Lower Bounds for some Algebraic Models of Computation

Prerona Chatterjee

April 2, 2024

Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints

Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

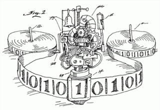
Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



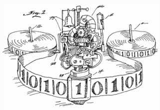
Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

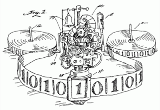
Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

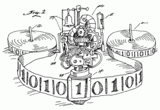
Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

Communication Complexity

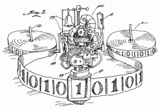
Complexity Theory

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity | **Communication Complexity** | **Quantum Complexity**

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? More tools to work with.

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? More tools to work with.

Usually, Upper Bounds in this setting $\implies$ Upper Bounds in the boolean setting.

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? More tools to work with.

Usually, Upper Bounds in this setting $\implies$ Upper Bounds in the boolean setting.

Lower Bound in this setting is like a step towards Lower Bound in the boolean setting.

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? More tools to work with.

Usually, Upper Bounds in this setting $\Longrightarrow$ Upper Bounds in the boolean setting.

Lower Bound in this setting is like a step towards Lower Bound in the boolean setting.

[Shamir 79, Lipton 94]: If $h(x) = \prod_{i=1}^{d}(x - i)$ can be computed using $\text{poly}(\log d)$ additions and multiplications, then integer factoring is easy for boolean circuits.

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? More tools to work with.

Usually, Upper Bounds in this setting $\Longrightarrow$ Upper Bounds in the boolean setting.

Lower Bound in this setting is like a step towards Lower Bound in the boolean setting.

[Shamir 79, Lipton 94]: If $h(x) = \prod_{i=1}^{d}(x - i)$ can be computed using $\text{poly}(\log d)$ additions and multiplications, then integer factoring is easy for boolean circuits.

Why? Polynomials are central to many algoritms.

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? More tools to work with.

Usually, Upper Bounds in this setting $\implies$ Upper Bounds in the boolean setting.

Lower Bound in this setting is like a step towards Lower Bound in the boolean setting.

[Shamir 79, Lipton 94]: If $h(x) = \prod_{i=1}^{d}(x - i)$ can be computed using poly$(\log d)$ additions and multiplications, then integer factoring is easy for boolean circuits.

Why? Polynomials are central to many algoritms.

Matrix Multiplication Exponent (ω): Smallest number k such that the product of two $n \times n$ matrices can be found using n^k multiplications.

Other Problems that I have worked on

Algebraic Independence Testing: Given polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, check if there exists $0 \not\equiv A \in \mathbb{F}[x_1, \ldots, x_n]$ such that $A(f_1, \ldots, f_n) \equiv 0$.

Partial results in restricted setting with Garg, Saptharishi, Saxena.

Other Problems that I have worked on

Algebraic Independence Testing: Given polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, check if there exists $0 \not\equiv A \in \mathbb{F}[x_1, \ldots, x_n]$ such that $A(f_1, \ldots, f_n) \equiv 0$.

Partial results in restricted setting with Garg, Saptharishi, Saxena.

Polynomial Identity Testing: Given a blackbox computing a polynomial f, along with some added guarantees, check if $f \equiv 0$.

Results in restricted setting with Saptharishi: **[CS 23]**.

Other Problems that I have worked on

Algebraic Independence Testing: Given polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, check if there exists $0 \not\equiv A \in \mathbb{F}[x_1, \ldots, x_n]$ such that $A(f_1, \ldots, f_n) \equiv 0$.

Partial results in restricted setting with Garg, Saptharishi, Saxena.

Polynomial Identity Testing: Given a blackbox computing a polynomial f, along with some added guarantees, check if $f \equiv 0$.

Results in restricted setting with Saptharishi: **[CS 23]**.

Meta Questions on Computing Polynomials: How easy is it to capture efficiently computable polynomials using efficiently computable polynomials?

Results in restricted setting with Kumar, Ramya, Saptharishi, Tengse: **[CKRST 20]**, **[CT 23]**.

Other Problems that I have worked on

Algebraic Independence Testing: Given polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, check if there exists $0 \not\equiv A \in \mathbb{F}[x_1, \ldots, x_n]$ such that $A(f_1, \ldots, f_n) \equiv 0$.

Partial results in restricted setting with Garg, Saptharishi, Saxena.

Polynomial Identity Testing: Given a blackbox computing a polynomial f, along with some added guarantees, check if $f \equiv 0$.

Results in restricted setting with Saptharishi: **[CS 23]**.

Meta Questions on Computing Polynomials: How easy is it to capture efficiently computable polynomials using efficiently computable polynomials?

Results in restricted setting with Kumar, Ramya, Saptharishi, Tengse: **[CKRST 20]**, **[CT 23]**.

Parametric Shortest Paths: Variants of the shortest path problem when the edge weights are labelled with polynomials.

Results in restricted setting with Gajjar, Radhakrishnan, Varma: **[GVCR 21]**.

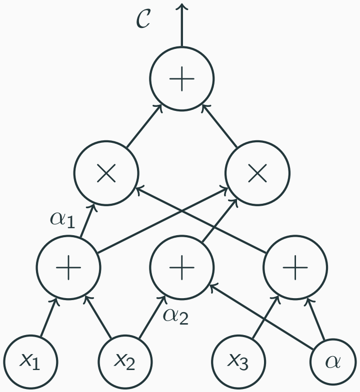
Complexity of Computing Polynomials

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?

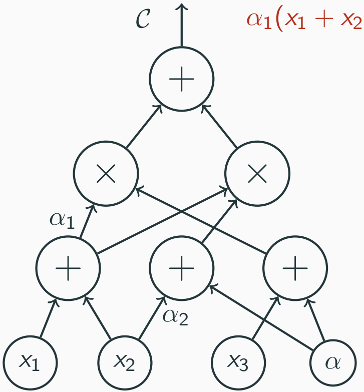
Algebraic Models of Computation

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?



Algebraic Models of Computation

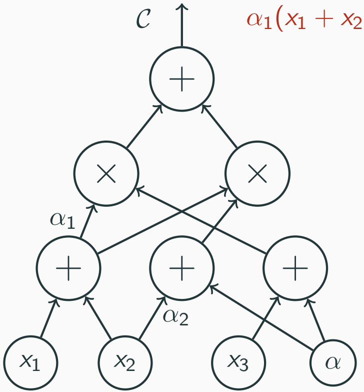
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?



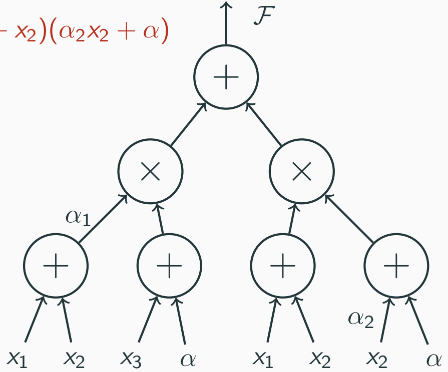
$$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$$

Algebraic Models of Computation

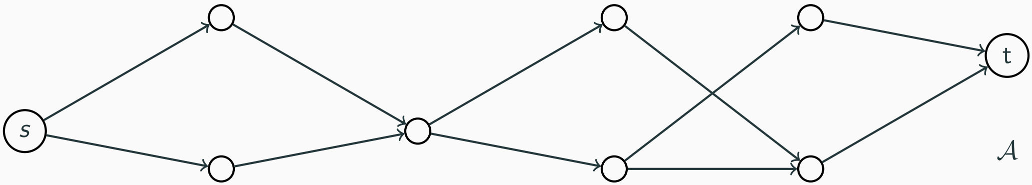
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?



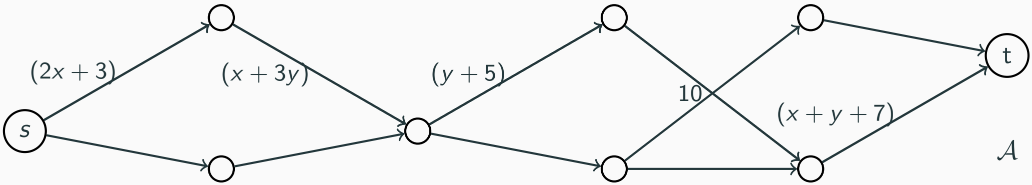
$$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$$



Algebraic Branching Programs

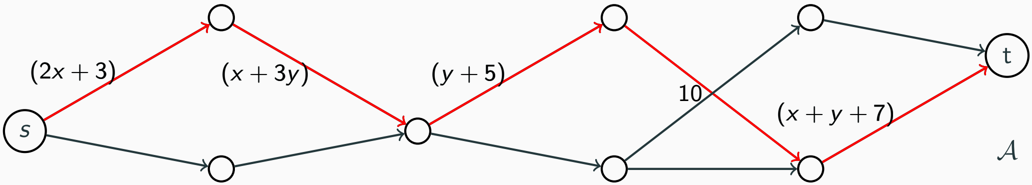


Algebraic Branching Programs



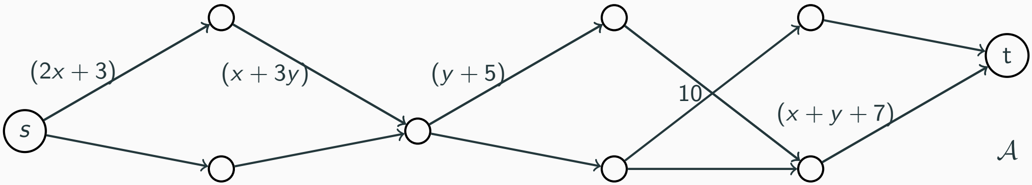
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$

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: $f_{\mathcal{A}}(\mathbf{x}) = \sum_p \mathrm{wt}(p)$

Lower Bounds in Algebraic Circuit Complexity

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

Lower Bounds in Algebraic Circuit Complexity

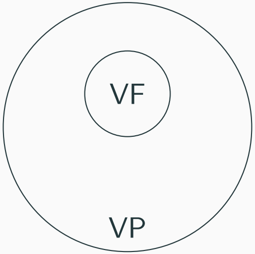
Objects of Study: Polynomials over n variables of degree d.

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

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

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

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

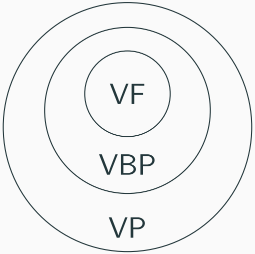


Objects of Study: Polynomials over n variables of degree d.

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

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

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



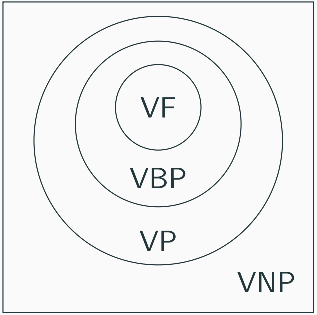
Objects of Study: Polynomials over n variables of degree d.

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

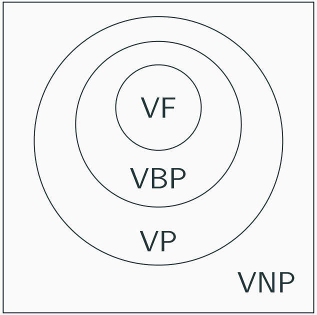
Objects of Study: Polynomials over n variables of degree d.

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.

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

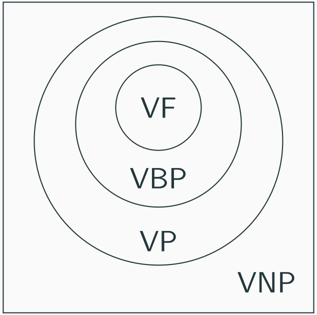
VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

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

$$\boxed{\mathsf{VP} = \mathsf{VNP} \overset{\text{G.R.H.}}{\Longrightarrow} \mathsf{P} = \mathsf{NP}}$$



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

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

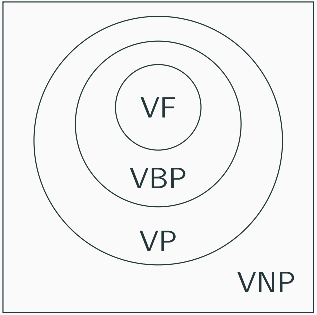
VF: Polynomials computable by formulas of size poly(n, d).

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

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

VNP: Explicit Polynomials

$$\mathrm{VP} = \mathrm{VNP} \overset{\text{G.R.H.}}{\Longrightarrow} \mathrm{P} = \mathrm{NP}$$



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.

$$\mathrm{ESYM}_{n,d}(\mathbf{x}) = \sum_{i_1 < \cdots < i_d \in [n]} x_{i_1} \cdots x_{i_d}.$$

How does one make progress?

Structural Results

Show that if a structured n-variate, degree-d polynomial is computable by a general model of size s, then they can also be computed by a structured model of size $\mathrm{func}(s, n, d)$ for some function func.

How does one make progress?

Structural Results

Show that if a structured n-variate, degree-d polynomial is computable by a general model of size s, then they can also be computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Study Structured Models

Prove strong lower bounds against structured models computing f.

How does one make progress?

Structural Results

Show that if a structured n-variate, degree-d polynomial is computable by a general model of size s, then they can also be computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

[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 $s^{O(\sqrt{d})}$.

Study Structured Models

Prove strong lower bounds against structured models computing f.

How does one make progress?

Structural Results

Show that if a structured n-variate, degree-d polynomial is computable by a general model of size s, then they can also be computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Study Structured Models

Prove strong lower bounds against structured models computing f.

[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 $s^{O(\sqrt{d})}$.

How does one make progress?

Structural Results

Show that if a structured n-variate, degree-d polynomial is computable by a general model of size s, then they can also be computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Study Structured Models

Prove strong lower bounds against structured models computing f.

[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 $s^{O(\sqrt{d})}$.

A lot of work that culminated in

[Limaye-Srinivasan-Tavenas 24]
Any constant depth circuit computing $\mathrm{IMM}_{n,\log n}(\mathbf{x})$ must have super-polynomial size.

How does one make progress?

Structural Results

Show that if a structured n-variate, degree-d polynomial is computable by a general model of size s, then they can also be computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

[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 $s^{O(\sqrt{d})}$.

Study Structured Models

Prove strong lower bounds against structured models computing f.

A lot of work that culminated in

[Limaye-Srinivasan-Tavenas 24]

Any constant depth circuit computing $\mathrm{IMM}_{n,\log n}(\mathbf{x})$ must have super-polynomial size.
The lower bound is $n^{\Omega(\sqrt{d})}$ for depth-3 and depth-4.

Towards Better ABP Lower Bounds

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

Towards Better ABP Lower Bounds

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

[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 Better ABP Lower Bounds

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

[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 $\omega(\log n) = d \leq n$, there is a polynomial $G_{n,d}(\mathbf{x})$ which is set-multilinear w.r.t $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$, where $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that:

- $G_{n,d}$ is computable by a set-multilinear ABP of size $\text{poly}(n)$,
- any $\sum$osmABP computing $G_{n,d}$ must have super-polynomial total-width.

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.

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

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 $i \in [d]$.

An ABP is set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if every path in it

computes a set-multilinear monomial with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$.

Near Tightness of ABP Set-Multilinearisation

For $\sigma \in S_d$, an ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

Near Tightness of ABP Set-Multilinearisation

For $\sigma \in S_d$, an ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

Near Tightness of ABP Set-Multilinearisation

For $\sigma \in S_d$, an ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$ osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

[B-D-S 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.

Near Tightness of ABP Set-Multilinearisation

For $\sigma \in S_d$, an ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

[B-D-S 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-K-S-S 24]: Super polynomial lower bound against total-width of $\sum$osmABP for a polynomial of degree $d = \omega(\log n)$ that is computable by polynomial-sized ABPs.

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Can we do better in this setting?

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

Non-Commutativity

$$f(x,y) = (x+y)\times(x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1\le i_1 < \cdots < i_d \le n} x_{i_1}\cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

Non-Commutativity

$$f(x,y) = (x+y)\times(x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1\leq i_1 < \cdots < i_d \leq n} x_{i_1}\cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \leq \frac{n}{2}$. The lower bound is tight for homogeneous non-commutative circuits.

Non-Commutativity

$$f(x,y) = (x+y)\times(x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$. The lower bound is tight for homogeneous non-commutative circuits.

Further, there is a non-commutative circuit of size $O(n \log^2 n)$ that computes $\mathrm{OSym}_{n,n/2}(\mathbf{x})$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

homogeneous non-commutative ABPs, formulas $\equiv$ ordered set-multilinear ABPs, formulas

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

homogeneous non-commutative ABPs, formulas $\equiv$ ordered set-multilinear ABPs, formulas

$$x_1 x_2 + x_2 x_1 \longrightarrow x_{1,1} x_{2,2} + x_{1,2} x_{2,1}$$

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

homogeneous non-commutative ABPs, formulas $\equiv$ ordered set-multilinear ABPs, formulas

$$x_1 x_2 + x_2 x_1 \longrightarrow x_{1,1} x_{2,2} + x_{1,2} x_{2,1}$$

$$x_2 x_3 + x_1 x_2 \longleftarrow x_{1,2} x_{2,3} + x_{1,1} x_{2,2}$$

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

homogeneous non-commutative ABPs, formulas $\equiv$ ordered set-multilinear ABPs, formulas

$$x_1 x_2 + x_2 x_1 \longrightarrow x_{1,1} x_{2,2} + x_{1,2} x_{2,1}$$

$$x_2 x_3 + x_1 x_2 \longleftarrow x_{1,2} x_{2,3} + x_{1,1} x_{2,2}$$

position indices $\equiv$ bucket indices

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$,

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ such that

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ such that

- There is an abecedarian ABP of size $O(nd)$ that computes f.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ such that

- There is an abecedarian ABP of size $O(nd)$ that computes f.
- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ such that

- There is an abecedarian ABP of size $O(nd)$ that computes f.
- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes f.

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ such that

- There is an abecedarian ABP of size $O(nd)$ that computes f.
- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes f.

If an n-variate polynomial is abecedarian with respect to $\{X_1, \ldots, X_m\}$ for $m = \log n$,

Tight Separation in a Structured Setting

$\{X_1, \ldots, X_m\}$: Partition of the underlying set of variables $\{x_1, \ldots, x_n\}$.

Ordered Set-Multilinear Polynomials: Every monomial has the form $X_1 X_2 \cdots X_m$.

Abecedarian Polynomials: Every monomial has the form $X_1^* X_2^* \cdots X_m^*$.

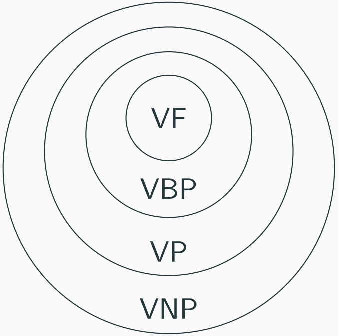
Abecedarian Formulas: Every gate can be labelled by bucket indices of the end points.

[**Cha 21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ such that

- There is an abecedarian ABP of size $O(nd)$ that computes f.
- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes f.

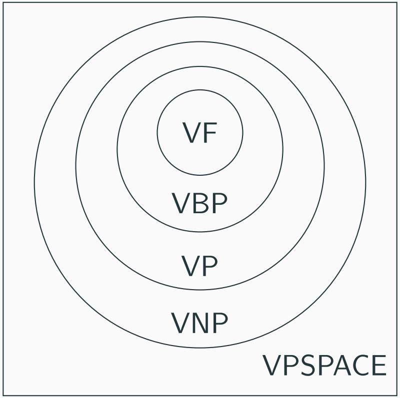
If an n-variate polynomial is abecedarian with respect to $\{X_1, \ldots, X_m\}$ for $m = \log n$, then any formula computing f can be made abecedarian with only $\text{poly}(n)$ blow-up in size.

Classes Beyond VNP



Classes Beyond VNP

$\mathsf{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/ poly and have degree bounded by poly(n).

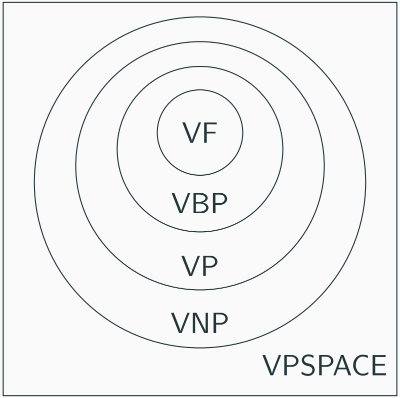


Classes Beyond VNP

$\mathrm{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/poly and have degree bounded by $\mathrm{poly}(n)$.

[Koiran-Perifel 09]

$$\mathrm{VNP} \neq \mathrm{VPSPACE}_b \implies \mathrm{P/poly} \neq \mathrm{PSPACE/poly}.$$



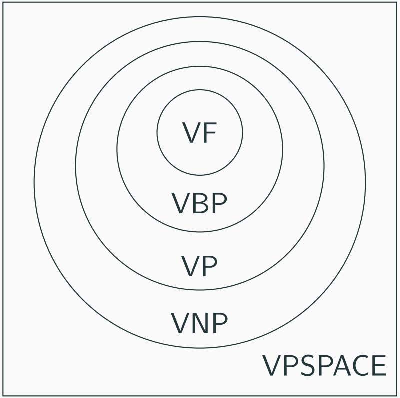
Classes Beyond VNP

$\mathsf{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/poly and have degree bounded by $\mathrm{poly}(n)$.

[Koiran-Perifel 09]

$$\mathsf{VNP} \neq \mathsf{VPSPACE}_b \implies \mathsf{P/poly} \neq \mathsf{PSPACE/poly}.$$

$$\boxed{\mathsf{VNP} \stackrel{?}{=} \mathsf{VPSPACE}_b}$$



Classes Beyond VNP

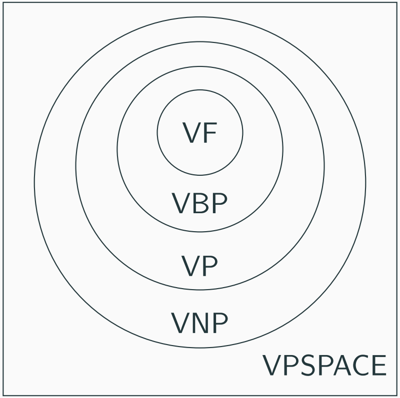
$\mathsf{VPSPACE}_b$: Polynomials whose coefficients can be computed in $\mathsf{PSPACE/poly}$ and have degree bounded by $\mathrm{poly}(n)$.

[Koiran-Perifel 09]

$$\mathsf{VNP} \neq \mathsf{VPSPACE}_b \implies \mathsf{P/poly} \neq \mathsf{PSPACE/poly}.$$

$$\boxed{\mathsf{VNP} \stackrel{?}{=} \mathsf{VPSPACE}_b}$$

[C-Gajjar-Tengse 23]: $\mathsf{VNP} \neq \mathsf{VPSPACE}_b$ in the monotone setting.



Some Proof Ideas

Super-Polynomial Lower Bound against $\sum$osmABPs

An ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

Super-Polynomial Lower Bound against $\sum$osmABPs

An ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

Super-Polynomial Lower Bound against $\sum$osmABPs

An ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

[C-Kush-Saraf-Shpilka 24]: For $\omega(\log n) = d \leq n$, there is a polynomial $G_{n,d}(\mathbf{x})$ which is set-multilinear w.r.t $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$, where $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that:

Super-Polynomial Lower Bound against $\sum$osmABPs

An ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

[C-Kush-Saraf-Shpilka 24]: For $\omega(\log n) = d \leq n$, there is a polynomial $G_{n,d}(\mathbf{x})$ which is set-multilinear w.r.t $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$, where $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that:

- $G_{n,d}$ is computable by a set-multilinear ABP of size $\mathrm{poly}(n, d)$,

Super-Polynomial Lower Bound against $\sum$ osmABPs

An ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$ osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

[C-Kush-Saraf-Shpilka 24]: For $\omega(\log n) = d \leq n$, there is a polynomial $G_{n,d}(\mathbf{x})$ which is set-multilinear w.r.t $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$, where $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that:

- $G_{n,d}$ is computable by a set-multilinear ABP of size $\text{poly}(n, d)$,
- any $\sum$ osmABP of max-width $\text{poly}(n)$ computing $G_{n,d}$ requires total-width $2^{\Omega(d)}$,

Super-Polynomial Lower Bound against $\sum$ osmABPs

An ABP is σ-ordered set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

- there are d layers in the ABP
- every edge in layer i is labelled by a homogeneous linear form in $\mathbf{x}_{\sigma(i)}$

$\sum$ osmABP: Sum of ordered set-multilinear ABPs, each with a possibly different ordering.

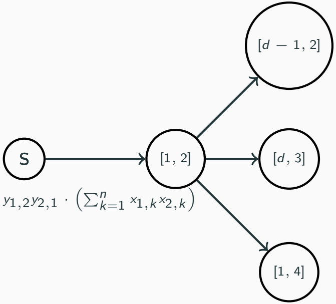
[C-Kush-Saraf-Shpilka 24]: For $\omega(\log n) = d \leq n$, there is a polynomial $G_{n,d}(\mathbf{x})$ which is set-multilinear w.r.t $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$, where $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that:

- $G_{n,d}$ is computable by a set-multilinear ABP of size $\text{poly}(n, d)$,
- any $\sum$ osmABP of max-width $\text{poly}(n)$ computing $G_{n,d}$ requires total-width $2^{\Omega(d)}$,
- any ordered set-multilinear branching program computing $G_{n,d}$ requires width $n^{\Omega(d)}$.

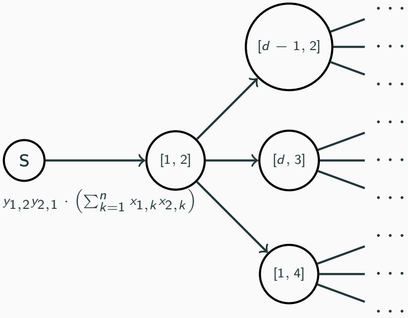
The Hard Polynomial

$y_{1,2}y_{2,1} \cdot \left(\sum_{k=1}^{n} x_{1,k}x_{2,k}\right)$

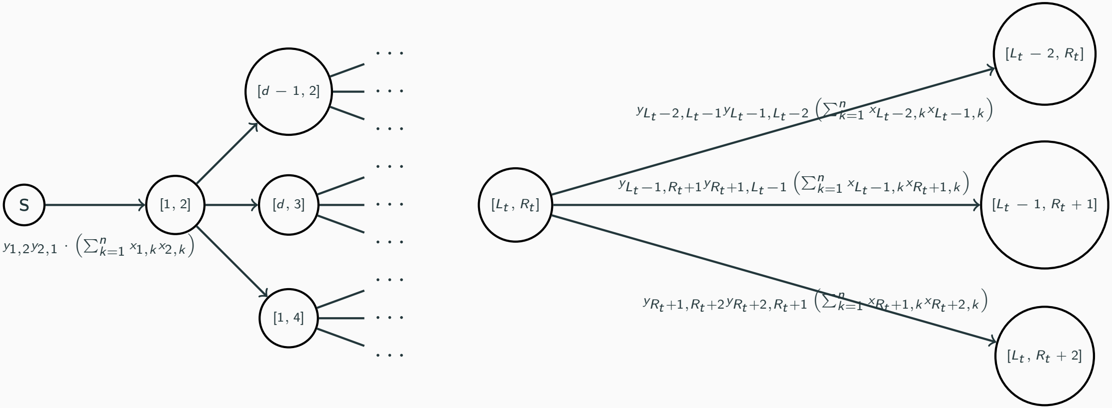
The Hard Polynomial



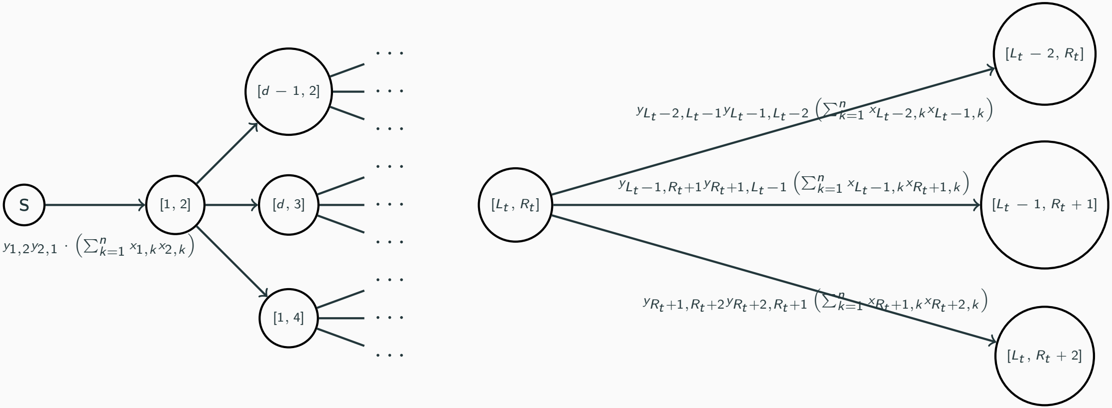
The Hard Polynomial



The Hard Polynomial

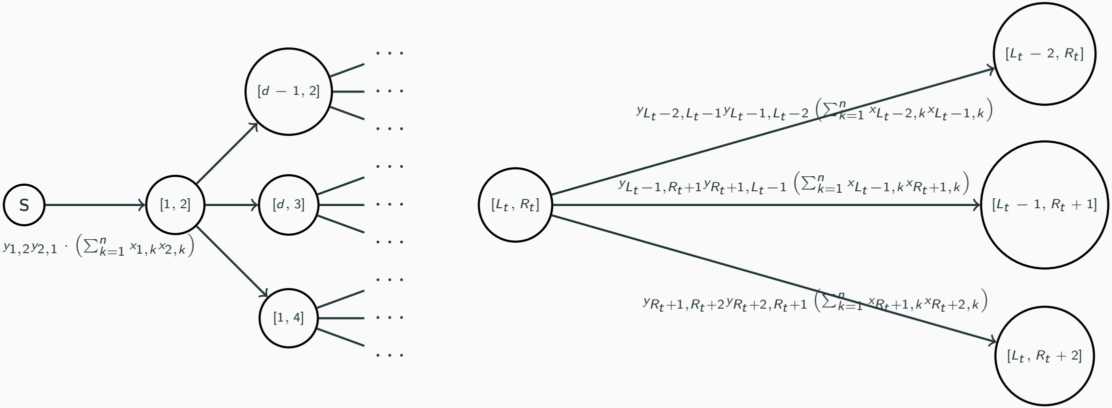


The Hard Polynomial



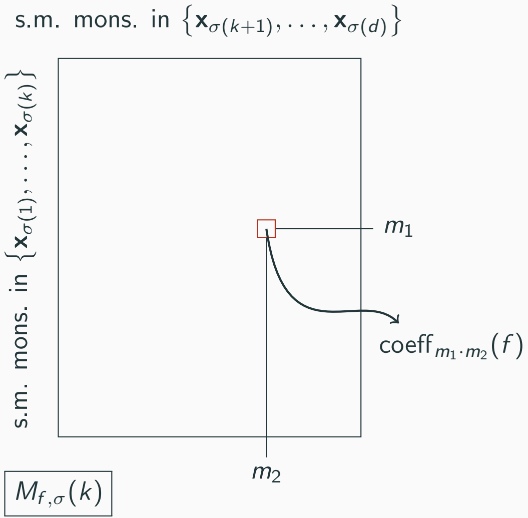
Every path corresponds to a sequence of $d/2$ pairs.

The Hard Polynomial



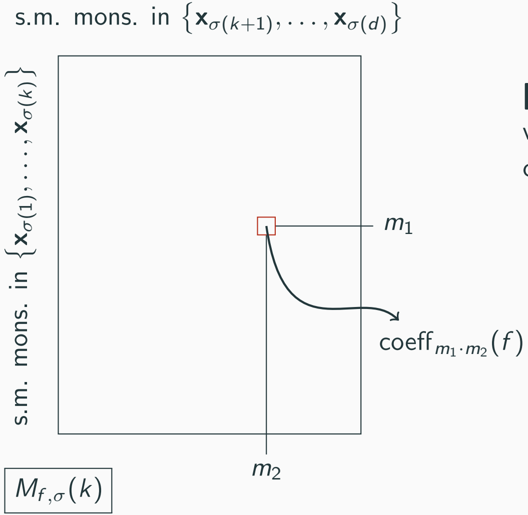
Every path corresponds to a sequence of $d/2$ pairs. $\mathcal{P}_{d/2}$: Set of all such sequences of pairs.

Lower Bound for a single osmABP



f is a set-multilinear poly. w.r.t $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$.

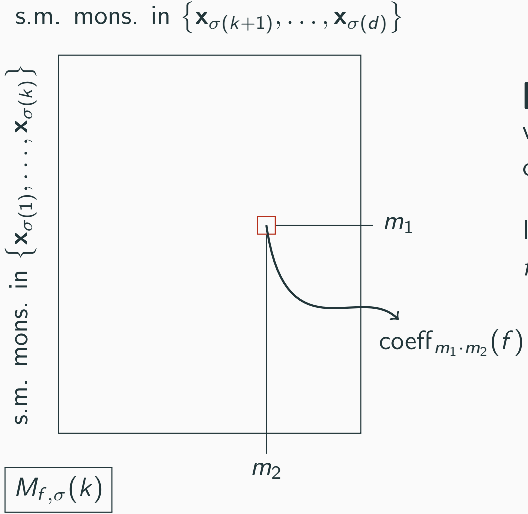
Lower Bound for a single osmABP



f is a set-multilinear poly. w.r.t $\{\mathbf{x}_1, \dots, \mathbf{x}_d\}$.

[Nisan 91]: For every $1 \leq k \leq d$, the number of vertices in the k-th layer of the smallest osmABP(σ) computing f is equal to the rank of $M_{f,\sigma}(k)$.

Lower Bound for a single osmABP



f is a set-multilinear poly. w.r.t $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$.

[Nisan 91]: For every $1 \leq k \leq d$, the number of vertices in the k-th layer of the smallest osmABP(σ) computing f is equal to the rank of $M_{f,\sigma}(k)$.

If $\mathcal{A}$ is the smallest osmABP (in order σ) computing f, then

$$\mathrm{size}(\mathcal{A}) = \sum_{i=1}^{d} \mathrm{rank}(M_{f,\sigma}(k)).$$

Lower Bound for a single osmABP (contd.)

$$G_{n,d} = \sum_{\mathcal{P} \in \mathcal{P}_{d/2}} \prod_{(i,j) \in \mathcal{P}} y_{i,j} y_{j,i} \cdot \left(\sum_{k=1}^{n} x_{i,k} x_{j,k} \right).$$

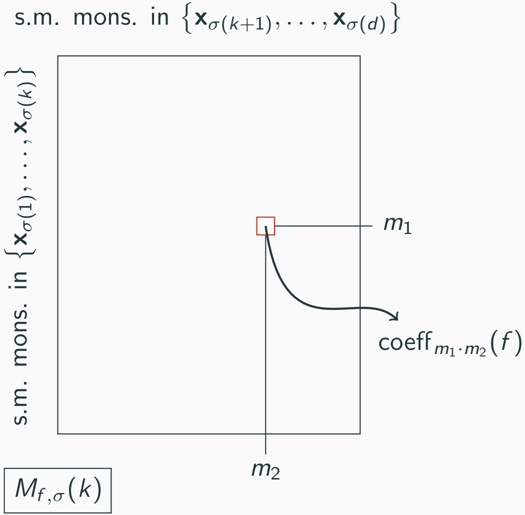
Lower Bound for a single osmABP (contd.)

$$G_{n,d} = \sum_{\mathcal{P} \in \mathcal{P}_{d/2}} \prod_{(i,j) \in \mathcal{P}} y_{i,j} y_{j,i} \cdot \left(\sum_{k=1}^{n} x_{i,k} x_{j,k} \right).$$

Properties:

- $G_{n,d}$ is computable by a set-multilinear ABP of size poly(n, d).

Lower Bound for a single osmABP (contd.)

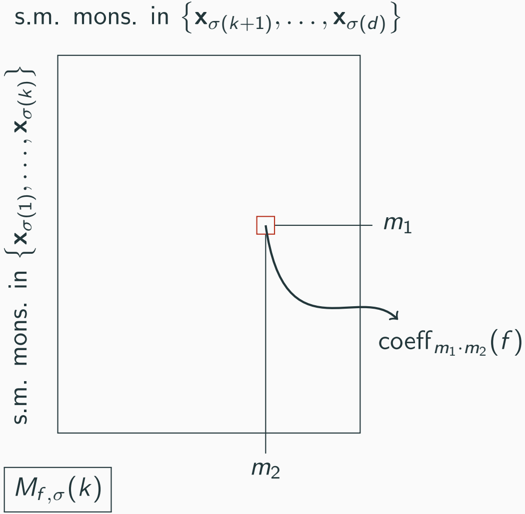


$$G_{n,d} = \sum_{\mathcal{P} \in \mathcal{P}_{d/2}} \prod_{(i,j) \in \mathcal{P}} y_{i,j} y_{j,i} \cdot \left(\sum_{k=1}^{n} x_{i,k} x_{j,k} \right).$$

Properties:

- $G_{n,d}$ is computable by a set-multilinear ABP of size poly(n, d).
- For every $\sigma \in S_d$, there is some $\mathcal{P}$ such that for at least $d/8$ of the $P = (i, j) \in \mathcal{P}$, $i \in \{\sigma(1), \ldots \sigma(\frac{d}{2})\}$ & $j \in \{\sigma(1 + \frac{d}{2})), \ldots \sigma(d)\}$.

Lower Bound for a single osmABP (contd.)



$$G_{n,d} = \sum_{\mathcal{P} \in \mathcal{P}_{d/2}} \prod_{(i,j) \in \mathcal{P}} y_{i,j} y_{j,i} \cdot \left(\sum_{k=1}^{n} x_{i,k} x_{j,k} \right).$$

Properties:

- $G_{n,d}$ is computable by a set-multilinear ABP of size $\mathrm{poly}(n, d)$.
- For every $\sigma \in S_d$, there is some $\mathcal{P}$ such that for at least $d/8$ of the $P = (i,j) \in \mathcal{P}$, $i \in \{\sigma(1), \ldots \sigma(\frac{d}{2})\}$ & $j \in \{\sigma(1 + \frac{d}{2})), \ldots \sigma(d)\}$.

  Therefore,

$$\mathrm{rank}(M_{G_{n,d},\sigma}(d/2)) = \Omega(n^{d/8}).$$

Lower Bound for a Sum of osmABPs

- $\{M_w(f) \;:\; w \in \mathcal{S}\}$ is a set of matrices such that $M_w(G_{n,d})$ has full rank for every $w \in \mathcal{S}$.

Lower Bound for a Sum of osmABPs

- $\{M_w(f) \ : \ w \in \mathcal{S}\}$ is a set of matrices such that $M_w(G_{n,d})$ has full rank for every $w \in \mathcal{S}$.
- If $G_{n,d}$ is computed by a sum of t osmABPs, then

$$G_{n,d} = \sum_{i=1}^{t} g_i \quad \text{where} \quad g_i = \sum_{u_1,\ldots,u_{q-1}} \prod_{j=1}^{q} g^{(i)}_{u_{j-1},u_j}.$$

Lower Bound for a Sum of osmABPs

- $\{M_w(f) \ : \ w \in \mathcal{S}\}$ is a set of matrices such that $M_w(G_{n,d})$ has full rank for every $w \in \mathcal{S}$.
- If $G_{n,d}$ is computed by a sum of t osmABPs, then

$$G_{n,d} = \sum_{i=1}^{t} g_i \quad \text{where} \quad g_i = \sum_{u_1,\ldots,u_{q-1}} \prod_{j=1}^{q} g^{(i)}_{u_{j-1},u_j}.$$

- Define a distribution $\mathcal{D}$ on $\mathcal{S}$ such that when $w \sim \mathcal{D}$, if g_is are computable by osmABPs efficiently, then

Lower Bound for a Sum of osmABPs

- $\{M_w(f) \ : \ w \in \mathcal{S}\}$ is a set of matrices such that $M_w(G_{n,d})$ has full rank for every $w \in \mathcal{S}$.
- If $G_{n,d}$ is computed by a sum of t osmABPs, then

$$G_{n,d} = \sum_{i=1}^{t} g_i \quad \text{where} \quad g_i = \sum_{u_1,\ldots,u_{q-1}} \prod_{j=1}^{q} g^{(i)}_{u_{j-1},u_j}.$$

- Define a distribution $\mathcal{D}$ on $\mathcal{S}$ such that when $w \sim \mathcal{D}$, if g_is are computable by osmABPs efficiently, then

for every i, w.h.p. there are many js, for which $M_w(g^{(i)}_{u_{j-1},u_j})$ is far from full rank

Lower Bound for a Sum of osmABPs

- $\{M_w(f) \ : \ w \in \mathcal{S}\}$ is a set of matrices such that $M_w(G_{n,d})$ has full rank for every $w \in \mathcal{S}$.
- If $G_{n,d}$ is computed by a sum of t osmABPs, then

$$G_{n,d} = \sum_{i=1}^{t} g_i \quad \text{where} \quad g_i = \sum_{u_1,\ldots,u_{q-1}} \prod_{j=1}^{q} g^{(i)}_{u_{j-1},u_j}.$$

- Define a distribution $\mathcal{D}$ on $\mathcal{S}$ such that when $w \sim \mathcal{D}$, if g_is are computable by osmABPs efficiently, then

for every i, w.h.p. there are many js, for which $M_w(g^{(i)}_{u_{j-1},u_j})$ is far from full rank

$\implies$ for every i, w.h.p. $M_w(g_i)$ is far from full rank

Lower Bound for a Sum of osmABPs

- $\{M_w(f) \ : \ w \in \mathcal{S}\}$ is a set of matrices such that $M_w(G_{n,d})$ has full rank for every $w \in \mathcal{S}$.
- If $G_{n,d}$ is computed by a sum of t osmABPs, then

$$G_{n,d} = \sum_{i=1}^{t} g_i \quad \text{where} \quad g_i = \sum_{u_1,\ldots,u_{q-1}} \prod_{j=1}^{q} g^{(i)}_{u_{j-1},u_j}.$$

- Define a distribution $\mathcal{D}$ on $\mathcal{S}$ such that when $w \sim \mathcal{D}$, if g_is are computable by osmABPs efficiently, then

for every i, w.h.p. there are many js, for which $M_w(g^{(i)}_{u_{j-1},u_j})$ is far from full rank

$\implies$ for every i, w.h.p. $M_w(g_i)$ is far from full rank

$\implies$ $M_w(G_{n,d})$ is far from full rank unless t is large.

Improved Lower Bound against Homogeneous Non-Commutative Circuits

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Improved Lower Bound against Homogeneous Non-Commutative Circuits

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

Improved Lower Bound against Homogeneous Non-Commutative Circuits

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \leq i_1 < \cdots < i_d \leq n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \leq \frac{n}{2}$.

Improved Lower Bound against Homogeneous Non-Commutative Circuits

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \leq i_1 < \cdots < i_d \leq n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \leq \frac{n}{2}$. The lower bound is tight for homogeneous non-commutative circuits.

Improved Lower Bound against Homogeneous Non-Commutative Circuits

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates, additionally, respect the order.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \leq i_1 < \cdots < i_d \leq n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \leq \frac{n}{2}$. The lower bound is tight for homogeneous non-commutative circuits.

[Carmosino-Impagliazzo-Lovett-Mihajlin 18]

$\Omega(n^{\frac{\omega}{2}+\varepsilon})$ lower bound for an n-variate, degree-poly(n) polynomial $\implies$ arbitrarily large poly(n) lower bound for n-variate, degree-n polynomial.

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

The Measure

f: Hom. non-commutative polynomial of degree d.

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

The Measure

f: Hom. non-commutative polynomial of degree d.

$f^{(i)}$: Polynomial got from f by setting variables in positions other than i, $i+1$ to 1.

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

<u>**The Measure**</u>

f: Hom. non-commutative polynomial of degree d.

$f^{(i)}$: Polynomial got from f by setting variables in positions other than i, $i+1$ to 1.

Example: $f = x_1 x_2 \cdots x_d + x_d x_{d-1} \cdots x_1$

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

The Measure

f: Hom. non-commutative polynomial of degree d.

$f^{(i)}$: Polynomial got from f by setting variables in positions other than i, $i+1$ to 1.

Example: $f = x_1 x_2 \cdots x_d + x_d x_{d-1} \cdots x_1 \implies f^{(0)} = x_1 + x_d$

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \leq i_1 < \cdots < i_d \leq n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \leq \frac{n}{2}$.

The Measure

f: Hom. non-commutative polynomial of degree d.

$f^{(i)}$: Polynomial got from f by setting variables in positions other than i, $i+1$ to 1.

Example: $f = x_1 x_2 \cdots x_d + x_d x_{d-1} \cdots x_1 \implies f^{(0)} = x_1 + x_d, \quad f^{(1)} = x_1 x_2 + x_d x_{d-1}.$

Our Measure

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$.

The Measure

f: Hom. non-commutative polynomial of degree d.

$f^{(i)}$: Polynomial got from f by setting variables in positions other than i, $i+1$ to 1.

Example: $f = x_1 x_2 \cdots x_d + x_d x_{d-1} \cdots x_1 \implies f^{(0)} = x_1 + x_d, \quad f^{(1)} = x_1 x_2 + x_d x_{d-1}$.

$$\mu(f) = \mathrm{rank}\left(\mathrm{span}_{\mathbb{F}}\left(\left\{f^{(0)}, f^{(1)}, \ldots, f^{(d)}\right\}\right)\right).$$

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}$.

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}$.

- Suppose a similar result was true in the homogeneous non-commutative setting.

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}$.

- Suppose a similar result was true in the homogeneous non-commutative setting.
- Suppose there is an n-variate, degree-d polynomial f such that

$$\mu(\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}) \geq \Omega(nd).$$

Proof Overview

Main Observation: If $f_1, \dots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \dots, f_k) \le s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \dots, \partial_{x_n} f\}$.

- Suppose a similar result was true in the homogeneous non-commutative setting.
- Suppose there is an n-variate, degree-d polynomial f such that

$$\mu(\{\partial_{x_1} f, \partial_{x_2} f, \dots, \partial_{x_n} f\}) \ge \Omega(nd).$$

Then we would have an $\Omega(nd)$ lower bound against homogeneous non-commutative circuits.

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}$.

- A similar result is true in the homogeneous non-commutative setting.
- Suppose there is an n-variate, degree-d polynomial f such that

$$\mu(\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}) \geq \Omega(nd).$$

Then we would have an $\Omega(nd)$ lower bound against homogeneous non-commutative circuits.

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}$.

- A similar result is true in the homogeneous non-commutative setting.
- There is an n-variate, degree-d polynomial f such that

$$\mu(\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}) \geq \Omega(nd).$$

Then we would have an $\Omega(nd)$ lower bound against homogeneous non-commutative circuits.

Proof Overview

Main Observation: If $f_1, \ldots, f_k$ are simultaneously computable by a homogeneous non-commutative circuit of size s,

$$\mu(f_1, \ldots, f_k) \leq s + 1.$$

[Baur-Strassen 83]: If there is a circuit of size s computing $f \in \mathbb{F}[\mathbf{x}]$, then there is a circuit of size at most $5s$ that simultaneously compute $\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}$.

- A similar result is true in the homogeneous non-commutative setting.
- There is an n-variate, degree-d polynomial f such that

$$\mu(\{\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f\}) \geq \Omega(nd).$$

Therefore we have an $\Omega(nd)$ lower bound against homogeneous non-commutative circuits.

Upper Bounding the Measure

$\mathcal{C}$: Homogeneous non-commutative circuit.

$$\mu(\mathcal{C}) = \mathsf{rank}\left(\mathsf{span}_{\mathbb{F}}\left(\bigcup_{g \in \mathcal{C}} \left\{g^{(0)}, g^{(1)}, \ldots, g^{(d)}\right\}\right)\right).$$

Upper Bounding the Measure

$\mathcal{C}$: Homogeneous non-commutative circuit.

$$\mu(\mathcal{C}) = \mathsf{rank}\left(\mathsf{span}_{\mathbb{F}}\left(\bigcup_{g \in \mathcal{C}} \left\{g^{(0)}, g^{(1)}, \ldots, g^{(d)}\right\}\right)\right).$$

Note: $\mu(f_{\mathcal{C}}) \leq \mu(\mathcal{C})$.

Upper Bounding the Measure

$\mathcal{C}$: Homogeneous non-commutative circuit.

$$\mu(\mathcal{C}) = \mathsf{rank}\left(\mathsf{span}_{\mathbb{F}}\left(\bigcup_{g \in \mathcal{C}} \left\{g^{(0)}, g^{(1)}, \ldots, g^{(d)}\right\}\right)\right).$$

Note: $\mu(f_{\mathcal{C}}) \leq \mu(\mathcal{C})$.

Need to show: $\mu(\mathcal{C}) \leq \mathsf{size}(\mathcal{C}) + 1$.

Upper Bounding the Measure

$\mathcal{C}$: Homogeneous non-commutative circuit.

$$\mu(\mathcal{C}) = \mathsf{rank}\left(\mathsf{span}_{\mathbb{F}}\left(\bigcup_{g\in\mathcal{C}}\left\{g^{(0)}, g^{(1)}, \ldots, g^{(d)}\right\}\right)\right).$$

Note: $\mu(f_{\mathcal{C}}) \leq \mu(\mathcal{C})$.

Need to show: $\mu(\mathcal{C}) \leq \mathsf{size}(\mathcal{C}) + 1$.

Idea: Use induction

Upper Bounding the Measure

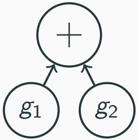
$\mathcal{C}$: Homogeneous non-commutative circuit.

$$\mu(\mathcal{C}) = \text{rank}\left(\text{span}_{\mathbb{F}}\left(\bigcup_{g\in\mathcal{C}}\left\{g^{(0)}, g^{(1)}, \ldots, g^{(d)}\right\}\right)\right).$$

Note: $\mu(f_{\mathcal{C}}) \leq \mu(\mathcal{C})$.

Need to show: $\mu(\mathcal{C}) \leq \text{size}(\mathcal{C}) + 1$.

Idea: Use induction



Upper Bounding the Measure

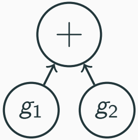
$\mathcal{C}$: Homogeneous non-commutative circuit.

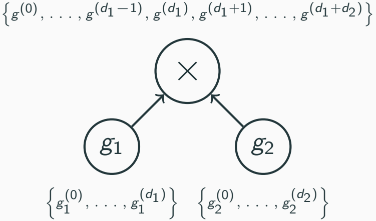
$$\mu(\mathcal{C}) = \text{rank}\left(\text{span}_{\mathbb{F}}\left(\bigcup_{g \in \mathcal{C}} \left\{g^{(0)}, g^{(1)}, \ldots, g^{(d)}\right\}\right)\right).$$

Note: $\mu(f_{\mathcal{C}}) \leq \mu(\mathcal{C})$.

Need to show: $\mu(\mathcal{C}) \leq \text{size}(\mathcal{C}) + 1$.

Idea: Use induction





Open Questions in Algebraic Complexity

Some Open Directions

- Better lower bounds against homogeneous formulas?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Better lower bounds against set-multilinear ABPs?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Better lower bounds against set-multilinear ABPs?
- PIT for $\sum$ osmABP?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Better lower bounds against set-multilinear ABPs?
- PIT for $\sum$osmABP?
- Bootstrapping statement, similar to **[C-I-L-M 18]**, which is sensitive to both degree and number of variables?

Some Open Directions

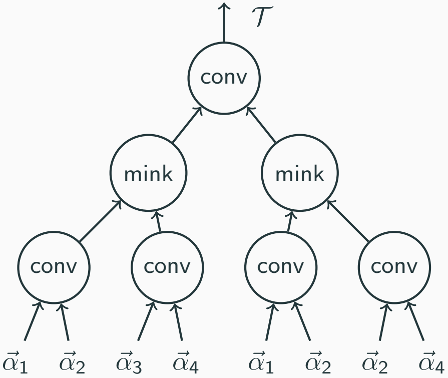
- Better lower bounds against homogeneous formulas?
- Better lower bounds against set-multilinear ABPs?
- PIT for $\sum$ osmABP?
- Bootstrapping statement, similar to **[C-I-L-M 18]**, which is sensitive to both degree and number of variables?
- Separating formulas and ABPs in the non-commutative setting?

Some Open Directions

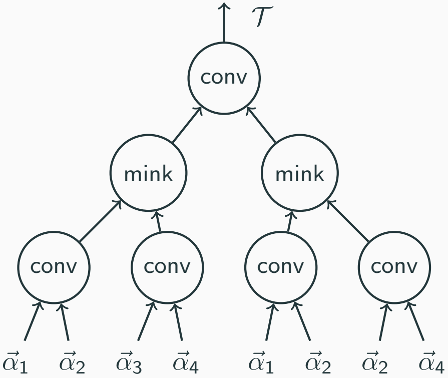
- Better lower bounds against homogeneous formulas?
- Better lower bounds against set-multilinear ABPs?
- PIT for $\sum$osmABP?
- Bootstrapping statement, similar to **[C-I-L-M 18]**, which is sensitive to both degree and number of variables?
- Separating formulas and ABPs in the non-commutative setting?
- Meaningful definition of VPH?

Branching Out

Upper Bounding Vertices in Some Structured Polytopes



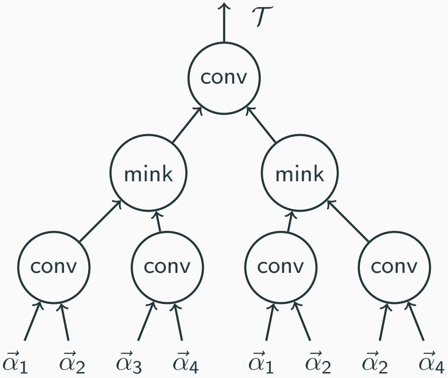
Upper Bounding Vertices in Some Structured Polytopes



[C-Gajjar-Radhakrishnan] (ongoing work):
Let $d \geq 2$ and let $\mathcal{T}$ be a computational tree over $\mathbb{R}^d$ such that $\text{depth}^*(\mathcal{F}) \geq 1$. Then, the number of vertices in $\mathrm{P}(\mathcal{T})$ is at most

$$\text{size}(\mathcal{T})^{4\ \text{depth}^*(\mathcal{T})^{d-2}}.$$

Upper Bounding Vertices in Some Structured Polytopes



[C-Gajjar-Radhakrishnan] (ongoing work):
Let $d \geq 2$ and let $\mathcal{T}$ be a computational tree over $\mathbb{R}^d$ such that $\text{depth}^*(\mathcal{F}) \geq 1$. Then, the number of vertices in $\mathrm{P}(\mathcal{T})$ is at most

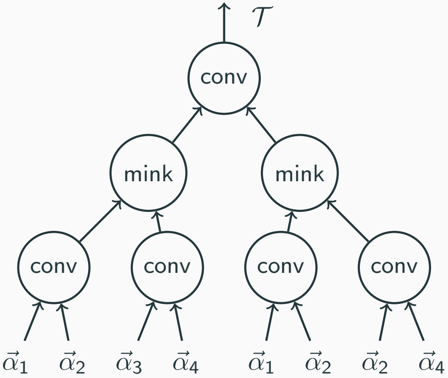
$$\text{size}(\mathcal{T})^{4\ \text{depth}^*(\mathcal{T})^{d-2}}.$$

Corollary: Let G be a directed graph on n vertices with two special vertices s and t, and edge weights of the form

$$w_e(\lambda_1, \lambda_2, \ldots, \lambda_d) = a_{1,e}\lambda_1 + a_{2,e}\lambda_2 + \ldots + a_{d,e}\lambda_d.$$

Then the number of different shortest s–t paths in G (as $\lambda_1, \ldots, \lambda_d$ varies) is at most $n^{4(\log n)^{d-1}}$.

Upper Bounding Vertices in Some Structured Polytopes



Note: Known to be tight for $d = 2$.

[C-Gajjar-Radhakrishnan] (ongoing work):
Let $d \geq 2$ and let $\mathcal{T}$ be a computational tree over $\mathbb{R}^d$ such that depth$^*(\mathcal{F}) \geq 1$. Then, the number of vertices in $\mathrm{P}(\mathcal{T})$ is at most

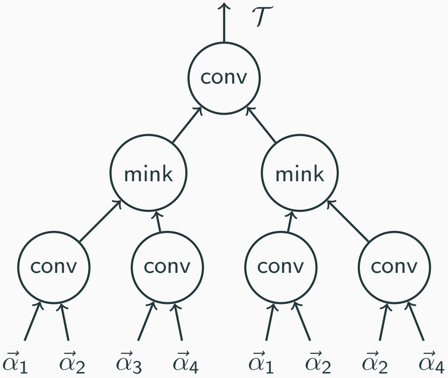
$$\mathrm{size}(\mathcal{T})^{4\ \mathrm{depth}^*(\mathcal{T})^{d-2}}.$$

Corollary: Let G be a directed graph on n vertices with two special vertices s and t, and edge weights of the form

$$w_e(\lambda_1, \lambda_2, \ldots, \lambda_d) = a_{1,e}\lambda_1 + a_{2,e}\lambda_2 + \ldots + a_{d,e}\lambda_d.$$

Then the number of different shortest s–t paths in G (as $\lambda_1, \ldots, \lambda_d$ varies) is at most $n^{4(\log n)^{d-1}}$.

Upper Bounding Vertices in Some Structured Polytopes



Note: Known to be tight for $d = 2$.
Open for $d \geq 3$.

[C-Gajjar-Radhakrishnan] (ongoing work):
Let $d \geq 2$ and let $\mathcal{T}$ be a computational tree over $\mathbb{R}^d$ such that depth$^*(\mathcal{F}) \geq 1$. Then, the number of vertices in $\mathrm{P}(\mathcal{T})$ is at most

$$\mathrm{size}(\mathcal{T})^{4\ \mathrm{depth}^*(\mathcal{T})^{d-2}}.$$

Corollary: Let G be a directed graph on n vertices with two special vertices s and t, and edge weights of the form

$$w_e(\lambda_1, \lambda_2, \ldots, \lambda_d) = a_{1,e}\lambda_1 + a_{2,e}\lambda_2 + \ldots + a_{d,e}\lambda_d.$$

Then the number of different shortest s–t paths in G (as $\lambda_1, \ldots, \lambda_d$ varies) is at most $n^{4(\log n)^{d-1}}$.

What Next??

Most questions that are theoretical in nature interest me!

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

$$f = (x \wedge z) \vee (y \wedge \neg z)$$

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

$$f = (x \wedge z) \vee (y \wedge \neg z)$$

- $f(1, 1, 1) = 1 = f(1, 1, 0)$.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

$$f = (x \wedge z) \vee (y \wedge \neg z)$$

- $f(1, 1, 1) = 1 = f(1, 1, 0)$.
- For $\mathcal{C} \equiv (x \wedge z) \vee (y \wedge \neg z)$, $\mathcal{C}(1, 1, u) = u$.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

$$f = (x \wedge z) \vee (y \wedge \neg z)$$

- $f(1, 1, 1) = 1 = f(1, 1, 0)$.
- For $\mathcal{C} \equiv (x \wedge z) \vee (y \wedge \neg z)$, $\mathcal{C}(1, 1, u) = u$.
  - $\mathcal{C}$ has a hazard at $(1, 1, u)$.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

$$f = (x \wedge z) \vee (y \wedge \neg z)$$

- $f(1, 1, 1) = 1 = f(1, 1, 0)$.
- For $\mathcal{C} \equiv (x \wedge z) \vee (y \wedge \neg z)$, $\mathcal{C}(1, 1, u) = u$.
  - $\mathcal{C}$ has a hazard at $(1, 1, u)$.
- Let $\mathcal{C}' \equiv (x \wedge (y \vee z)) \vee (y \wedge \neg z)$.

What Next??

Most questions that are theoretical in nature interest me!

What I like doing the most: Abstracting out concrete questions to create mathematical models and studying them.

Boolean Circuits and Hazards

- Circuits where the firing of input gates might be delayed.
- Include symbol $u \equiv 0/1$.
- Define $\wedge, \vee, \neg : \{0, 1, u\}^2 \mapsto \{0, 1, u\}$ meaningfully.

$$f = (x \wedge z) \vee (y \wedge \neg z)$$

- $f(1, 1, 1) = 1 = f(1, 1, 0)$.
- For $\mathcal{C} \equiv (x \wedge z) \vee (y \wedge \neg z)$, $\mathcal{C}(1, 1, u) = u$.
  - $\mathcal{C}$ has a hazard at $(1, 1, u)$.
- Let $\mathcal{C}' \equiv (x \wedge (y \vee z)) \vee (y \wedge \neg z)$.
  - $\mathcal{C}'$ is hazard-free.

Teaching

Courses I would be happy to teach

Basic Courses

- Discrete Structures
- Automata Theory
- Data Structures and Algorithms
- Theory of Computation
- Algorithms and Complexity
- Automata Theory and Logic
- Computer Programming
- Formal Methods in CS
- Numerical Computation

Courses I would be happy to teach

Basic Courses

- Discrete Structures
- Automata Theory
- Data Structures and Algorithms
- Theory of Computation
- Algorithms and Complexity
- Automata Theory and Logic
- Computer Programming
- Formal Methods in CS
- Numerical Computation

Advanced Courses

- Applied Algorithms
- Topics in Complexity Theory
- Randomness in Computation
- Algebra in Computation
- Pseudorandomness

Courses I would be happy to teach

Basic Courses

- Discrete Structures
- Automata Theory
- Data Structures and Algorithms
- Theory of Computation
- Algorithms and Complexity
- Automata Theory and Logic
- Computer Programming
- Formal Methods in CS
- Numerical Computation

Advanced Courses

- Applied Algorithms
- Topics in Complexity Theory
- Randomness in Computation
- Algebra in Computation
- Pseudorandomness

Research Level Courses

- Communication Complexity
- Circuit Complexity
- Algebraic Complexity Theory

Courses I would be happy to teach

Basic Courses

- Discrete Structures
- Automata Theory
- Data Structures and Algorithms
- Theory of Computation
- Algorithms and Complexity
- Automata Theory and Logic
- Computer Programming
- Formal Methods in CS
- Numerical Computation

Advanced Courses

- Applied Algorithms
- Topics in Complexity Theory
- Randomness in Computation
- Algebra in Computation
- Pseudorandomness

Research Level Courses

- Communication Complexity
- Circuit Complexity
- Algebraic Complexity Theory

I would be happy to teach/design other courses depending on interest and/or requirement.