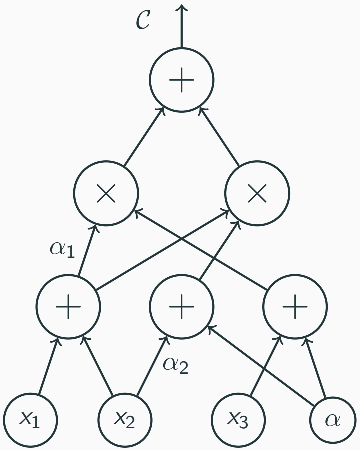
# Separating ABPs and some Structured Formulas in the Non-Commutative Setting

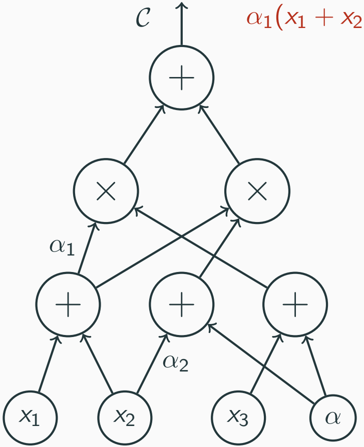
**Prerona Chatterjee**
Institute of Mathematics, Czech Academy of Sciences

June 29, 2022

# Algebraic Models of Computation

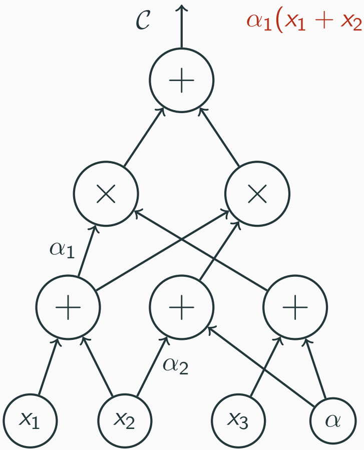


# Algebraic Models of Computation

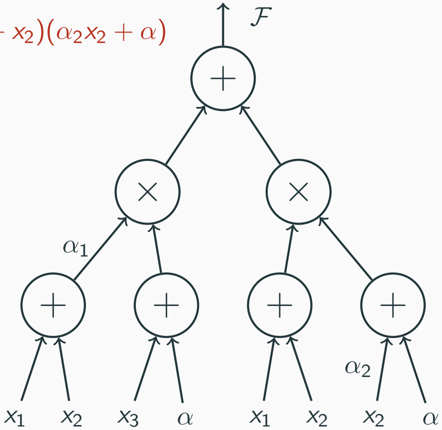



$$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$$

# Algebraic Models of Computation




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



# Lower Bounds in Algebraic Circuit Complexity

**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 $\mathrm{poly}(n, d)$.

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

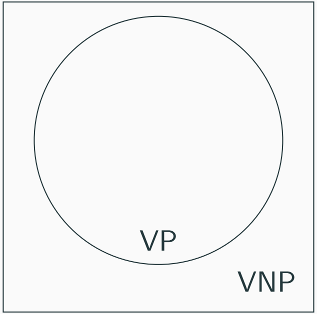
VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.


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

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

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

VNP: Explicit Polynomials



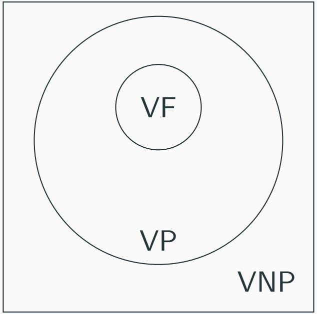

**Central Question**: Find explicit polynomials that cannot be computed by efficient circuits.

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

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

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

VNP: Explicit Polynomials



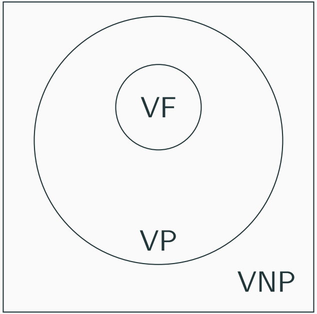

**Central Question**: Find explicit polynomials that cannot be computed by efficient circuits.

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

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

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

VNP: Explicit Polynomials




Are the inclusions tight?

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

# The Current Best Bounds

### General Circuits

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

# The Current Best Bounds

**General Circuits**

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

**General Formulas**

**[C-Kumar-She-Volk]**: Any formula computing $\sum_{\substack{S \subseteq [n] \\ |S| = 0.1n}} \prod_{i \in S} x_i$ requires $\Omega(n^2)$ vertices.

# The Current Best Bounds

**General Circuits**

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

**General Formulas**

**[C-Kumar-She-Volk]**: Any formula computing $\sum_{\substack{S \subseteq [n] \\ |S| = 0.1n}} \prod_{i \in S} x_i$ requires $\Omega(n^2)$ vertices.

**Note**: There is a circuit of size $O(n \log^2 n)$ computing $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$.

# The Current Best Bounds

**General Circuits**

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

**General Formulas**

**[C-Kumar-She-Volk]**: Any formula computing $\sum_{\substack{S \subseteq [n] \\ |S| = 0.1n}} \prod_{i \in S} x_i$ requires $\Omega(n^2)$ vertices.

**Note**: There is a circuit of size $O(n \log^2 n)$ computing $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$. Therefore this shows a super-linear separation between formulas and circuits for a multilinear polynomial.

# The Non-Commutative Setting

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

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

**Formulas** [Nisan]: $2^{\Omega(n)}$ for a 2-variate, degree $n$ polynomial in $\mathsf{VP}_{\mathsf{nc}}$.

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

**Formulas** [Nisan]: $2^{\Omega(n)}$ for a 2-variate, degree $n$ polynomial in $\mathsf{VP}_{\mathsf{nc}}$. So, $\mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VP}_{\mathsf{nc}}$.

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

**Formulas** [Nisan]: $2^{\Omega(n)}$ for a 2-variate, degree $n$ polynomial in $\mathsf{VP}_{\mathsf{nc}}$. So, $\mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VP}_{\mathsf{nc}}$.

But the proof is via a lower bound against non-commutative Algebraic Branching Programs.

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

**Formulas** [Nisan]: $2^{\Omega(n)}$ for a 2-variate, degree $n$ polynomial in $\mathrm{VP_{nc}}$. So, $\mathrm{VF_{nc}} \neq \mathrm{VP_{nc}}$.

But the proof is via a lower bound against non-commutative Algebraic Branching Programs.

$\mathrm{VBP_{nc}}$

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

**Formulas** [Nisan]: $2^{\Omega(n)}$ for a 2-variate, degree $n$ polynomial in $\mathsf{VP}_{\mathsf{nc}}$. So, $\mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VP}_{\mathsf{nc}}$.

But the proof is via a lower bound against non-commutative Algebraic Branching Programs.

$$\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{VBP}_{\mathsf{nc}} \subseteq \mathsf{VP}_{\mathsf{nc}}$$

# The Non-Commutative Setting

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

**Lower Bounds for General Models**

**Circuits** [Baur-Strassen]: $\Omega(n \log d)$ for an $n$-variate, degree $d$ polynomial.

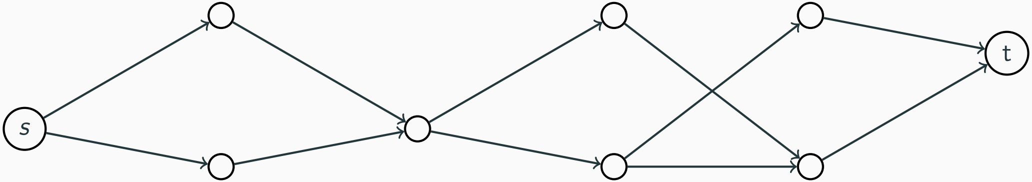
**Formulas** [Nisan]: $2^{\Omega(n)}$ for a 2-variate, degree $n$ polynomial in $\mathrm{VP}_{\mathrm{nc}}$. So, $\mathrm{VF}_{\mathrm{nc}} \neq \mathrm{VP}_{\mathrm{nc}}$.

But the proof is via a lower bound against non-commutative Algebraic Branching Programs.

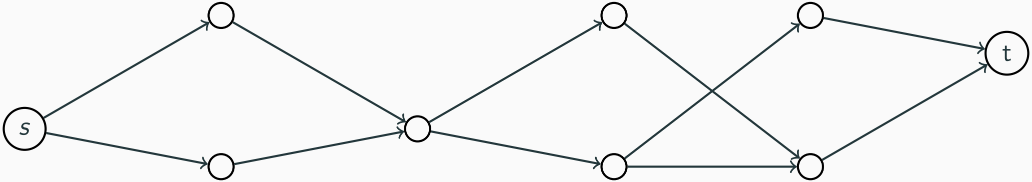
$$\mathrm{VF}_{\mathrm{nc}} \subseteq \mathrm{VBP}_{\mathrm{nc}} \subseteq \mathrm{VP}_{\mathrm{nc}}$$

So Nisan actually showed that $\mathrm{VBP}_{\mathrm{nc}} \neq \mathrm{VP}_{\mathrm{nc}}$.

# Algebraic Branching Programs

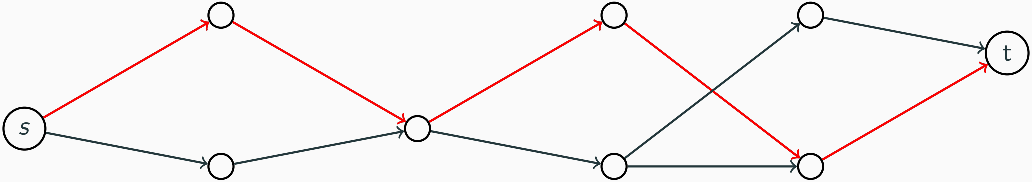


# Algebraic Branching Programs



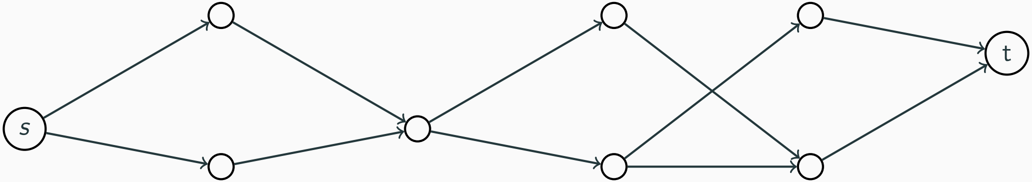

- Label on each edge: Linear polynomials in $\{x_1, x_2, \ldots, x_n\}$

# Algebraic Branching Programs



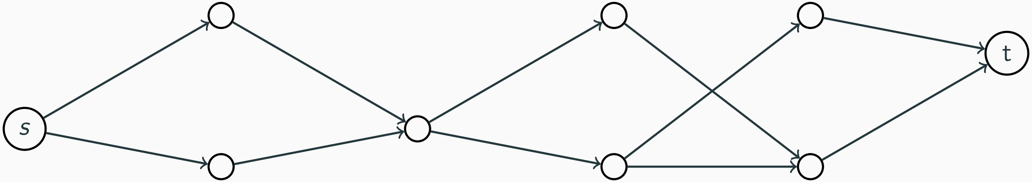

- Label on each edge: Linear polynomials in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on $p$

# Algebraic Branching Programs



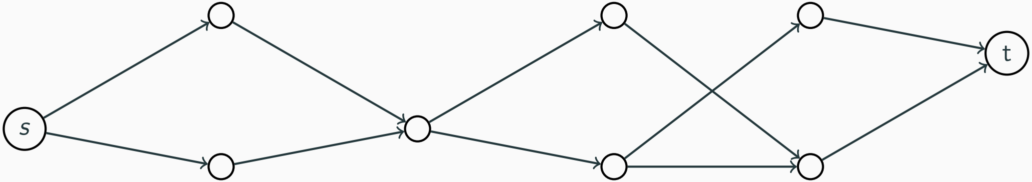

- Label on each edge: Linear polynomials in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \text{wt}(p)$: Product of the edge labels on $p$
- Polynomial computed by the ABP: $\sum_p \text{wt}(p)$

# Algebraic Branching Programs




- Label on each edge: Linear polynomials in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \text{wt}(p)$: Product of the edge labels on $p$
- Polynomial computed by the ABP: $\sum_p \text{wt}(p)$
- Size of the ABP: Number of vertices in the ABP

# Algebraic Branching Programs




- Label on each edge: Linear polynomials in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \text{wt}(p)$: Product of the edge labels on $p$
- Polynomial computed by the ABP: $\sum_p \text{wt}(p)$
- Size of the ABP: Number of vertices in the ABP

# The ABP vs Formulas Question

**Nisan**: 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 $2^{\Omega(n)}$.

# The ABP vs Formulas Question

**Nisan**: 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 $2^{\Omega(n)}$.

In comparison, the best lower bound against ABPs in the commutative setting is just quadratic.

[**C-Kumar-She-Volk**]: Any ABP computing $\sum_{i=1}^{n} x_i^n$ has size $\Omega(n^2)$.

# The ABP vs Formulas Question

**Nisan**: 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 $2^{\Omega(n)}$.

Since $\mathrm{Pal}_n(x_0, x_1)$ can be computed by a non-commutative circuit of size $O(n)$, $\mathsf{VBP}_{\mathsf{nc}} \neq \mathsf{VP}_{\mathsf{nc}}$.

# The ABP vs Formulas Question

**Nisan**: 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 $2^{\Omega(n)}$.

Since $\mathrm{Pal}_n(x_0, x_1)$ can be computed by a non-commutative circuit of size $O(n)$, $\mathsf{VBP}_{\mathsf{nc}} \neq \mathsf{VP}_{\mathsf{nc}}$.

Since $\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{VBP}_{\mathsf{nc}}$, this gives an exponential lower bound against $\mathsf{VF}_{\mathsf{nc}}$.

# The ABP vs Formulas Question

**Nisan**: 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 $2^{\Omega(n)}$.

Since $\mathrm{Pal}_n(x_0, x_1)$ can be computed by a non-commutative circuit of size $O(n)$, $\mathrm{VBP}_{\mathrm{nc}} \neq \mathrm{VP}_{\mathrm{nc}}$.

Since $\mathrm{VF}_{\mathrm{nc}} \subseteq \mathrm{VBP}_{\mathrm{nc}}$, this gives an exponential lower bound against $\mathrm{VF}_{\mathrm{nc}}$.

**The Question**[Nisan]: Is $\mathrm{VBP}_{\mathrm{nc}} = \mathrm{VF}_{\mathrm{nc}}$?

# Possible Approaches

[**Hrubes-Yehudayoff, Raz, Dvir-Malod-Perifel-Yehudayoff**]

We know how to prove lower bounds against multilinear formulas.

# Possible Approaches

**[Hrubes-Yehudayoff, Raz, Dvir-Malod-Perifel-Yehudayoff]**

We know how to prove lower bounds against multilinear formulas.

- It is easy to view non-commutative polynomials as commutative multilinear polynomials:

$$x_i \text{ in position p} \rightarrow x_{p,i}$$

# Possible Approaches

[**Hrubes-Yehudayoff, Raz, Dvir-Malod-Perifel-Yehudayoff**]

We know how to prove lower bounds against multilinear formulas.

- It is easy to view non-commutative polynomials as commutative multilinear polynomials:

$$x_i \text{ in position p} \to x_{p,i}$$

- It is easy to view multilinear polynomials as non-commutative polynomials:

in each monomial, order the variables in ascending order

# Possible Approaches

[**Hrubes-Yehudayoff, Raz, Dvir-Malod-Perifel-Yehudayoff**]

We know how to prove lower bounds against multilinear formulas.

- It is easy to view non-commutative polynomials as commutative multilinear polynomials:

$x_i$ in position p $\to x_{p,i}$

- It is easy to view multilinear polynomials as non-commutative polynomials:

in each monomial, order the variables in ascending order

We only know how to multilinearise formulas when the degree is small [Raz].

# The Homogeneous Case

$x_i$ in position p $\rightarrow x_{p,i}$

# The Homogeneous Case

$x_i$ in position p $\rightarrow x_{p,i}$

Any non-commutative formula is automatically multilinear, in fact set-multilinear.

# The Homogeneous Case

$x_i$ in position p $\rightarrow x_{p,i}$

Any non-commutative formula is automatically multilinear, in fact set-multilinear.

Unclear how to use the previous known techniques even in this case.

# The Homogeneous Case

$x_i$ in position p $\rightarrow x_{p,i}$

Any non-commutative formula is automatically multilinear, in fact set-multilinear.

Unclear how to use the previous known techniques even in this case.

**[Tavenas, Limaye, Srinivasan]**

Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,n}$ must have size $n^{\Omega(\log \log n)}$.

# Main Result

**Note**: Every monomial in a non-commutative polynomial $f(x_1, \ldots, x_n)$ can be thought of as a word over the underlying variables $\{x_1, \ldots, x_n\}$.

# Main Result

**Note**: Every monomial in a non-commutative polynomial $f(x_1, \ldots, x_n)$ can be thought of as a word over the underlying variables $\{x_1, \ldots, x_n\}$.

**Definitions**

# Main Result

**Note**: Every monomial in a non-commutative polynomial $f(x_1, \ldots, x_n)$ can be thought of as a word over the underlying variables $\{x_1, \ldots, x_n\}$.

**Definitions** Let $\{X_1, \ldots, X_m\}$ be a partition of the variables into buckets.

# Main Result

**Note**: Every monomial in a non-commutative polynomial $f(x_1, \ldots, x_n)$ can be thought of as a word over the underlying variables $\{x_1, \ldots, x_n\}$.

**Definitions** Let $\{X_1, \ldots, X_m\}$ be a partition of the variables into buckets.

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

# Main Result

**Note**: Every monomial in a non-commutative polynomial $f(x_1, \ldots, x_n)$ can be thought of as a word over the underlying variables $\{x_1, \ldots, x_n\}$.

**Definitions** Let $\{X_1, \ldots, X_m\}$ be a partition of the variables into buckets.

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

*Syntactically Abecedarian Formulas*: Non-commutative formulas with a syntactic restriction that makes them naturally compute abecedarian polynomials.

# Main Result

**Note**: Every monomial in a non-commutative polynomial $f(x_1, \ldots, x_n)$ can be thought of as a word over the underlying variables $\{x_1, \ldots, x_n\}$.

**Definitions** Let $\{X_1, \ldots, X_m\}$ be a partition of the variables into buckets.

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

*Syntactically Abecedarian Formulas*: Non-commutative formulas with a syntactic restriction that makes them naturally compute abecedarian polynomials.

**Main Result**:

There is a tight superpolynomial separation between *abecedarian* formulas and ABPs.

# Abecedarian Polynomials

Generalises the notion of ordered polynomials ([Hrubes-Wigderson-Yehudayoff]).

# Abecedarian Polynomials

Generalises the notion of ordered polynomials ([Hrubes-Wigderson-Yehudayoff]).

Variables can be partitioned into buckets such that every variable in position $i$ is from bucket $i$.

# Abecedarian Polynomials

Generalises the notion of ordered polynomials ([Hrubes-Wigderson-Yehudayoff]).

$$\mathrm{Det}_n(\mathbf{x}) = \sum_{\sigma \in S_n} (-1)^{\mathrm{sgn}(\sigma)} x_{1,\sigma(1)} \cdots x_{n,\sigma(n)}$$

| **Buckets** | **Example** |
|---|---|
| | |
| $\{X_i\}_{i \in [n]}$ where $X_i = \{x_{ij}\}_{j \in [n]}$ | $\mathrm{Det}_n(\mathbf{x})$ |
| | |

# Abecedarian Polynomials

Generalises the notion of ordered polynomials ([Hrubes-Wigderson-Yehudayoff]).

$$\mathrm{Perm}_n(\mathbf{x}) = \sum_{\sigma \in S_n} x_{1,\sigma(1)} \cdots x_{n,\sigma(n)}$$

| **Buckets** | **Example** |
|---|---|
| | |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_{ij}\}_{j\in[n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| | |

Generalises the notion of ordered polynomials ([Hrubes-Wigderson-Yehudayoff]).

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

| Buckets | Example |
|---|---|
| | |
| $\{X_i\}_{i \in [n]}$ where $X_i = \{x_{ij}\}_{j \in [n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| | |

# Abecedarian Polynomials

Variables in every monomial arranged in non-decreasing order of bucket indices.

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

| **Buckets** | **Example** |
|---|---|
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_{ij}\}_{j\in[n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_i\}$ | $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ |

# Abecedarian Polynomials

Variables in every monomial arranged in non-decreasing order of bucket indices.

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

| Buckets | Example |
|---|---|
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_{ij}\}_{j\in[n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_i\}$ | $\mathrm{CHSYM}_{n,d}(\mathbf{x})$, $\mathrm{ESYM}_{n,d}(\mathbf{x})$ |

# Abecedarian Polynomials

Variables in every monomial arranged in non-decreasing order of bucket indices.

$$f(\mathbf{x}) \xrightarrow[\text{in ascending order}]{\text{Order the monomials}} f^{(nc)}(\mathbf{x})$$

| **Buckets** | **Example** |
|---|---|
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_{ij}\}_{j\in[n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_i\}$ | $\mathrm{CHSYM}_{n,d}(\mathbf{x})$, $\mathrm{ESYM}_{n,d}(\mathbf{x})$<br>Non-Commutative version of any $f \in \mathbb{F}[x_1, \ldots, x_n]$ |

# Abecedarian Polynomials

| Buckets | Example |
| --- | --- |
| | |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_{ij}\}_{j\in[n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_i\}$ | $\mathrm{CHSYM}_{n,d}(\mathbf{x})$, $\mathrm{ESYM}_{n,d}(\mathbf{x})$<br>Non-Commutative version of any $f \in \mathbb{F}[x_1, \ldots, x_n]$ |

**Note**:

$$\mathrm{ESYM}^{(\mathrm{ord})}_{n,d} = \sum_{1\le i_1<\ldots<i_d\le n} x^{(1)}_{i_1}\cdots x^{(d)}_{i_d}$$

is abecedarian w.r.t. both $\left\{X_k = \left\{x_i^{(k)}\right\}_{i\in[n]}\right\}_{k\in[d]}$ as well as $\left\{X_i = \left\{x_i^{(k)}\right\}_{k\in[d]}\right\}_{i\in[n]}$.

# Abecedarian Polynomials

Abecedarian Polynomials: Non-commutative polynomials in which variables in every monomial arranged in non-decreasing order of bucket indices.

| **Buckets** | **Example** |
|---|---|
| | |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_{ij}\}_{j\in[n]}$ | $\mathrm{Det}_n(\mathbf{x})$, $\mathrm{Perm}_n(\mathbf{x})$ |
| $\{X_i\}_{i\in[n]}$ where $X_i = \{x_i\}$ | $\mathrm{CHSYM}_{n,d}(\mathbf{x})$, $\mathrm{ESYM}_{n,d}(\mathbf{x})$<br>Non-Commutative version of any $f \in \mathbb{F}[x_1, \ldots, x_n]$ |

Abecedarian Formulas: Non-commutative formulas with a syntactic restriction that makes them naturally compute abecedarian polynomials.

# The Explicit Statement

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left( \sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

# The Explicit Statement

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left( \sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

- Abecedarian with respect to $\{X_i \ : \ 1 \le i \le n\}$ where $X_i = \{x_{ij} \ : \ 1 \le j \le n\}$.

## The Explicit Statement

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left( \sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

- Abecedarian with respect to $\{X_i \ : \ 1 \le i \le n\}$ where $X_i = \{x_{ij} \ : \ 1 \le j \le n\}$.
- There is an abecedarian ABP of size $O(nd)$ that computes $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$.

# The Explicit Statement

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left( \sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

- Abecedarian with respect to $\{X_i \ : \ 1 \le i \le n\}$ where $X_i = \{x_{ij} \ : \ 1 \le j \le n\}$.
- There is an abecedarian ABP of size $O(nd)$ that computes $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$.
- Any abecedarian formula computing $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ has size $n^{\Omega(\log \log n)}$.

# The Explicit Statement

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left( \sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

- Abecedarian with respect to $\{X_i \ : \ 1 \le i \le n\}$ where $X_i = \{x_{ij} \ : \ 1 \le j \le n\}$.
- There is an abecedarian ABP of size $O(nd)$ that computes $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$.
- Any abecedarian formula computing $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ has size $n^{\Omega(\log\log n)}$.
- There is an abecedarian formula of size $n^{O(\log\log n)}$ that computes $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$.

# The Explicit Statement

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left( \sum_{i_0 \leq i_1 \leq \ldots \leq i_d \leq n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

- Abecedarian with respect to $\{X_i \ : \ 1 \leq i \leq n\}$ where $X_i = \{x_{ij} \ : \ 1 \leq j \leq n\}$.
- There is an abecedarian ABP of size $O(nd)$ that computes $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$.
- Any abecedarian formula computing $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$.

If a formula of size $s$ computes a polynomial that is abecedarian with respect to a partition of size $O(\log n)$, then it can be converted into an abecedarian formula of size $\text{poly}(s)$.

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.

- Use the lower bound against homogeneous multilinear formulas for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ [HY11].

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.

- There is a small homogeneous multilinear formula computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.
- Use the lower bound against homogeneous multilinear formulas for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ [HY11].

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2, \log n}(\mathbf{x})$.

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

- There is a small homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.
- There is a small homogeneous multilinear formula computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.
- Use the lower bound against homogeneous multilinear formulas for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ [HY11].

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.
- Convert to a small homogeneous structured abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.

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

- There is a small homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.
- There is a small homogeneous multilinear formula computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.
- Use the lower bound against homogeneous multilinear formulas for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ [HY11].

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.
- Convert to a small homogeneous structured abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.
- There is a small homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.

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

- There is a small homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.
- There is a small homogeneous multilinear formula computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.
- Use the lower bound against homogeneous multilinear formulas for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ [HY11].

# Proof Idea of the Abecedarian Formula Lower Bound

- Assume that there is a small abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.
- Convert to a small homogeneous structured abecedarian formula computing $h_{n/2,\log n}(\mathbf{x})$.
- There is a small homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.

If there is a homogeneous structured abecedarian formula of size $s$ computing $h_{n/2,d}(\mathbf{x})$ and a homogeneous abecedarian formula of size $s'$ computing $\mathrm{CHSYM}_{n/2,d'}(\mathbf{x})$, then there is a homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,d\cdot d'}(\mathbf{x})$ of size $s \cdot s'$.

- There is a small homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.
- There is a small homogeneous multilinear formula computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.
- Use the lower bound against homogeneous multilinear formulas for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ [HY11].

# Proof Idea for Converting Formulas into Abecedarian Ones

1. Let $\mathcal{F}$ be a formula computing an abecedarian polynomial.

# Proof Idea for Converting Formulas into Abecedarian Ones

1. Let $\mathcal{F}$ be a formula computing an abecedarian polynomial.
2. Convert $\mathcal{F}$ into an abecedarian circuit $\mathcal{C}$.

# Proof Idea for Converting Formulas into Abecedarian Ones

1. Let $\mathcal{F}$ be a formula computing an abecedarian polynomial.
2. Convert $\mathcal{F}$ into an abecedarian circuit $\mathcal{C}$.
3. Unravel $\mathcal{C}$ to get a syntactiaclly abecedarian formula $\mathcal{F}'$ computing the same polynomial.

# Proof Idea for Converting Formulas into Abecedarian Ones

1. Let $\mathcal{F}$ be a formula computing an abecedarian polynomial.
2. Convert $\mathcal{F}$ into an abecedarian circuit $\mathcal{C}$.
3. Unravel $\mathcal{C}$ to get a syntactiaclly abecedarian formula $\mathcal{F}'$ computing the same polynomial.

**Note**: The last step uses ideas similar to those used by Raz to *multilinearise* formulas. This is why the transformation is efficient only when the number of buckets in the partition is small.

# Back to Nisan's Question

- Can we prove an $n^{\Omega(\log d)}$ lower bound against homogeneous formulas?

# Back to Nisan's Question

- Can we prove an $n^{\Omega(\log d)}$ lower bound against homogeneous formulas?
- Can we prove a super-polynomial lower bound against homogeneous formulas for a polynomial of degree $\log n$?

# Back to Nisan's Question

- Can we prove an $n^{\Omega(\log d)}$ lower bound against homogeneous formulas?
- Can we prove a super-polynomial lower bound against homogeneous formulas for a polynomial of degree $\log n$?
- Can we prove a super-polynomial lower bound against abecedarian formulas for a polynomial when the partition size is $O(\log n)$?

# Back to Nisan's Question

- Can we prove an $n^{\Omega(\log d)}$ lower bound against homogeneous formulas?
- Can we prove a super-polynomial lower bound against homogeneous formulas for a polynomial of degree $\log n$?
- Can we prove a super-polynomial lower bound against abecedarian formulas for a polynomial when the partition size is $O(\log n)$?
- Can ideas from [Raz] or [DMPY] be modified to work for the non-commutative setting?

**Thank you!**