Separating ABPs and Structured Formulas in the Non-Commutative Setting

Prerona Chatterjee

Tata Institute of Fundamental Research, Mumbai

March 19, 2021

The Non-Commutative Setting

$$f(x, y) = (x + y) \times (x + y)$$

The Non-Commutative Setting

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

The Non-Commutative Setting

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

Natural restriction with long line of work, beginning with [**Hya ’77**], [**Nis ’91**].

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

Natural restriction with long line of work, beginning with [**Hya ’77**], [**Nis ’91**].

Since it is a restricted setting, it is possibly easier to prove lower bounds here.

The Non-Commutative Setting

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

Natural restriction with long line of work, beginning with [**Hya '77**], [**Nis '91**].

Since it is a restricted setting, it is possibly easier to prove lower bounds here.

- $\mathrm{Det}_n(\mathbf{x})$ has small ($\mathrm{poly}(n)$) complexity in the commutative setting but is expected to have very large ($2^{\Omega(n)}$) complexity in this setting (**AS '18**).

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

Natural restriction with long line of work, beginning with [**Hya '77**], [**Nis '91**].

Since it is a restricted setting, it is possibly easier to prove lower bounds here.

- $\mathrm{Det}_n(\mathbf{x})$ has small (poly(n)) complexity in the commutative setting but is expected to have very large ($2^{\Omega(n)}$) complexity in this setting (**AS '18**).
- Exponential lower bounds known for certain models via exact characterisation (**Nis '91**). Best lower bound for corresponding models in the commutative setting is quadratic (**Kal '85, CKSV '20**).

The Non-Commutative Setting

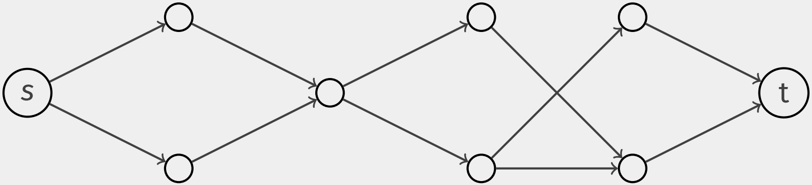
$$f(x,y) = (x+y) \times (x+y) = x^2 + xy + yx + y^2$$

Natural restriction with long line of work, beginning with [**Hya ’77**], [**Nis ’91**].

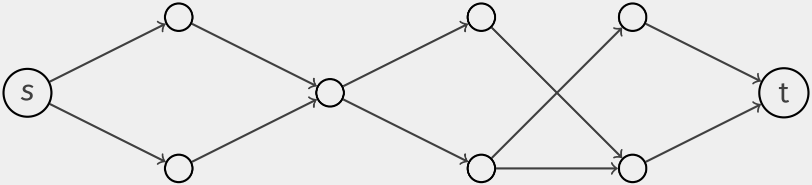
Since it is a restricted setting, it is possibly easier to prove lower bounds here.

- $\mathrm{Det}_n(\mathbf{x})$ has small ($\mathrm{poly}(n)$) complexity in the commutative setting but is expected to have very large ($2^{\Omega(n)}$) complexity in this setting (**AS ’18**).
- Exponential lower bounds known for certain models via exact characterisation (**Nis ’91**). Best lower bound for corresponding models in the commutative setting is quadratic (**Kal ’85, CKSV ’20**).
- Hardness Amplification is known (**CILM ’18**).

Algebraic Branching Programs

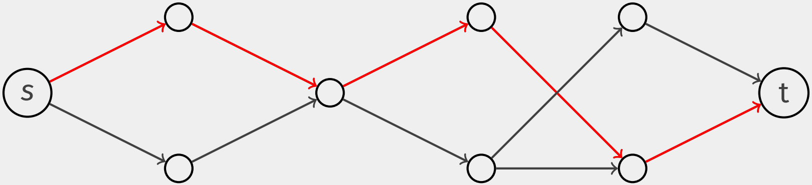


Algebraic Branching Programs

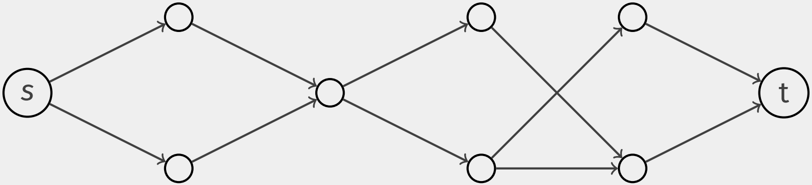


- **Label on each edge:** A homogeneous linear form in $\{x_1, x_2, \ldots, x_n\}$

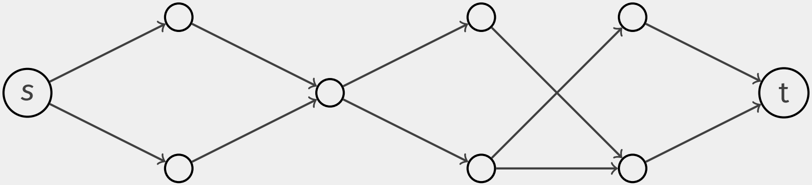
Algebraic Branching Programs



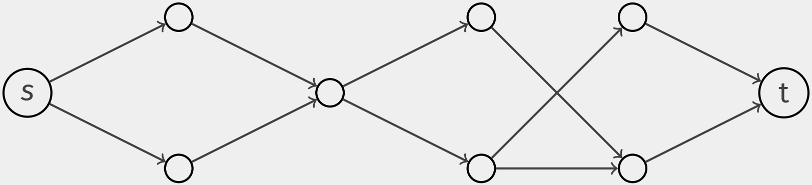
- **Label on each edge:** A homogeneous linear form in $\{x_1, x_2, \ldots, x_n\}$
- **Weight of path p = wt(p):** Product of the edge labels on p



- **Label on each edge:** A homogeneous linear form in $\{x_1, x_2, \ldots, x_n\}$
- **Weight of path** p **=** $\mathrm{wt}(p)$**:** Product of the edge labels on p
- **Polynomial computed by the ABP:** $\sum_p \mathrm{wt}(p)$

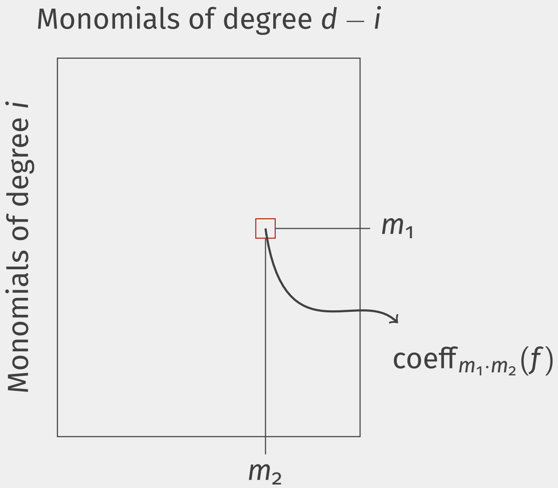


- **Label on each edge:** A homogeneous linear form in $\{x_1, x_2, \ldots, x_n\}$
- **Weight of path p = wt(p):** Product of the edge labels on p
- **Polynomial computed by the ABP:** $\sum_p \mathrm{wt}(p)$
- **Size of the ABP:** Number of vertices in the ABP



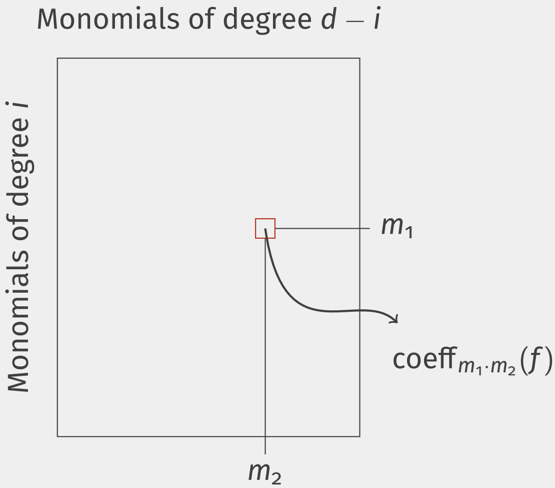
- **Label on each edge:** A homogeneous linear form in $\{x_1, x_2, \ldots, x_n\}$
- **Weight of path p = wt(p):** Product of the edge labels on p
- **Polynomial computed by the ABP:** $\sum_p \text{wt}(p)$
- **Size of the ABP:** Number of vertices in the ABP

For a general polynomial f of degree d, $f = \text{Hom}_0(f) + \text{Hom}_1(f) + \cdots + \text{Hom}_d(f)$.



f is a polynomial of degree d.

For every $1 \leq i \leq d$, consider the matrix $M_f(i)$ described alongside.

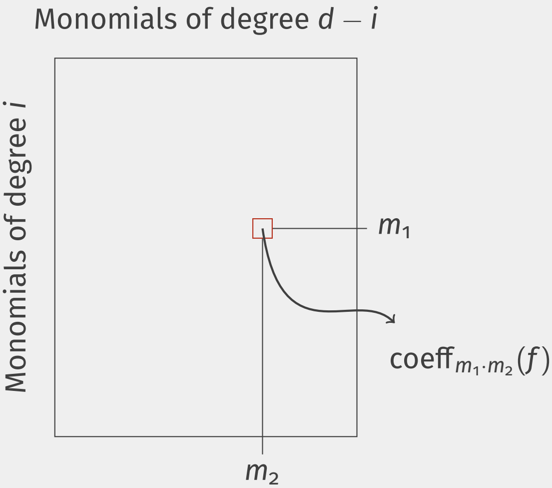


f is a polynomial of degree d.

For every $1 \leq i \leq d$, consider the matrix $M_f(i)$ described alongside.

Nisan (1991): For every $1 \leq i \leq d$, The number of vertices in the i-th layer of the smallest ABP computing f is equal to the rank of $M_f(i)$.

Nisan's Characterisation



f is a polynomial of degree d.

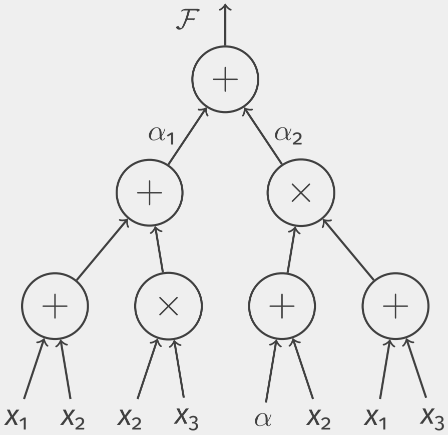
For every $1 \leq i \leq d$, consider the matrix $M_f(i)$ described alongside.

Nisan (1991): For every $1 \leq i \leq d$, The number of vertices in the i-th layer of the smallest ABP computing f is equal to the rank of $M_f(i)$.

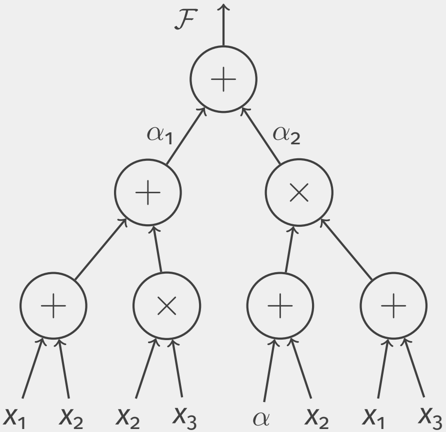
If $\mathcal{A}$ is the smallest ABP computing f,

$$\text{size}(\mathcal{A}) = \sum_{i=1}^{d} \text{rank}(M_f(i)).$$

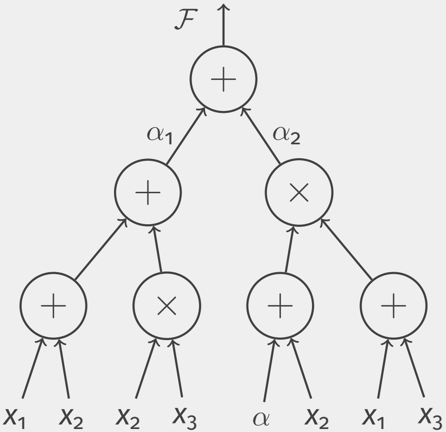
Algebraic Formulas



Algebraic Formulas



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

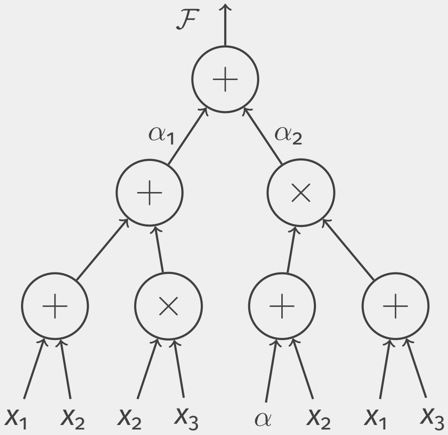


$$\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{VBP}_{\mathsf{nc}}$$

Super-polynomial ABP lower bounds

$\Downarrow$

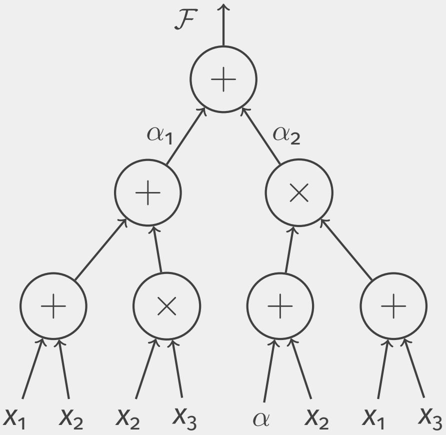
super-polynomial formula lower bounds.



$\mathsf{VF_{nc}} \subseteq \mathsf{VBP_{nc}}$

Super-polynomial ABP lower bounds
$\Downarrow$
super-polynomial formula lower bounds.

Exponential Lower Bound [Nisan]:
Any ABP (or formula) computing $\mathrm{Det}_n(\mathbf{x})$ has size at least $2^{\Omega(n)}$.



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

Super-polynomial ABP lower bounds
$\Downarrow$
super-polynomial formula lower bounds.

Exponential Lower Bound [Nisan]:
Any ABP (or formula) computing $\mathrm{Det}_n(\mathbf{x})$ has size at least $2^{\Omega(n)}$.

Question: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

Generalises the notion of *ordered polynomials* (defined in **[HWY11]**).

Generalises the notion of *ordered polynomials* (defined in **[HWY11]**).

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

Generalises the notion of *ordered polynomials* (defined in **[HWY11]**).

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

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

Generalises the notion of *ordered polynomials* (defined in **[HWY11]**).

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

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* (defined in **[HWY11]**).

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

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

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

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

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

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 $X_i = \{x_i\}$	$\mathrm{CHSYM}_{n,d}(\mathbf{x})$, $\mathrm{ESYM}_{n,d}(\mathbf{x})$

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

$$f(\mathbf{x}) \xrightarrow[\text{using some fixed } \sigma \in S_n]{\text{Order the monomials}} f^{(nc)}(\mathbf{x})$$

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 $X_i = \{x_i\}$	$\mathrm{CHSYM}_{n,d}(\mathbf{x})$, $\mathrm{ESYM}_{n,d}(\mathbf{x})$ Non-Commutative version of any $f \in \mathbb{F}[x_1, \ldots, x_n]$

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

Notation: Consider $1 \leq a \leq b \leq m+1$ where m: size of the bucketing system.

Notation: Consider $1 \leq a \leq b \leq m+1$ where m: size of the bucketing system.

- For any $a \in [m+1]$, $f[a,a)$ is the constant term in f.

Notation: Consider $1 \leq a \leq b \leq m+1$ where m: size of the bucketing system.

- For any $a \in [m+1]$, $f[a,a)$ is the constant term in f.
- For $1 \leq a < b \leq m+1$, $f[a,b)$ contains only those monomials of f in which the first variable is from bucket X_a; last variable is from bucket X_a or X_{a+1} or $\ldots$ or X_{b-1}.

Notation: Consider $1 \leq a \leq b \leq m+1$ where m: size of the bucketing system.

- For any $a \in [m+1]$, $f[a,a)$ is the constant term in f.
- For $1 \leq a < b \leq m+1$, $f[a,b)$ contains only those monomials of f in which the first variable is from bucket X_a; last variable is from bucket X_a or X_{a+1} or $\ldots$ or X_{b-1}.

Abecedarian Formulas

Every vertex v is labelled by a tuple (a,b)

Notation: Consider $1 \leq a \leq b \leq m+1$ where m: size of the bucketing system.

- For any $a \in [m+1]$, $f[a, a)$ is the constant term in f.
- For $1 \leq a < b \leq m+1$, $f[a, b)$ contains only those monomials of f in which the first variable is from bucket X_a; last variable is from bucket X_a or X_{a+1} or ... or X_{b-1}.

Abecedarian Formulas

Every vertex v is labelled by a tuple (a, b)

f_v is the polynomial at $v \implies f_v = f_v[a, b)$

Notation: Consider $1 \leq a \leq b \leq m+1$ where m: size of the bucketing system.

- For any $a \in [m+1]$, $f[a,a)$ is the constant term in f.
- For $1 \leq a < b \leq m+1$, $f[a,b)$ contains only those monomials of f in which the first variable is from bucket X_a; last variable is from bucket X_a or X_{a+1} or $\ldots$ or X_{b-1}.

Abecedarian Formulas

Every vertex v is labelled by a tuple (a,b)

f_v is the polynomial at $v \implies f_v = f_v[a,b)$

Abecedarian ABPs

Every vertex is labelled by a bucket index

Notation: Consider $1 \le a \le b \le m+1$ where m: size of the bucketing system.

- For any $a \in [m+1]$, $f[a,a)$ is the constant term in f.
- For $1 \le a < b \le m+1$, $f[a,b)$ contains only those monomials of f in which the first variable is from bucket X_a; last variable is from bucket X_a or X_{a+1} or $\ldots$ or X_{b-1}.

Abecedarian Formulas

Every vertex v is labelled by a tuple (a,b)

f_v is the polynomial at $v \implies f_v = f_v[a,b)$

Abecedarian ABPs

Every vertex is labelled by a bucket index

f is the polynomial computed between (u,a) and $(v,b) \implies f = f[a,b+1)$.

What was known and what we show

[LLS '19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathbf{x})$ must have size $n^{\Omega(\log n)}$.

What was known and what we show

[LLS '19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathbf{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

1. There is an explicit n^2-variate, degree d polynomial $f_{n,d}(\mathbf{x})$ which is abecedarian with respect to a bucketing system of size n such that

[LLS '19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathbf{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

1. There is an explicit n^2-variate, degree d polynomial $f_{n,d}(\mathbf{x})$ which is abecedarian with respect to a bucketing system of size n such that
   - $f_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of polynomial size;

[LLS '19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathbf{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

1. There is an explicit n^2-variate, degree d polynomial $f_{n,d}(\mathbf{x})$ which is abecedarian with respect to a bucketing system of size n such that
   - $f_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of polynomial size;
   - any abecedarian formula computing $f_{n,\log n}(\mathbf{x})$ must have size that is super-polynomial in n.

[LLS '19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathbf{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

1. There is an explicit n^2-variate, degree d polynomial $f_{n,d}(\mathbf{x})$ which is abecedarian with respect to a bucketing system of size n such that
   - $f_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of polynomial size;
   - any abecedarian formula computing $f_{n,\log n}(\mathbf{x})$ must have size that is super-polynomial in n.
2. Let f be an n-variate abecedarian polynomial with respect to a bucketing system of size $O(\log n)$ that can be computed by an ABP of size $\mathrm{poly}(n)$.

[LLS '19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathbf{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

1. There is an explicit n^2-variate, degree d polynomial $f_{n,d}(\mathbf{x})$ which is abecedarian with respect to a bucketing system of size n such that
   - $f_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of polynomial size;
   - any abecedarian formula computing $f_{n,\log n}(\mathbf{x})$ must have size that is super-polynomial in n.
2. Let f be an n-variate abecedarian polynomial with respect to a bucketing system of size $O(\log n)$ that can be computed by an ABP of size $\mathrm{poly}(n)$. A super-polynomial lower bound against abecedarian formulas for f would imply that $\mathrm{VF}_{\mathrm{nc}} \neq \mathrm{VBP}_{\mathrm{nc}}$.

Two natural questions that arise at this point:

Two natural questions that arise at this point:

1. Can any formula computing an abecedarian polynomial be converted to an abecedarian formula without much blow-up in size, irrespective of the size of the bucketing system?

Possible New Approaches to Solving the General Question

Two natural questions that arise at this point:

1. Can any formula computing an abecedarian polynomial be converted to an abecedarian formula without much blow-up in size, irrespective of the size of the bucketing system?
2. Is there a polynomial f which is abecedarian with respect to a bucketing system that has small size such that f witnesses a separation between abecedarian formulas and ABPs?

Two natural questions that arise at this point:

1. Can any formula computing an abecedarian polynomial be converted to an abecedarian formula without much blow-up in size, irrespective of the size of the bucketing system?
2. Is there a polynomial f which is abecedarian with respect to a bucketing system that has small size such that f witnesses a separation between abecedarian formulas and ABPs?

A positive answer to either of these questions would imply that $\mathsf{VBP}_{nc} \neq \mathsf{VF}_{nc}$.

The Explicit Statement

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

The Explicit Statement

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

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

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 $X_i = \left\{x_{ij} \ : \ 1 \le j \le n\right\}$.
- linked_$\mathrm{CHSYM}_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of size $O(nd)$.

The Explicit Statement

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

- Abecedarian with respect to $\{X_i \,:\, 1 \le i \le n\}$ where $X_i = \left\{x_{ij} \,:\, 1 \le j \le n\right\}$.
- $\text{linked\_}\mathrm{CHSYM}_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of size $O(nd)$.
- Any abecedarian formula computing $\text{linked\_}\mathrm{CHSYM}_{n,\log n}(\mathbf{x})$ has size $n^{\Omega(\log\log n)}$.

The Explicit Statement

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

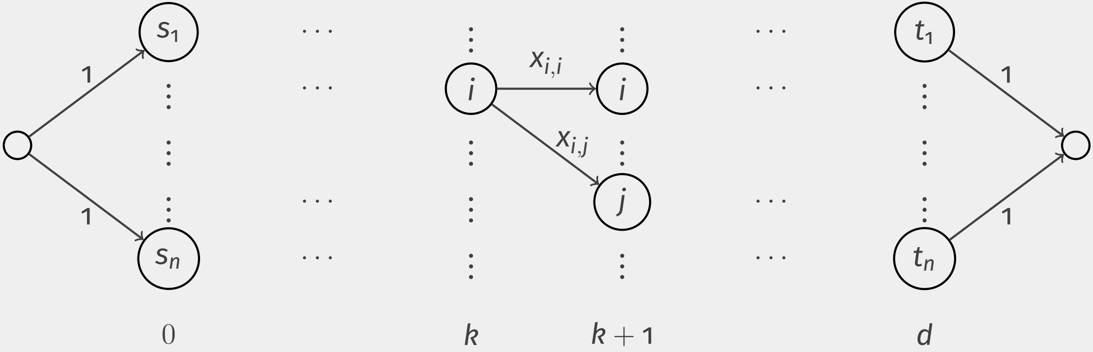
- Abecedarian with respect to $\{X_i \ : \ 1 \le i \le n\}$ where $X_i = \{x_{ij} \ : \ 1 \le j \le n\}$.
- linked_$\mathrm{CHSYM}_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of size $O(nd)$.
- Any abecedarian formula computing linked_$\mathrm{CHSYM}_{n,\log n}(\mathbf{x})$ has size $n^{\Omega(\log \log n)}$.
- A super-polynomial lower bound against abecedarian formulas for linked_$\mathrm{CHSYM}_{\log n,n}(\mathbf{x})$ would imply that $\mathrm{VF}_{\mathrm{nc}} \neq \mathrm{VBP}_{\mathrm{nc}}$.

The Structured ABP Upper Bound

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

The Structured ABP Upper Bound

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



Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log \log n})$ computing $h_{n/2, \log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\mathrm{poly}(s)$.

Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log \log n})$ computing $h_{n/2, \log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\text{poly}(s)$.
2. Using $\mathcal{F}'_1$ construct a more structured formula $\mathcal{F}'_2$ computing $h_{n/2, \log n}(\mathbf{x})$, of size $s'_2 = \text{poly}(s)$.

Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log \log n})$ computing $h_{n/2, \log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\mathrm{poly}(s)$.
2. Using $\mathcal{F}'_1$ construct a more structured formula $\mathcal{F}'_2$ computing $h_{n/2, \log n}(\mathbf{x})$, of size $s'_2 = \mathrm{poly}(s)$.
3. Construct a homogeneous, abecedarian formula $\mathcal{F}$ of size $s = \mathrm{poly}(n)$, that computes $\mathrm{CHSYM}_{n/2, \log n}(\mathbf{x})$.

Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log\log n})$ computing $h_{n/2,\log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\mathrm{poly}(s)$.
2. Using $\mathcal{F}'_1$ construct a more structured formula $\mathcal{F}'_2$ computing $h_{n/2,\log n}(\mathbf{x})$, of size $s'_2 = \mathrm{poly}(s)$.
3. Construct a homogeneous, abecedarian formula $\mathcal{F}$ of size $s = \mathrm{poly}(n)$, that computes $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.
4. Show that there is a homogeneous abecedarian formula of size $(s \cdot s'_2)$ computing $\mathrm{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$.

Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log\log n})$ computing $h_{n/2,\log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\mathrm{poly}(s)$.
2. Using $\mathcal{F}'_1$ construct a more structured formula $\mathcal{F}'_2$ computing $h_{n/2,\log n}(\mathbf{x})$, of size $s'_2 = \mathrm{poly}(s)$.
3. Construct a homogeneous, abecedarian formula $\mathcal{F}$ of size $s = \mathrm{poly}(n)$, that computes $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.
4. Show that there is a homogeneous abecedarian formula of size $(s \cdot s'_2)$ computing $\mathrm{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$.
5. Reuse Step 4 repeatedly at most $O({}^{\log n}/_{\log\log n})$ times to obtain a homogeneous abecedarian formula $\mathcal{F}_1$ of size $O(n^{c \cdot \varepsilon \log n})$, that computes $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.

Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log \log n})$ computing $h_{n/2, \log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\mathrm{poly}(s)$.
2. Using $\mathcal{F}'_1$ construct a more structured formula $\mathcal{F}'_2$ computing $h_{n/2,\log n}(\mathbf{x})$, of size $s'_2 = \mathrm{poly}(s)$.
3. Construct a homogeneous, abecedarian formula $\mathcal{F}$ of size $s = \mathrm{poly}(n)$, that computes $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.
4. Show that there is a homogeneous abecedarian formula of size $(s \cdot s'_2)$ computing $\mathrm{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$.
5. Reuse Step 4 repeatedly at most $O(\log n / \log \log n)$ times to obtain a homogeneous abecedarian formula $\mathcal{F}_1$ of size $O(n^{c \cdot \varepsilon \log n})$, that computes $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.
6. Using $\mathcal{F}_1$ construct a homogeneous multilinear formula of size $O(n^{c \cdot \varepsilon \log n})$ computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.

Proof Idea of the Structured Formula Lower Bound

Let $\mathcal{F}'$ be an abecedarian formula of size $s = O(n^{\varepsilon \log\log n})$ computing $h_{n/2,\log n}(\mathbf{x})$.

1. Homogenise to get $\mathcal{F}'_1$ of size $\mathrm{poly}(s)$.
2. Using $\mathcal{F}'_1$ construct a more structured formula $\mathcal{F}'_2$ computing $h_{n/2,\log n}(\mathbf{x})$, of size $s'_2 = \mathrm{poly}(s)$.
3. Construct a homogeneous, abecedarian formula $\mathcal{F}$ of size $s = \mathrm{poly}(n)$, that computes $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.
4. Show that there is a homogeneous abecedarian formula of size $(s \cdot s'_2)$ computing $\mathrm{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$.
5. Reuse Step 4 repeatedly at most $O(\log n/\log\log n)$ times to obtain a homogeneous abecedarian formula $\mathcal{F}_1$ of size $O(n^{c\cdot\varepsilon \log n})$, that computes $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$.
6. Using $\mathcal{F}_1$ construct a homogeneous multilinear formula of size $O(n^{c\cdot\varepsilon \log n})$ computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.
7. Choose ε such that Step 6 contradicts the known lower bound against this model for $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$.

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Crucial Observations

- When degree is small, structural changes can be made on formulas.

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Crucial Observations

- When degree is small, structural changes can be made on formulas.
- $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ is structured enough to allow *degree amplification* (Step 4).

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Crucial Observations

- When degree is small, structural changes can be made on formulas.
- $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ is structured enough to allow *degree amplification* (Step 4).
- All the other steps are to facilitate the degree amplification.

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Crucial Observations

- When degree is small, structural changes can be made on formulas.
- $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ is structured enough to allow *degree amplification* (Step 4).
- All the other steps are to facilitate the degree amplification.

A little about Step 1

- Brent's Depth Reduction proof works in the non-commutative setting as well.

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Crucial Observations

- When degree is small, structural changes can be made on formulas.
- $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ is structured enough to allow *degree amplification* (Step 4).
- All the other steps are to facilitate the degree amplification.

A little about Step 1

- Brent's Depth Reduction proof works in the non-commutative setting as well.
- This allows Raz's Homogenisation proof to go through in this setting as well.

Highlighting the Key Steps

Want to use the lower bound in [HY11] against homogeneous multilinear formulas.

Crucial Observations

- When degree is small, structural changes can be made on formulas.
- $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ is structured enough to allow *degree amplification* (Step 4).
- All the other steps are to facilitate the degree amplification.

A little about Step 1

- Brent's Depth Reduction proof works in the non-commutative setting as well.
- This allows Raz's Homogenisation proof to go through in this setting as well.
- It also answers a question by Nisan.

  $D(f)$: Depth Complexity; $F(f)$: Formula Complexity

  $$\text{Is } D(f) \leq O(\log F(f))?$$

We answer the question in the positive.

Proof Idea for Converting Formulas into Abecedarian Ones

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

Proof Idea for Converting Formulas into Abecedarian Ones

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

1. Convert the given formula $\mathcal{F}$ into an abecedarian circuit $\mathcal{C}$.

Proof Idea for Converting Formulas into Abecedarian Ones

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

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

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

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

Step 1

- Make m^2 copies in $\mathcal{C}$ of each vertex in $\mathcal{F}$: $\{(a, b)\}_{a,b \in [m+1]}$.

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

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

Step 1

- Make m^2 copies in $\mathcal{C}$ of each vertex in $\mathcal{F}$: $\{(a,b)\}_{a,b\in[m+1]}$.
- f_v is the polynomial at v

  $\Downarrow$

  the polynomial at $(v,(a,b))$ is $f_v[a,b]$.

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

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

Step 1

- Make m^2 copies in $\mathcal{C}$ of each vertex in $\mathcal{F}$: $\{(a, b)\}_{a,b \in [m+1]}$.
- f_v is the polynomial at v
  $\Downarrow$
  the polynomial at $(v, (a, b))$ is $f_v[a, b]$.

Step 2

- Convert $\mathcal{C}$ into a formula $\mathcal{F}'$ by recomputing vertices every time it is reused.

Proof Idea for Converting Formulas into Abecedarian Ones

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

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

Step 1

- Make m^2 copies in $\mathcal{C}$ of each vertex in $\mathcal{F}$: $\{(a,b)\}_{a,b\in[m+1]}$.
- f_v is the polynomial at v $$\Downarrow$$ the polynomial at $(v,(a,b))$ is $f_v[a,b]$.

Step 2

- Convert $\mathcal{C}$ into a formula $\mathcal{F}'$ by recomputing vertices every time it is reused.
- Analyse (similar to Raz) the number of distinct paths from any vertex in $\mathcal{C}$ to the root.

Proof Idea for Converting Formulas into Abecedarian Ones

Let $\mathcal{F}$ be a formula of size s computing a polynomial that is abecedarian with respect to a bucketing system of size m.

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

Step 1

- Make m^2 copies in $\mathcal{C}$ of each vertex in $\mathcal{F}$: $\{(a, b)\}_{a,b \in [m+1]}$.
- f_v is the polynomial at v

  $\Downarrow$

  the polynomial at $(v, (a, b))$ is $f_v[a, b]$.

Step 2

- Convert $\mathcal{C}$ into a formula $\mathcal{F}'$ by recomputing vertices every time it is reused.
- Analyse (similar to Raz) the number of distinct paths from any vertex in $\mathcal{C}$ to the root.
- When $m = O(\log s)$, the size of $\mathcal{F}'$ remains $\text{poly}(s)$.

Other Observations

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.

Other Observations

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
2. $\mathsf{VBP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.

Other Observations

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
2. $\mathsf{VBP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
3. $\mathsf{abcd} - \mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.

Other Observations

1. $\mathsf{VP_{nc}} = \mathsf{abcd} - \mathsf{VP_{nc}}$.
2. $\mathsf{VBP_{nc}} = \mathsf{abcd} - \mathsf{VBP_{nc}}$.
3. $\mathsf{abcd} - \mathsf{VF_{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP_{nc}}$.
4. $\mathsf{abcd} - \mathsf{VBP_{nc}} \subsetneq \mathsf{abcd} - \mathsf{VP_{nc}}$.

1. $\mathrm{VP_{nc}} = \mathrm{abcd} - \mathrm{VP_{nc}}$.
2. $\mathrm{VBP_{nc}} = \mathrm{abcd} - \mathrm{VBP_{nc}}$.
3. $\mathrm{abcd} - \mathrm{VF_{nc}} \subseteq \mathrm{abcd} - \mathrm{VBP_{nc}}$.
4. $\mathrm{abcd} - \mathrm{VBP_{nc}} \subsetneq \mathrm{abcd} - \mathrm{VP_{nc}}$.
5. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s

   $$\Downarrow$$

   there is an abcd-formula computing f of size $O(s^{\log d})$.

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
2. $\mathsf{VBP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
3. $\mathsf{abcd} - \mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
4. $\mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
5. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s

   $\Downarrow$

   there is an abcd-formula computing f of size $O(s^{\log d})$.

Corollaries of the Homogenisation Observation

$2^{\omega(n)}$ lower bound against homogeneous formulas for $\mathrm{Det}_n(\mathbf{x})$

Other Observations

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd}-\mathsf{VP}_{\mathsf{nc}}$.
2. $\mathsf{VBP}_{\mathsf{nc}} = \mathsf{abcd}-\mathsf{VBP}_{\mathsf{nc}}$.
3. $\mathsf{abcd}-\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd}-\mathsf{VBP}_{\mathsf{nc}}$.
4. $\mathsf{abcd}-\mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd}-\mathsf{VP}_{\mathsf{nc}}$.
5. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s

   $\Downarrow$

   there is an abcd-formula computing f of size $O(s^{\log d})$.

Corollaries of the Homogenisation Observation

$2^{\omega(n)}$ lower bound against homogeneous formulas for $\mathrm{Det}_n(\mathbf{x})$

or

$n^{\omega(1)}$ lower bound against homogeneous formulas for $\mathrm{IMM}_{n,\log n}(\mathbf{x})$

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
2. $\mathsf{VBP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
3. $\mathsf{abcd} - \mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
4. $\mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
5. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s

   $\Downarrow$

   there is an abcd-formula computing f of size $O(s^{\log d})$.

Corollaries of the Homogenisation Observation

$2^{\omega(n)}$ lower bound against homogeneous formulas for $\mathrm{Det}_n(\mathbf{x})$

or

$n^{\omega(1)}$ lower bound against homogeneous formulas for $\mathrm{IMM}_{n,\log n}(\mathbf{x})$

or

$n^{\omega(1)}$ lower bound against homogeneous formulas for linked_$\mathrm{CHSYM}_{n,\log n}(\mathbf{x})$

1. $\mathsf{VP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
2. $\mathsf{VBP}_{\mathsf{nc}} = \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
3. $\mathsf{abcd} - \mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}}$.
4. $\mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
5. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s

   $\Downarrow$

   there is an abcd-formula computing f of size $O(s^{\log d})$.

Corollaries of the Homogenisation Observation

$2^{\omega(n)}$ lower bound against homogeneous formulas for $\mathrm{Det}_n(\mathbf{x})$

or

$n^{\omega(1)}$ lower bound against homogeneous formulas for $\mathrm{IMM}_{n,\log n}(\mathbf{x})$

or

$n^{\omega(1)}$ lower bound against homogeneous formulas for linked_$\mathrm{CHSYM}_{n,\log n}(\mathbf{x})$

$\Longrightarrow \quad \mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VBP}_{\mathsf{nc}}$

Thank you!