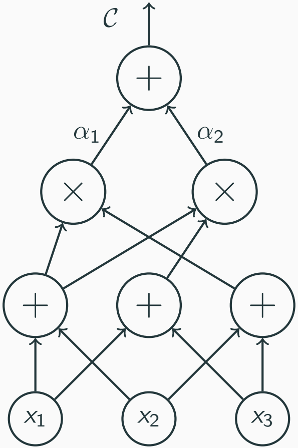
Separating ABPs and some Structured Formulas in the Non-Commutative Setting

Prerona Chatterjee

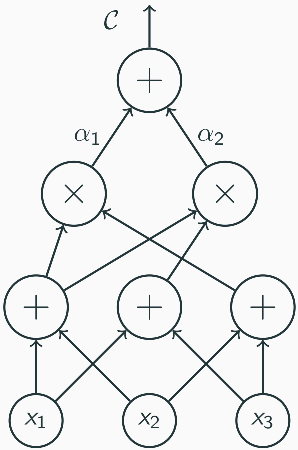
Tata Institute of Fundamental Research, Mumbai

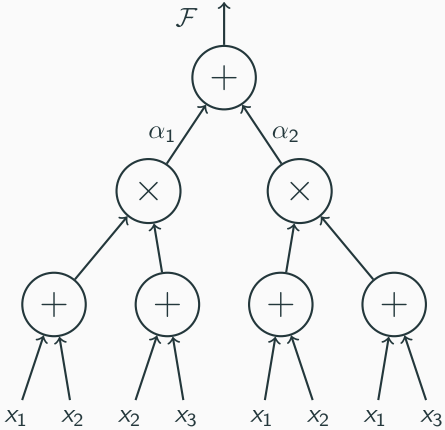
July 20, 2021

Algebraic Circuits and Formulas



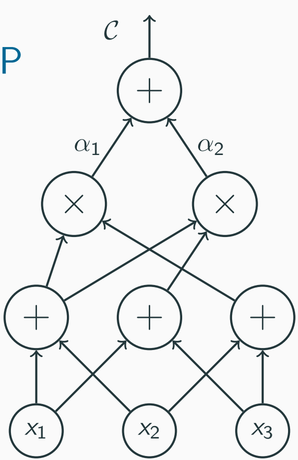
Algebraic Circuits and Formulas

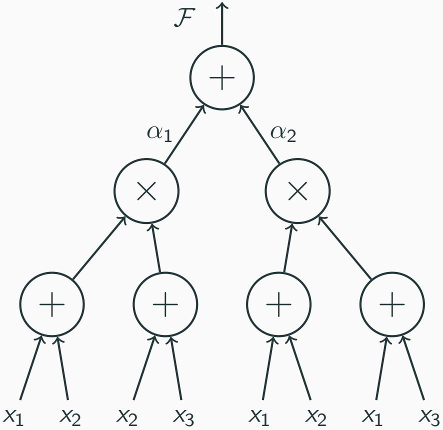




Algebraic Circuits and Formulas

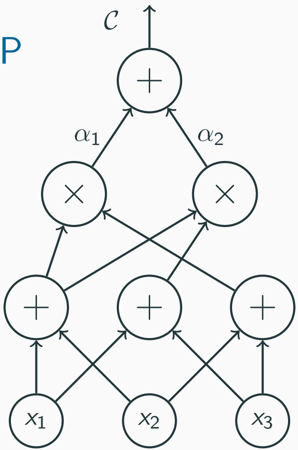
VP

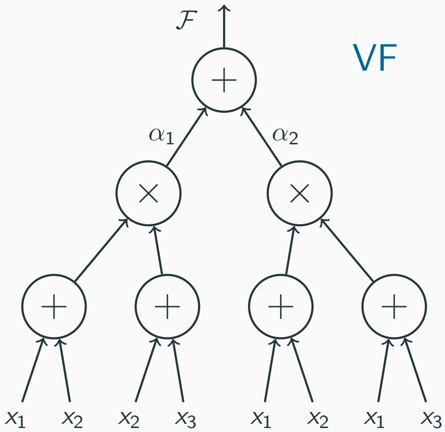




Algebraic Circuits and Formulas

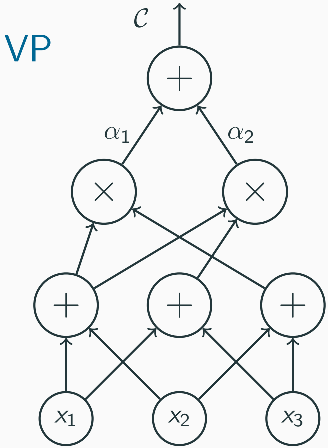
VP



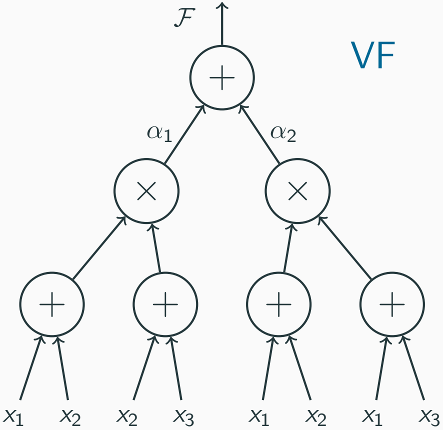


VF

Algebraic Circuits and Formulas



$\boxed{\mathrm{VP} \supseteq \mathrm{VF}}$



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 [Str73, BS83]: $\Omega(n \log d)$ 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 [Str73, BS83]: $\Omega(n \log d)$ for an n-variate, degree d polynomial.

Formulas [Nis91]: $2^{\Omega(n)}$ 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 [Str73, BS83]: $\Omega(n \log d)$ for an n-variate, degree d polynomial.

Formulas [Nis91]: $2^{\Omega(n)}$ 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 [Str73, BS83]: $\Omega(n \log d)$ for an n-variate, degree d polynomial.

Formulas [Nis91]: $2^{\Omega(n)}$ 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 [Str73, BS83]: $\Omega(n \log d)$ for an n-variate, degree d polynomial.

Formulas [Nis91]: $2^{\Omega(n)}$ 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.

$\mathrm{VBP}_{\mathrm{nc}}$

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 [Str73, BS83]: $\Omega(n \log d)$ for an n-variate, degree d polynomial.

Formulas [Nis91]: $2^{\Omega(n)}$ 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 [Str73, BS83]: $\Omega(n \log d)$ for an n-variate, degree d polynomial.

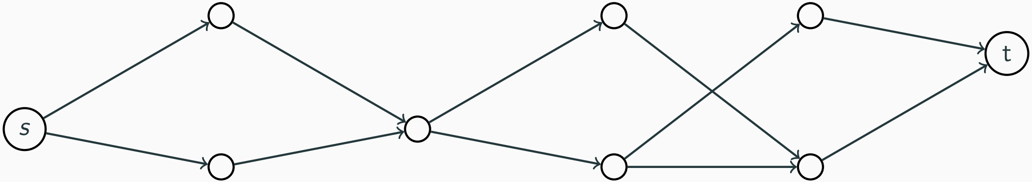
Formulas [Nis91]: $2^{\Omega(n)}$ for a 2-variate, degree n polynomial in $\mathsf{VP}_{\mathsf{nc}}$. So, $\mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VP}_{\mathsf{nc}}$.

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

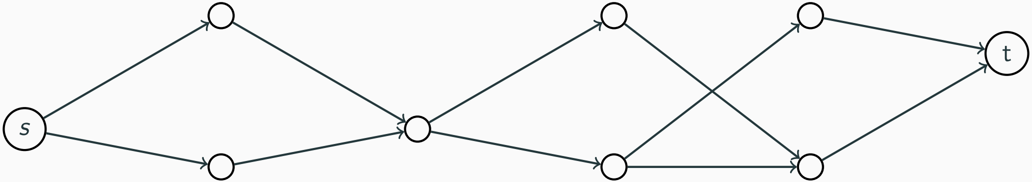
$$\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{VBP}_{\mathsf{nc}} \subseteq \mathsf{VP}_{\mathsf{nc}}$$

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

Algebraic Branching Programs

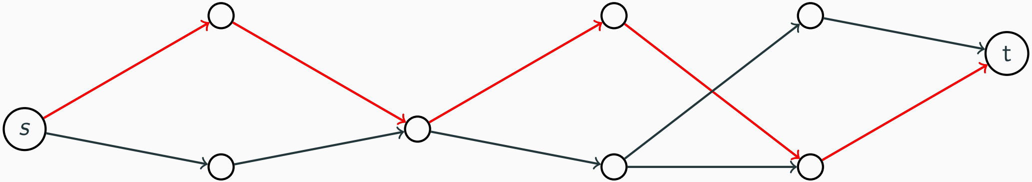


Algebraic Branching Programs



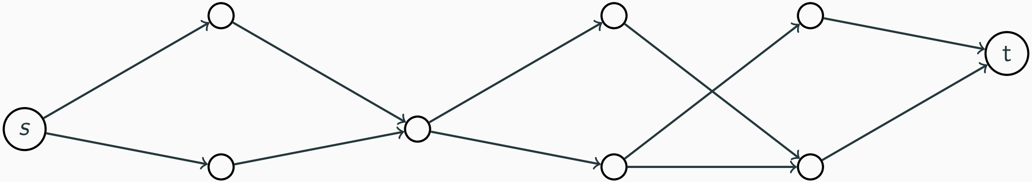
- Label on each edge: Homogeneous linear forms in $\{x_1, x_2, \ldots, x_n\}$

Algebraic Branching Programs



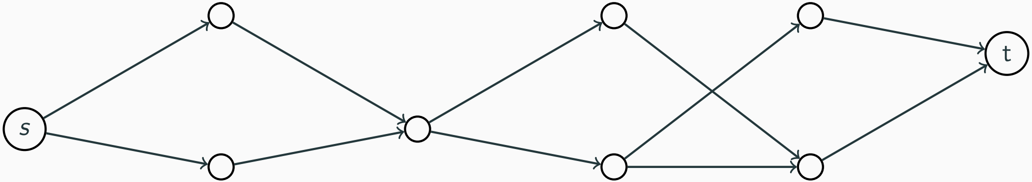
- Label on each edge: Homogeneous linear forms in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on p

Algebraic Branching Programs



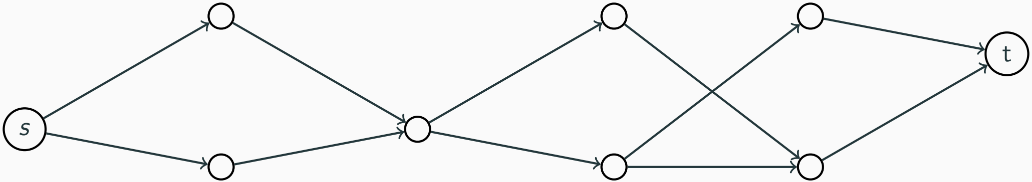
- Label on each edge: Homogeneous linear forms in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on p
- Polynomial computed by the ABP: $\sum_p \mathrm{wt}(p)$

Algebraic Branching Programs



- Label on each edge: Homogeneous linear forms 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: $\sum_p \mathrm{wt}(p)$
- Size of the ABP: Number of vertices in the ABP

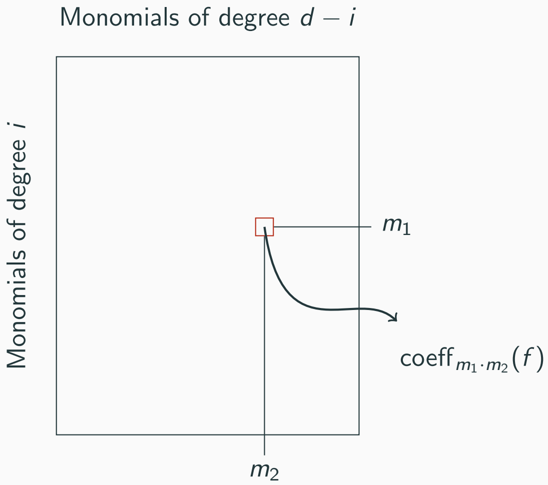
Algebraic Branching Programs



- Label on each edge: Homogeneous linear forms 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: $\sum_p \mathrm{wt}(p)$
- Size of the ABP: Number of vertices in the ABP

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

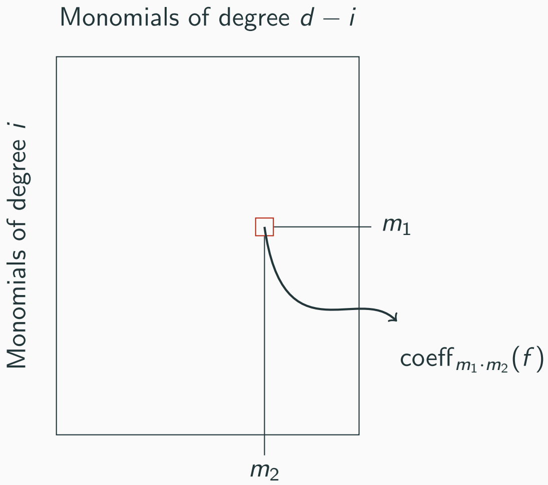
Nisan's Characterisation



f is a polynomial of degree d.

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

Nisan's Characterisation

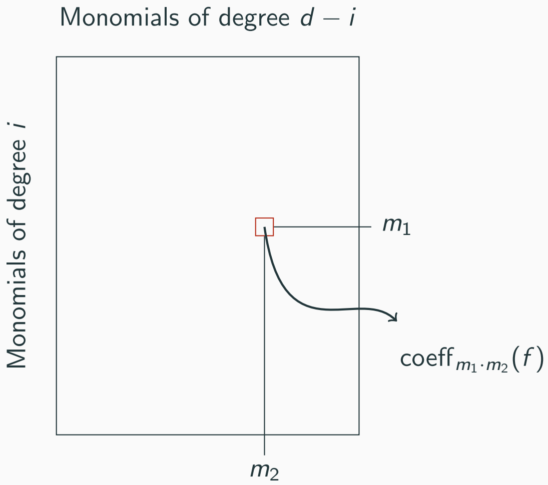


f is a polynomial of degree d.

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 $M_f(i)$.

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,

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

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

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 $\{x_1, \ldots, x_n\}$.

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

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

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

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.

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

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 $X_1^* X_2^* \cdots X_m^*$.

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}$?

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.

The ABP vs Formulas Question

The Question [Nis91]: Is $\mathrm{VF}_{\mathrm{nc}} = \mathrm{VBP}_{\mathrm{nc}}$?

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 (defined in [HWY11]).

Abecedarian Polynomials

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

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 (defined in [HWY11]).

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

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

Abecedarian Polynomials

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

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

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

Abecedarian Polynomials

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

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

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

Abecedarian Polynomials

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

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

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

Abecedarian Polynomials

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

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

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

Abecedarian Polynomials

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

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

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

Abecedarian Polynomials

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

Note:

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

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

What was known and what we show

$$\boxed{\text{Is } \mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}?}$$

What was known and what we show

$$\boxed{\text{Is } \mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}?}$$

[LLS19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathrm{x})$ must have size $n^{\Omega(\log n)}$.

What was known and what we show

$\boxed{\text{Is } \mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}\text{?}}$

[LLS19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathsf{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

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

What was known and what we show

$\boxed{\text{Is } \mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}?}$

[LLS19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathrm{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

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

What was known and what we show

$$\boxed{\text{Is } \mathrm{VF}_{\mathrm{nc}} = \mathrm{VBP}_{\mathrm{nc}}?}$$

[LLS19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathrm{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

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

What was known and what we show

$$\boxed{\text{Is } \mathsf{VF}_{\mathsf{nc}} = \mathsf{VBP}_{\mathsf{nc}}?}$$

[LLS19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathrm{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

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

What was known and what we show

$\boxed{\text{Is } \mathrm{VF}_{\mathrm{nc}} = \mathrm{VBP}_{\mathrm{nc}}?}$

[LLS19]: Any UPT formula computing $\mathrm{IMM}_{n,n}(\mathrm{x})$ must have size $n^{\Omega(\log n)}$.

Our Main Theorems:

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

Possible New Approaches to Solving the General Question

Two natural questions that arise at this point:

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 partition?

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 partition?
2. Is there a polynomial f which is abecedarian with respect to a partition of small size such that f witnesses a separation between abecedarian formulas and ABPs?

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 partition?
2. Is there a polynomial f which is abecedarian with respect to a partition of small size such that f witnesses a separation between abecedarian formulas and ABPs?

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

The Explicit Statement for the Separation

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

The Explicit Statement for the Separation

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

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

The Explicit Statement for the Separation

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

- 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 linked_$\mathrm{CHSYM}_{n,d}(\mathrm{x})$.

The Explicit Statement for the Separation

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

- 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 $\text{linked\_CHSYM}_{n,d}(\mathrm{x})$.
- Any abecedarian formula computing $\text{linked\_CHSYM}_{n,\log n}(\mathrm{x})$ has size $n^{\Omega(\log \log n)}$.

The Explicit Statement for the Separation

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

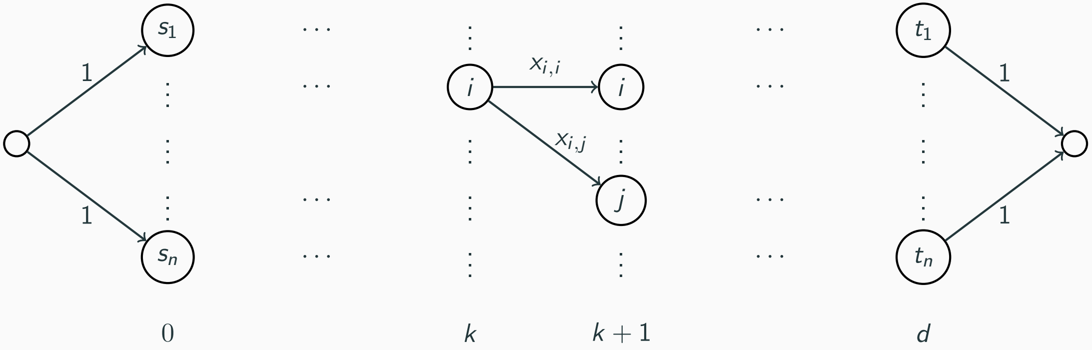
- Abecedarian with respect to $\{X_i \ : \ 1 \le i \le n\}$ where $X_i = \{x_{ij} \ : \ 1 \le j \le n\}$.
- There is an abecedarian ABP of size $O(nd)$ that computes $\text{linked\_CHSYM}_{n,d}(\mathrm{x})$.
- Any abecedarian formula computing $\text{linked\_CHSYM}_{n,\log n}(\mathrm{x})$ has size $n^{\Omega(\log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

The Abecedarian ABP Upper Bound

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

The Abecedarian ABP Upper Bound

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



Proof Idea of the Abecedarian Formula Lower Bound

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

Proof Idea of the Abecedarian Formula Lower Bound

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

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

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

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

Proof Idea of the Abecedarian Formula Lower Bound

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

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

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

Proof Idea of the Abecedarian Formula Lower Bound

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

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

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

Proof Idea of the Abecedarian Formula Lower Bound

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

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

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

Other Observations

1. Circuits and ABPs computing abecedarian polynomials can be made abecedarian.

Other Observations

1. Circuits and ABPs computing abecedarian polynomials can be made abecedarian.
2. $\mathsf{abcd}-\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd}-\mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd}-\mathsf{VP}_{\mathsf{nc}}$.

Other Observations

1. Circuits and ABPs computing abecedarian polynomials can be made abecedarian.
2. $\mathsf{abcd}-\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd}-\mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd}-\mathsf{VP}_{\mathsf{nc}}$.
3. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s implies that there is an abcd-formula computing f of size $s^{O(\log d)}$.

Other Observations

1. Circuits and ABPs computing abecedarian polynomials can be made abecedarian.
2. $\mathsf{abcd} - \mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
3. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s implies that there is an abcd-formula computing f of size $s^{O(\log d)}$.
4. $n^{\omega(1)}$ lower bound against homogeneous formulas for $\mathrm{linked\_CHSYM}_{n,\log n}(\mathsf{x})$ or even $\mathrm{IMM}_{n,\log n}(\mathsf{x})$ implies that $\mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VBP}_{\mathsf{nc}}$.

Other Observations

1. Circuits and ABPs computing abecedarian polynomials can be made abecedarian.
2. $\mathsf{abcd} - \mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{abcd} - \mathsf{VBP}_{\mathsf{nc}} \subsetneq \mathsf{abcd} - \mathsf{VP}_{\mathsf{nc}}$.
3. f is an abcd-polynomial of degree d that is computable by an abcd-ABP of size s implies that there is an abcd-formula computing f of size $s^{O(\log d)}$.
4. $n^{\omega(1)}$ lower bound against homogeneous formulas for linked_$\mathrm{CHSYM}_{n,\log n}(\mathrm{x})$ or even $\mathrm{IMM}_{n,\log n}(\mathrm{x})$ implies that $\mathsf{VF}_{\mathsf{nc}} \neq \mathsf{VBP}_{\mathsf{nc}}$.

Thank you!