lower bound against set-multilinear formulas for $\mathrm{DMPY}_{n,n}$.

So close, yet so far...

So Close...

So close, yet so far...

So Close...

- $\text{DMPY}_{n,n}$ is computable by an ABP of width $O(n^2)$ and length $O(n)$.

So close, yet so far...

So Close...

- $\mathrm{DMPY}_{n,n}$ is computable by an ABP of width $O(n^2)$ and length $O(n)$.
- Any polynomial computable by an ABP of width w and length ℓ is a projection of $\mathrm{IMM}_{w,\ell}$.

So close, yet so far...

So Close...

- $\mathrm{DMPY}_{n,n}$ is computable by an ABP of width $O(n^2)$ and length $O(n)$.
- Any polynomial computable by an ABP of width w and length ℓ is a projection of $\mathrm{IMM}_{w,\ell}$.

Why doesn't this imply a tight lower bound against $\mathrm{IMM}_{n^2,n}$?

So close, yet so far...

So Close...

- $\text{DMPY}_{n,n}$ is computable by an ABP of width $O(n^2)$ and length $O(n)$.
- Any polynomial computable by an ABP of width w and length ℓ is a projection of $\text{IMM}_{w,\ell}$.

Why doesn't this imply a tight lower bound against $\text{IMM}_{n^2,n}$?

$\text{DMPY}_{n,n}$ is not a set-multilinear projection of $\text{IMM}_{n^2,n}$.

So close, yet so far...

So Close...

- $\mathrm{DMPY}_{n,n}$ is computable by an ABP of width $O(n^2)$ and length $O(n)$.
- Any polynomial computable by an ABP of width w and length ℓ is a projection of $\mathrm{IMM}_{w,\ell}$.

Why doesn't this imply a tight lower bound against $\mathrm{IMM}_{n^2,n}$?

$\mathrm{DMPY}_{n,n}$ is not a set-multilinear projection of $\mathrm{IMM}_{n^2,n}$. In fact,

[C-Kush-Saraf-Shpilka 24]

If $\mathrm{DMPY}_{n,n}$ were to be a set-multilinear projection of $\mathrm{IMM}_{w,n}$, then $w = n^{\Omega(n)}$.

Towards Proving General ABP Lower Bounds

[Bhargav-Dwivedi-Saxena 24]

Super polynomial lower bound against total-width of $\sum$osmABP for a polynomial of degree $d = O\left(\frac{\log n}{\log\log n}\right) \implies$ super-polynomial lower bound against ABPs.

Towards Proving General ABP Lower Bounds

[Bhargav-Dwivedi-Saxena 24]

Super polynomial lower bound against total-width of $\sum$osmABP for a polynomial of degree $d = O\left(\frac{\log n}{\log\log n}\right) \implies$ super-polynomial lower bound against ABPs.

[C-Kush-Saraf-Shpilka 24]

For $\omega(\log n) = d \leq n$, $\mathrm{DMPY}_{n,d}$ is a set-multilinear ABP of size $\mathrm{poly}(n)$, but any $\sum$osmABP computing $G_{n,d}$ must have super-polynomial total-width.

Depth Reduction

[Agrawal-Vinay 08, Koiran 12, Tavenas 15]

Size s circuits computing n-variate degree d polynomials can be converted into depth-4 circuits of size $s^{O(\sqrt{d})}$.

Depth Reduction

[Agrawal-Vinay 08, Koiran 12, Tavenas 15]

Size s circuits computing n-variate degree d polynomials can be converted into depth-4 circuits of size $s^{O(\sqrt{d})}$.

[Gupta-Kamath-Kayal-Saptharishi 16]

Size s circuits computing n-variate degree d polynomials can be converted into depth-3 circuits of size $s^{O(\sqrt{d})}$.

Depth Reduction

[Agrawal-Vinay 08, Koiran 12, Tavenas 15]

Size s circuits computing n-variate degree d polynomials can be converted into depth-4 circuits of size $s^{O(\sqrt{d})}$.

[Gupta-Kamath-Kayal-Saptharishi 16]

Size s circuits computing n-variate degree d polynomials can be converted into depth-3 circuits of size $s^{O(\sqrt{d})}$.

In General (using the same techniques)

Size s circuits computing n-variate degree d polynomials can be converted into depth-Δ circuits of size $s^{O(d^{1/\Delta})}$.

[Limaye-Srinivasan-Tavenas 21]

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

Super-Polynomial Lower Bound against Constant Dept Circuits

[Limaye-Srinivasan-Tavenas 21]

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

In particular, it shows that any depth-3 circuit computing $\mathrm{IMM}_{n,\frac{\log n}{100}}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(\sqrt{d}\right)}$.

Super-Polynomial Lower Bound against Constant Dept Circuits

[Limaye-Srinivasan-Tavenas 21]

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

In particular, it shows that any depth-3 circuit computing $\mathrm{IMM}_{n,\frac{\log n}{100}}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega(\sqrt{d})}$.

This shows that the depth-reduction result of **[Gupta-Kamath-Kayal-Saptharishi 16]** is tight.

Super-Polynomial Lower Bound against Constant Dept Circuits

[Limaye-Srinivasan-Tavenas 21]

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

In particular, it shows that any depth-3 circuit computing $\mathrm{IMM}_{n,\frac{\log n}{100}}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega(\sqrt{d})}$.

This shows that the depth-reduction result of **[Gupta-Kamath-Kayal-Saptharishi 16]** is tight.

[Forbes 24]: The theorem is true over all fields.

Proof Overview

Step 1: Convert any arbitrary product-depth Δ circuit into a homogeneous product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $2^{O(\sqrt{d})}\,\mathrm{poly}(s)$.

Proof Overview

Step 1: Convert any arbitrary product-depth Δ circuit into a homogeneous product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $2^{O(\sqrt{d})}\,\mathrm{poly}(s)$.

Step 2: Convert any homogeneous product-depth 2Δ circuit computing a set-multilinear polynomial into a set-multilinear product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $d^{O(d)}\,\mathrm{poly}(s)$.

Proof Overview

Step 1: Convert any arbitrary product-depth Δ circuit into a homogeneous product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $2^{O(\sqrt{d})}\,\mathrm{poly}(s)$.

Step 2: Convert any homogeneous product-depth 2Δ circuit computing a set-multilinear polynomial into a set-multilinear product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $d^{O(d)}\,\mathrm{poly}(s)$.

Step 3: Prove a $n^{\Omega\left(\frac{d^{1/2^{\Delta}-1}}{\Delta}\right)}$ lower bound against set-multilinear constant depth circuits.

Proving Lower Bounds: Finding a Measure

1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.

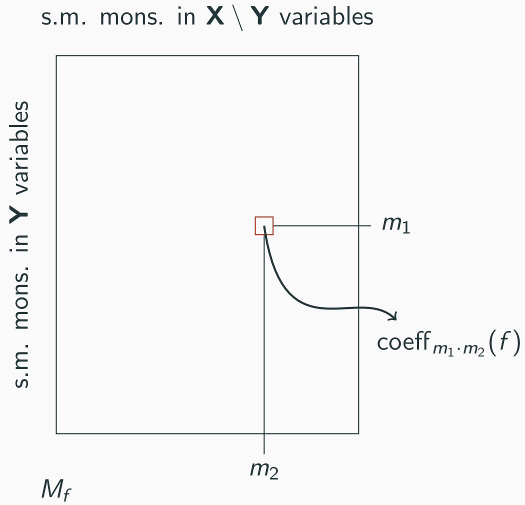
Proving Lower Bounds: Finding a Measure

1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.

Proving Lower Bounds: Finding a Measure

1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.
3. Find an explicit polynomial f such that $\Gamma(f)$ is large.

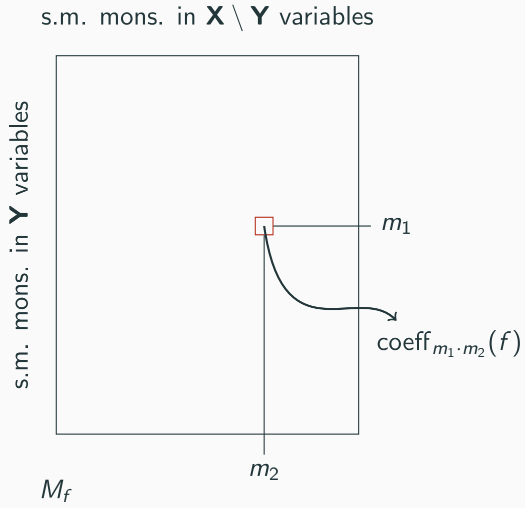
Proving Lower Bounds: Finding a Measure



1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.
3. Find an explicit polynomial f such that $\Gamma(f)$ is large.

Note: Γ is almost always the dimension of some algebraic object and most of the time is simply the rank of a matrix associated with f.

Proving Lower Bounds: Finding a Measure



1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.
3. Find an explicit polynomial f such that $\Gamma(f)$ is large.

Note: Γ is almost always the dimension of some algebraic object and most of the time is simply the rank of a matrix associated with f.

The property "a matrix has low-rank" can be captured by a polynomial equation.

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP?

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

[C-Kumar-Ramya-Saptharishi-Tengse 20]: Yes if the model of interest is $\text{VP}_{\text{bd-coeff}}$.

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

[C-Kumar-Ramya-Saptharishi-Tengse 20]: Yes if the model of interest is $\text{VP}_{\text{bd-coeff}}$.

[K-R-S-T 22]: Perm_n is optimally hard $\implies$ no if the model of interest is VNP.

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in$ VP?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

[C-Kumar-Ramya-Saptharishi-Tengse 20]: Yes if the model of interest is $\text{VP}_{\text{bd-coeff}}$.

[K-R-S-T 22]: Perm_n is optimally hard $\implies$ no if the model of interest is VNP.

The answer is not as clear as the boolean world.

Summary

What we saw...

Summary

What we saw...

- Some of the important models of computation.

Summary

What we saw...

- Some of the important models of computation.
- Some of the polynomials of interest.

Summary

What we saw...

- Some of the important models of computation.
- Some of the polynomials of interest.
- Attacking the main question $\equiv$ Proving structural Results $+$ Attacking structured models.

Summary

What we saw...

- Some of the important models of computation.
- Some of the polynomials of interest.
- Attacking the main question $\equiv$ Proving structural Results $+$ Attacking structured models.
- Some of the well-studied structural restrictions.

Summary

What we saw...

- Some of the important models of computation.
- Some of the polynomials of interest.
- Attacking the main question $\equiv$ Proving structural Results $+$ Attacking structured models.
- Some of the well-studied structural restrictions.
- High Level Proof Overview of most of the lower bounds in the area.

Summary

What we saw...

- Some of the important models of computation.
- Some of the polynomials of interest.
- Attacking the main question $\equiv$ Proving structural Results $+$ Attacking structured models.
- Some of the well-studied structural restrictions.
- High Level Proof Overview of most of the lower bounds in the area.
- Natural Proofs.

Summary

What we saw...

- Some of the important models of computation.
- Some of the polynomials of interest.
- Attacking the main question $\equiv$ Proving structural Results $+$ Attacking structured models.
- Some of the well-studied structural restrictions.
- High Level Proof Overview of most of the lower bounds in the area.
- Natural Proofs.

Next Big Goal of the Area: Proving lower bounds against Homogeneous Formulas.

Thank you!!!