Lower Bounds in Algebraic Circuit Complexity

Prerona Chatterjee
Tata Institute of Fundamental Research, Mumbai

September 14, 2021

Theory of Computation

Theory of Computation

Hilbert's Conjecture

There is a set of axioms such that:

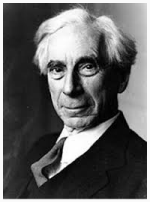
- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.

Theory of Computation

Hilbert's Conjecture

There is a set of axioms such that:

- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.



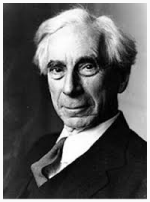
Russell's Principia Mathematica

Theory of Computation

Hilbert's Conjecture

There is a set of axioms such that:

- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.



Russell's Principia Mathematica

Gödel's Incompleteness Theorem

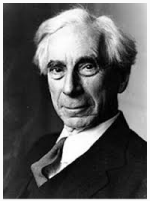
Theory of Computation

Hilbert's Conjecture

There is a set of axioms such that:

- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.

Ok, some statements are not provable.



Russell's Principia Mathematica

Gödel's Incompleteness Theorem

Theory of Computation

Hilbert's Conjecture

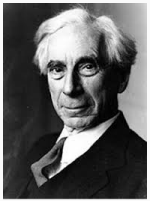
There is a set of axioms such that:

- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.

Ok, some statements are not provable.

But is there an algorithm that can prove every provable statement?

Entscheidungsproblem



Russell's Principia Mathematica

Gödel's Incompleteness Theorem

Theory of Computation

Hilbert's Conjecture

There is a set of axioms such that:

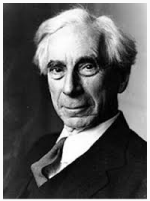
- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.

Ok, some statements are not provable.

But is there an algorithm that can prove every provable statement?

Entscheidungsproblem

What does that even mean?



Russell's Principia Mathematica

Gödel's Incompleteness Theorem

Theory of Computation

Hilbert's Conjecture

There is a set of axioms such that:

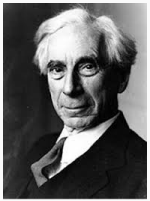
- Everything provable from these axioms is true.
- Every true statement is provable from these axioms.

Ok, some statements are not provable.

But is there an algorithm that can prove every provable statement?

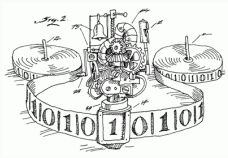
Entscheidungsproblem

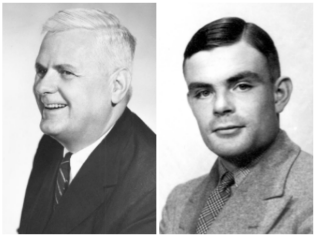
What does that even mean?



Russell's Principia Mathematica

Gödel's Incompleteness Theorem

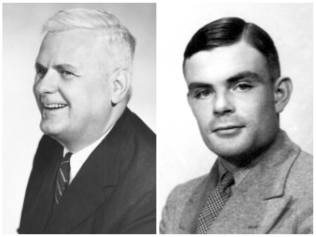




Church-Turing Thesis

Any computation can be simulated by a Turing Machine.

Complexity Theory

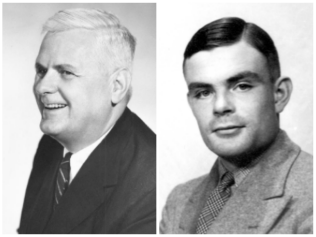


Church-Turing Thesis

Any computation can be simulated by a Turing Machine.

There is no Turing Machine that can solve every problem.

Complexity Theory



Church-Turing Thesis

Any computation can be simulated by a Turing Machine.

There is no Turing Machine that can solve every problem.

Is the given problem computable?

Complexity Theory

Church-Turing Thesis

Any computation can be simulated by a Turing Machine.

There is no Turing Machine that can solve every problem.

Is the given problem computable?

Is it computable efficiently?

Complexity Theory

Church-Turing Thesis

Any computation can be simulated by a Turing Machine.

There is no Turing Machine that can solve every problem.

Is the given problem computable?

Is it computable efficiently?

Edmonds: Efficient $\equiv$ Polynomial Time

Complexity Theory

Church-Turing Thesis

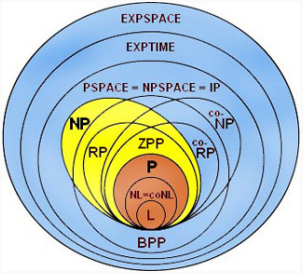
Any computation can be simulated by a Turing Machine.

There is no Turing Machine that can solve every problem.

Is the given problem computable?

Is it computable efficiently?

Edmonds: Efficient $\equiv$ Polynomial Time



Upper Bounds and Lower Bounds

Upper Bound Statements:

Lower Bound Statements:

Upper Bounds and Lower Bounds

Upper Bound Statements:

- There is an algorithm to solve the Sorting Problem in time at most $O(n \log n)$.

Lower Bound Statements:

Upper Bounds and Lower Bounds

Upper Bound Statements:

- There is an algorithm to solve the Sorting Problem in time at most $O(n \log n)$.

Lower Bound Statements:

- Any Sorting algorithm that uses only comparisons must take time at least $\Omega(n \log n)$.

Upper Bounds and Lower Bounds

Upper Bound Statements:

- There is an algorithm to solve the Sorting Problem in time at most $O(n \log n)$.
- There is an algorithm that solves the given problem in time at most $O(T(n))$.

Lower Bound Statements:

- Any Sorting algorithm that uses only comparisons must take time at least $\Omega(n \log n)$.

Upper Bounds and Lower Bounds

Upper Bound Statements:

- There is an algorithm to solve the Sorting Problem in time at most $O(n \log n)$.
- There is an algorithm that solves the given problem in time at most $O(T(n))$.

Lower Bound Statements:

- Any Sorting algorithm that uses only comparisons must take time at least $\Omega(n \log n)$.
- Any algorithm solving the given problem must take time at least $\Omega(T'(n))$.

Upper Bounds and Lower Bounds

Upper Bound Statements:

- There is an algorithm to solve the Sorting Problem in time at most $O(n \log n)$.
- There is an algorithm that solves the given problem in time at most $O(T(n))$.
- There is a circuit that computes the given function of size at most $O(S(n))$.

Lower Bound Statements:

- Any Sorting algorithm that uses only comparisons must take time at least $\Omega(n \log n)$.
- Any algorithm solving the given problem must take time at least $\Omega(T'(n))$.

Upper Bounds and Lower Bounds

Upper Bound Statements:

- There is an algorithm to solve the Sorting Problem in time at most $O(n \log n)$.
- There is an algorithm that solves the given problem in time at most $O(T(n))$.
- There is a circuit that computes the given function of size at most $O(S(n))$.

Lower Bound Statements:

- Any Sorting algorithm that uses only comparisons must take time at least $\Omega(n \log n)$.
- Any algorithm solving the given problem must take time at least $\Omega(T'(n))$.
- Any circuit computing the given problem must be of size at least $\Omega(S'(n))$.

Algebraic Complexity Theory

How hard is it perform the given computational task on the given algebraic object?

Algebraic Complexity Theory

How hard is it perform the given computational task on the given algebraic object?

Agrawal-Kayal-Saxena
PRIMES is in P

Algebraic Complexity Theory

How hard is it perform the given computational task on the given algebraic object?

Agrawal-Kayal-Saxena
PRIMES is in P

Algebraic Circuit Complexity
Can the given polynomial be computed efficiently?

Algebraic Complexity Theory

How hard is it perform the given computational task on the given algebraic object?

Agrawal-Kayal-Saxena
PRIMES is in P

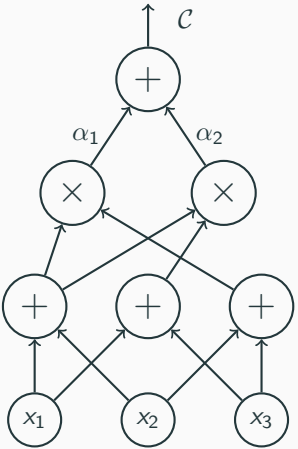
Algebraic Circuit Complexity
Can the given polynomial be computed efficiently?



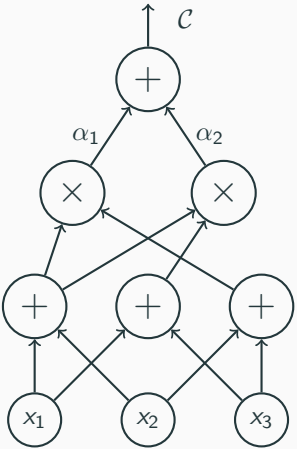


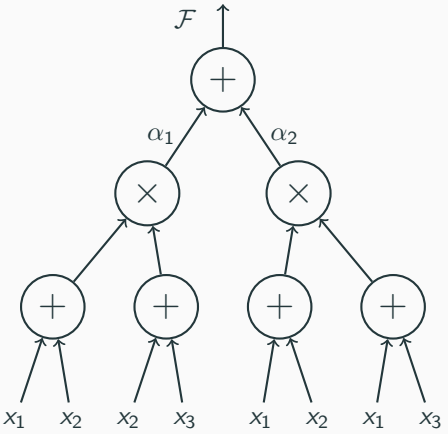



Algebraic Models of Computation

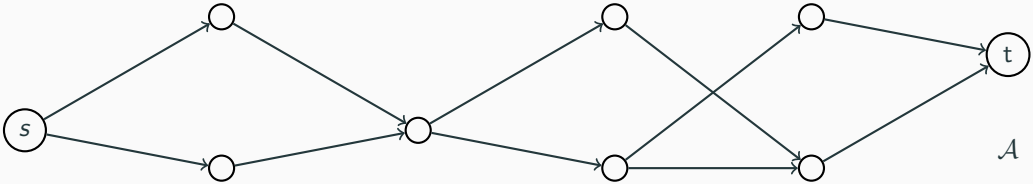


Algebraic Models of Computation

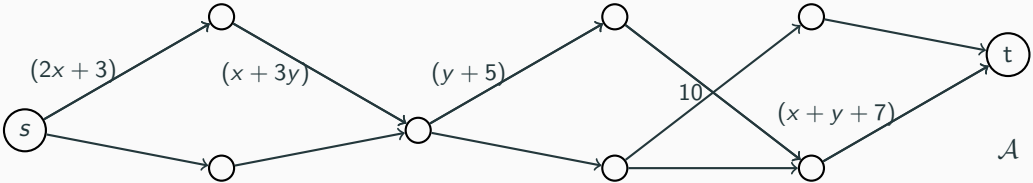




Algebraic Branching Programs

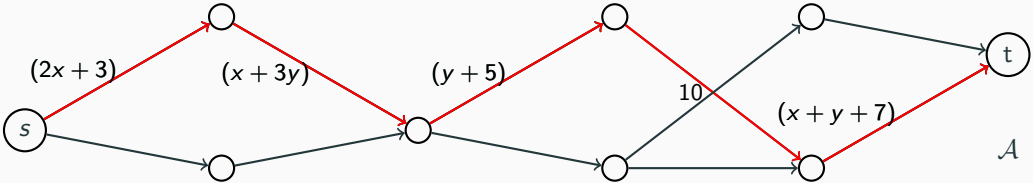


Algebraic Branching Programs



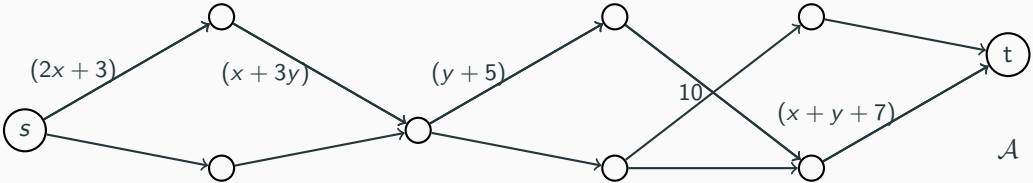
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$

Algebraic Branching Programs



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

Algebraic Branching Programs



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

Lower Bounds in Algebraic Circuit Complexity

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

Lower Bounds in Algebraic Circuit Complexity

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

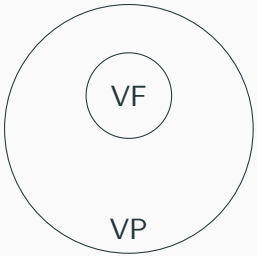
VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

Lower Bounds in Algebraic Circuit Complexity

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

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

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



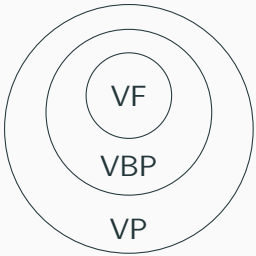
Lower Bounds in Algebraic Circuit Complexity

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

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

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

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



Lower Bounds in Algebraic Circuit Complexity

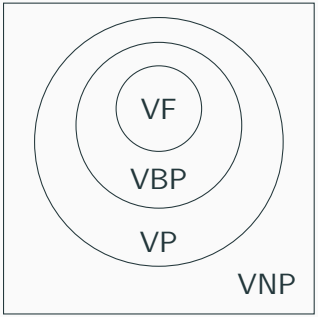
Objects of Study: Polynomials over n variables of degree d.

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

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

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

VNP: Explicit Polynomials



Lower Bounds in Algebraic Circuit Complexity

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

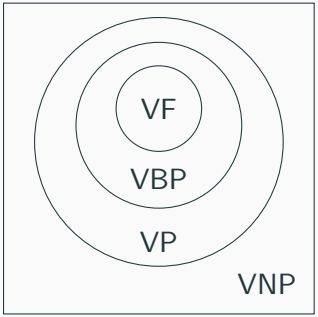
VF: Polynomials computable by formulas of size poly(n, d).

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

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

VNP: Explicit Polynomials

Are the inclusions tight?



Lower Bounds in Algebraic Circuit Complexity

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

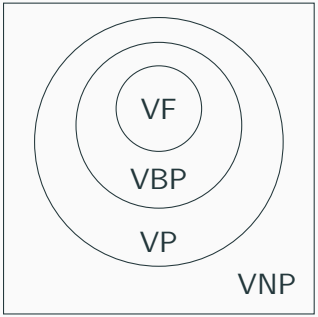
VF: Polynomials computable by formulas of size poly(n, d).

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

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

VNP: Explicit Polynomials

Are the inclusions tight?



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

Questions?

Some Natural Restrictions

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

Some Natural Restrictions

Homogenity
Every monomial is of the same degree.

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

Some Natural Restrictions

Homogenity

Every monomial is of the same degree.

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

Homogeneous Circuits

Every gate computes a homogeneous polynomial.

Some Natural Restrictions

Homogenity
Every monomial is of the same degree.

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

Multilinearity
No variable occurs more than once in any monomial.

Homogeneous Circuits
Every gate computes a homogeneous polynomial.

Some Natural Restrictions

Homogenity
Every monomial is of the same degree.

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

Multilinearity
No variable occurs more than once in any monomial.

Homogeneous Circuits
Every gate computes a homogeneous polynomial.

Multilinear Circuits
Every gate computes a multilinear polynomial.

Some Natural Restrictions

Homogenity
Every monomial is of the same degree.

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

Multilinearity
No variable occurs more than once in any monomial.

Homogeneous Circuits
Every gate computes a homogeneous polynomial.

Multilinear Circuits
Every gate computes a multilinear polynomial.

Non-Commutative Circuits
The multiplication gates are non-commutative.

Some Natural Restrictions

Homogenity
Every monomial is of the same degree.

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

Multilinearity
No variable occurs more than once in any monomial.

Homogeneous Circuits
Every gate computes a homogeneous polynomial.

Multilinear Circuits
Every gate computes a multilinear polynomial.

Non-Commutative Circuits
The multiplication gates are non-commutative.

Constant Depth Circuits
Length of the longest root-to-leaf path is a constant independent of n.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

[Agrawal-Vinay, Koiran, Tavenas]:
The lower bound is tight at depth 4.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

[Agrawal-Vinay, Koiran, Tavenas]:
The lower bound is tight at depth 4.

[Gupta-Kamath-Kayal-Saptharishi]:
The lower bound is tight at depth 3.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

[Agrawal-Vinay, Koiran, Tavenas]:
The lower bound is tight at depth 4.

[Gupta-Kamath-Kayal-Saptharishi]:
The lower bound is tight at depth 3.

Multilinear Setting

[Dvir-Malod-Perifel-Yehudayoff]:
$\mathrm{VF} \subsetneq \mathrm{VBP}$.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

[Agrawal-Vinay, Koiran, Tavenas]:
The lower bound is tight at depth 4.

[Gupta-Kamath-Kayal-Saptharishi]:
The lower bound is tight at depth 3.

Multilinear Setting

[Dvir-Malod-Perifel-Yehudayoff]:
VF $\subsetneq$ VBP.

Non-Commutative Setting

[Nisan]: VBP $\subsetneq$ VP.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

[Agrawal-Vinay, Koiran, Tavenas]:
The lower bound is tight at depth 4.

[Gupta-Kamath-Kayal-Saptharishi]:
The lower bound is tight at depth 3.

Multilinear Setting

[Dvir-Malod-Perifel-Yehudayoff]:
$\mathrm{VF} \subsetneq \mathrm{VBP}$.

Non-Commutative Setting

[Nisan]: $\mathrm{VBP} \subsetneq \mathrm{VP}$.

[L-S-T]: $\mathrm{hom} - \mathrm{VF} \subsetneq \mathrm{hom} - \mathrm{VBP}$.

Successes in Restricted Models

Constant Depth Setting

[Limaye-Srinivasan-Tavenas]:
Super-polynomial lower bound.

[Agrawal-Vinay, Koiran, Tavenas]:
The lower bound is tight at depth 4.

[Gupta-Kamath-Kayal-Saptharishi]:
The lower bound is tight at depth 3.

Multilinear Setting

[Dvir-Malod-Perifel-Yehudayoff]:
VF $\subsetneq$ VBP.

Non-Commutative Setting

[Nisan]: VBP $\subsetneq$ VP.

[L-S-T]: hom $-$ VF $\subsetneq$ hom $-$ VBP.

[Cha]: abcd $-$ VF $\subsetneq$ abcd $-$ VBP.

Lower Bounds for General Models

General Circuits

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

Lower Bounds for General Models

General Circuits

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

Homogeneous ABPs

[Kumar]: Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

Lower Bounds for General Models

General Circuits

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

Homogeneous ABPs

[Kumar]: Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General ABPs

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

Lower Bounds for General Models (contd.)

General Formulas

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

Lower Bounds for General Models (contd.)

General Formulas

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

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

Lower Bounds for General Models (contd.)

General Formulas

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

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

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

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

Natural Proofs

Are the proof techniques used against structured models useful against general models?

Natural Proofs

Are the proof techniques used against structured models useful against general models?

[Forbes-Shpilka-Volk, Grochow-Kumar-Saks-Saraf]: Defined Algebraically Natural Proofs.

Natural Proofs

Are the proof techniques used against structured models useful against general models?

[Forbes-Shpilka-Volk, Grochow-Kumar-Saks-Saraf]: Defined Algebraically Natural Proofs.

[F-S-V]: Natural proofs might not exist.

Natural Proofs

Are the proof techniques used against structured models useful against general models?

[Forbes-Shpilka-Volk, Grochow-Kumar-Saks-Saraf]: Defined Algebraically Natural Proofs.

[F-S-V]: Natural proofs might not exist.

[C-Kumar-Ramya-Saptharishi-Tengse]: Natural proofs might exist.

Natural Proofs

Are the proof techniques used against structured models useful against general models?

[Forbes-Shpilka-Volk, Grochow-Kumar-Saks-Saraf]: Defined Algebraically Natural Proofs.

[F-S-V]: Natural proofs might not exist.

[C-Kumar-Ramya-Saptharishi-Tengse]: Natural proofs might exist.

Let VP' be the polynomials in VP with $\{-1, 0, 1\}$ coefficients. Then VP' has VP natural proofs.

Natural Proofs

Are the proof techniques used against structured models useful against general models?

[Forbes-Shpilka-Volk, Grochow-Kumar-Saks-Saraf]: Defined Algebraically Natural Proofs.

[F-S-V]: Natural proofs might not exist.

[C-Kumar-Ramya-Saptharishi-Tengse]: Natural proofs might exist.

Let VP' be the polynomials in VP with $\{-1, 0, 1\}$ coefficients. Then VP' has VP natural proofs.

[K-R-S-T]: Natural proofs might not exist.

Natural Proofs

Are the proof techniques used against structured models useful against general models?

[Forbes-Shpilka-Volk, Grochow-Kumar-Saks-Saraf]: Defined Algebraically Natural Proofs.

[F-S-V]: Natural proofs might not exist.

[C-Kumar-Ramya-Saptharishi-Tengse]: Natural proofs might exist.

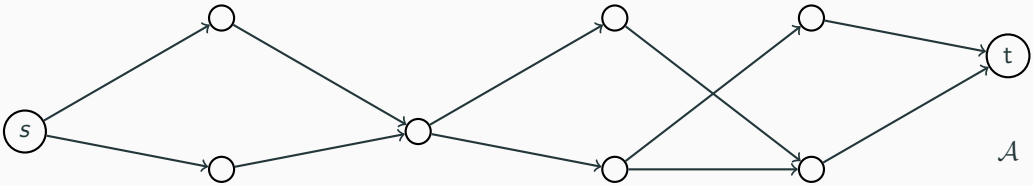
Let VP' be the polynomials in VP with $\{-1, 0, 1\}$ coefficients. Then VP' has VP natural proofs.

[K-R-S-T]: Natural proofs might not exist.

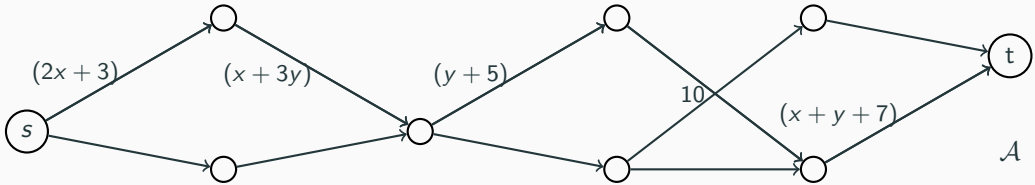
Under some believable assumption, VNP does not have VP natural proofs.

Questions?

Algebraic Branching Programs

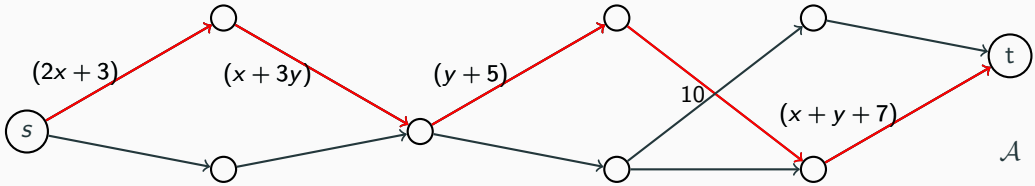


Algebraic Branching Programs



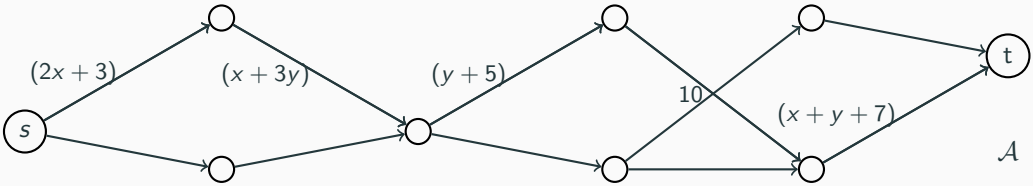
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$

Algebraic Branching Programs



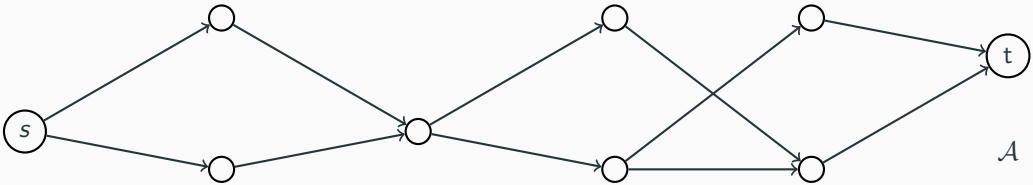
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on p

Algebraic Branching Programs



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

Algebraic Branching Programs



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

Question: Is there an explicit polynomial that can not be computed by efficient ABPs?

Lower Bounds Against ABPs

Previous Work

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

Lower Bounds Against ABPs

Previous Work

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

[Kumar]: Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Lower Bounds Against ABPs

Previous Work

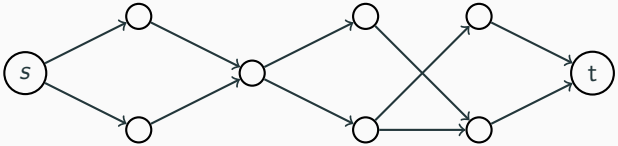
[Baur-Strassen]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

[Kumar]: Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Our Result

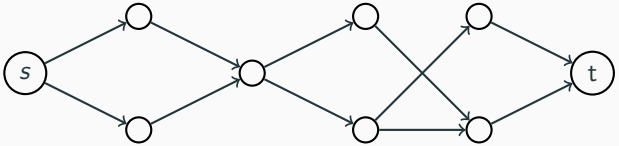
[C-Kumar-She-Volk]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

The Homogeneous Case: Proof Overview



$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

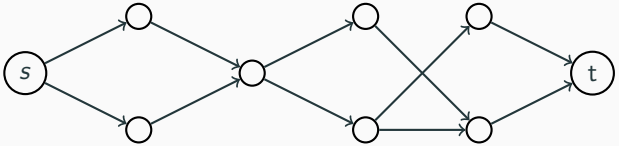
The Homogeneous Case: Proof Overview



$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t]$$

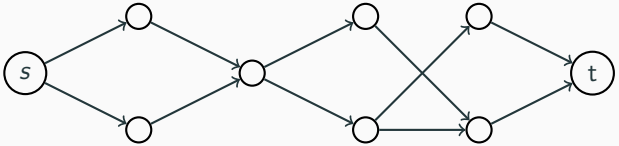
The Homogeneous Case: Proof Overview



$$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d.$$

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t] = \sum_{i=1}^{t_k} g_i \cdot h_i$$

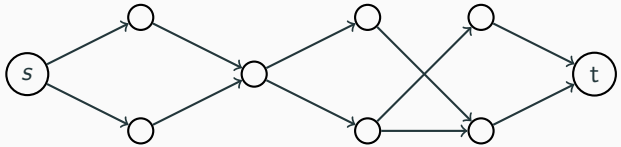
The Homogeneous Case: Proof Overview



$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t] = \sum_{i=1}^{t_k} g_i \cdot h_i = \sum_{i=1}^{t_k} (g_i' + \alpha_i) \cdot (h_i' + \beta_i)$$

The Homogeneous Case: Proof Overview

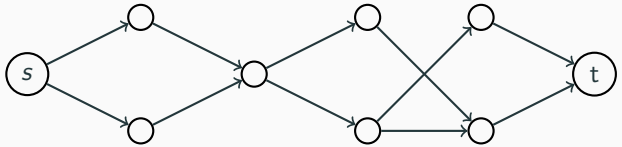


$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t] = \sum_{i=1}^{t_k} g_i \cdot h_i = \sum_{i=1}^{t_k} (g_i' + \alpha_i) \cdot (h_i' + \beta_i)$$

For $R = \sum_{i=1}^{t_k} (\alpha_i \cdot h_i' + \beta_i \cdot g_i' + \alpha_i \cdot \beta_i)$,

The Homogeneous Case: Proof Overview

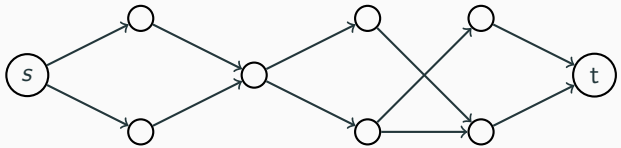


$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t] = \sum_{i=1}^{t_k} g_i \cdot h_i = \sum_{i=1}^{t_k} (g_i' + \alpha_i) \cdot (h_i' + \beta_i)$$

For $R = \sum_{i=1}^{t_k} (\alpha_i \cdot h_i' + \beta_i \cdot g_i' + \alpha_i \cdot \beta_i)$, $\quad P_{n,d} = \left(\sum_{i=1}^{t_k} g_i' \cdot h_i' \right) + R$

The Homogeneous Case: Proof Overview

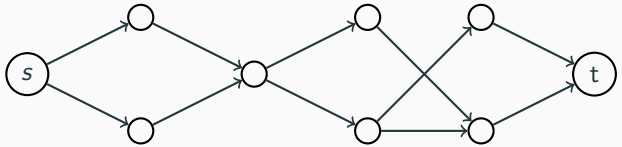


$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t] = \sum_{i=1}^{t_k} g_i \cdot h_i = \sum_{i=1}^{t_k} (g_i' + \alpha_i) \cdot (h_i' + \beta_i)$$

For $R = \sum_{i=1}^{t_k} (\alpha_i \cdot h_i' + \beta_i \cdot g_i' + \alpha_i \cdot \beta_i)$, $\quad P_{n,d} = \left(\sum_{i=1}^{t_k} g_i' \cdot h_i' \right) + R \implies P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i'$.

The Homogeneous Case: Proof Overview



$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

$$P_{n,d} = \sum_{i=1}^{t_k} [s, v_i] \cdot [v_i, t] = \sum_{i=1}^{t_k} g_i \cdot h_i = \sum_{i=1}^{t_k} (g_i' + \alpha_i) \cdot (h_i' + \beta_i)$$

$$\text{For } R = \sum_{i=1}^{t_k} (\alpha_i \cdot h_i' + \beta_i \cdot g_i' + \alpha_i \cdot \beta_i), \quad P_{n,d} = \left(\sum_{i=1}^{t_k} g_i' \cdot h_i' \right) + R \implies P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i'.$$

Note: $\deg(g_i'), \deg(h_i') \leq [d-1]$ for every $i \in [d-1]$. Thus $\deg(R) \leq d-1$.

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i \in [n]}$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V}$ = Set of common zeroes of S and S'

The Homogeneous Case: Proof Overview (contd.)

$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i'$ where $P_{n,d} = \sum_{i=1}^{n} x_i^d$

Look at the first order derivatives.

$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]}$ $S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$

$\mathcal{V}$ = Set of common zeroes of S and S' $\mathcal{V}'$ = Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

The Homogeneous Case: Proof Overview (contd.)

$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i'$ where $P_{n,d} = \sum_{i=1}^{n} x_i^d$

Look at the first order derivatives.

$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]}$ $S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$

$\mathcal{V}$ = Set of common zeroes of S and S' $\mathcal{V}'$ = Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V}$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V}$ = Set of common zeroes of S and S' $\qquad$ $\mathcal{V}'$ = Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

The Homogeneous Case: Proof Overview (contd.)

$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i'$ where $P_{n,d} = \sum_{i=1}^{n} x_i^d$

Look at the first order derivatives.

$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]}$ $S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$

$\mathcal{V}$ = Set of common zeroes of S and S' $\mathcal{V}'$ = Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$R = 0$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V} =$ Set of common zeroes of S and S' $\qquad$ $\mathcal{V}' =$ Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$$R = 0 \quad \Rightarrow \quad S = \left\{d \cdot x_1^{d-1}, \ldots, d \cdot x_n^{d-1}\right\}$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V} =$ Set of common zeroes of S and S' $\qquad$ $\mathcal{V}' =$ Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$$R = 0 \quad \Rightarrow \quad S = \left\{d \cdot x_1^{d-1}, \ldots, d \cdot x_n^{d-1}\right\} \quad \Rightarrow \quad \dim(\mathcal{V}) = 0$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V} =$ Set of common zeroes of S and S' $\qquad$ $\mathcal{V}' =$ Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$$\deg(R) \leq d-1 \quad \implies \quad \dim(\mathcal{V}) = 0$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V} =$ Set of common zeroes of S and S' $\qquad$ $\mathcal{V}' =$ Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$$\deg(R) \leq d-1 \quad \implies \quad \dim(\mathcal{V}) = 0$$

$$\dim(\mathcal{V}') \geq n - 2t_k$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V} =$ Set of common zeroes of S and S' $\qquad$ $\mathcal{V}' =$ Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$$\deg(R) \leq d-1 \quad \implies \quad \dim(\mathcal{V}) = 0$$

$$\dim(\mathcal{V}') \geq n - 2t_k \implies n - 2t_k \leq 0$$

The Homogeneous Case: Proof Overview (contd.)

$$P_{n,d} - R = \sum_{i=1}^{t_k} g_i' \cdot h_i' \qquad \text{where} \qquad P_{n,d} = \sum_{i=1}^{n} x_i^d$$

Look at the first order derivatives.

$$S = \{\partial_{x_i}(P_{n,d} - R)\} = \left\{d \cdot x_i^{d-1} - \partial_{x_i}(R)\right\}_{i\in[n]} \qquad S' = \left\{\sum_{j=1}^{t_k}(g_j' \cdot \partial_{x_i} h_j' + h_j' \cdot \partial_{x_i} g_j')\right\}_{i\in[n]}$$

$\mathcal{V} =$ Set of common zeroes of S and S' $\qquad$ $\mathcal{V}' =$ Set of common zeroes of $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

$$\mathcal{V}' \subseteq \mathcal{V} \implies \dim(\mathcal{V}') \leq \dim(\mathcal{V})$$

$$\deg(R) \leq d-1 \quad \implies \quad \dim(\mathcal{V}) = 0$$

$$\dim(\mathcal{V}') \geq n - 2t_k \implies n - 2t_k \leq 0 \implies t_k \geq n/2$$

Proof Overview Of Our Result

Step 0 ([Kumar]): Look at the homogeneous case

Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Proof Overview Of Our Result

Step 0 ([Kumar]): Look at the homogeneous case

Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Step 1: Generalise above statement to get the base case

Any ABP with $(d+1)$ layers

Proof Overview Of Our Result

Step 0 ([Kumar]): Look at the homogeneous case

Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Step 1: Generalise above statement to get the base case

Any ABP with $(d+1)$ layers computing a polynomial of the form

$$f = \sum_{i=1}^{n} x_i^d + \sum_{i=1}^{r} A_i(\mathbf{x}) \cdot B_i(\mathbf{x}) + \delta(\mathbf{x})$$

Proof Overview Of Our Result

Step 0 ([Kumar]): Look at the homogeneous case

Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Step 1: Generalise above statement to get the base case

Any ABP with $(d+1)$ layers computing a polynomial of the form

$$f = \sum_{i=1}^{n} x_i^d + \sum_{i=1}^{r} A_i(\mathbf{x}) \cdot B_i(\mathbf{x}) + \delta(\mathbf{x})$$

where $A_i(0) = 0 = B_i(0)$ and $\deg(\delta(\mathbf{x})) < d$,

Proof Overview Of Our Result

Step 0 ([Kumar]): Look at the homogeneous case

Any ABP with $(d+1)$ layers computing $\sum_{i=1}^{n} x_i^d$ has $\Omega(nd)$ vertices.

Step 1: Generalise above statement to get the base case

Any ABP with $(d+1)$ layers computing a polynomial of the form

$$f = \sum_{i=1}^{n} x_i^d + \sum_{i=1}^{r} A_i(\mathbf{x}) \cdot B_i(\mathbf{x}) + \delta(\mathbf{x})$$

where $A_i(0) = 0 = B_i(0)$ and $\deg(\delta(\mathbf{x})) < d$, has at least

$((n/2) - r) \cdot (d-1)$ vertices.

Proof Overview Of Our Result (contd.)

Step 2: Iteratively reduce to Base Case

In each iteration, reduce the number of layers till it becomes $(d+1)$ such that

Proof Overview Of Our Result (contd.)

Step 2: Iteratively reduce to Base Case

In each iteration, reduce the number of layers till it becomes $(d+1)$ such that

- the number of layers is reduced by a constant fraction,

Proof Overview Of Our Result (contd.)

Step 2: Iteratively reduce to Base Case

In each iteration, reduce the number of layers till it becomes $(d+1)$ such that

- the number of layers is reduced by a constant fraction,
- the size does not increase,

Proof Overview Of Our Result (contd.)

Step 2: Iteratively reduce to Base Case

In each iteration, reduce the number of layers till it becomes $(d+1)$ such that

- the number of layers is reduced by a constant fraction,
- the size does not increase,
- the polynomial being computed continues to look like

$$f_{\ell+1} = \sum_{i=1}^{n} x_i^d + \sum_{i=1}^{r_{\ell+1}} A_i(\mathbf{x}) \cdot B_i(\mathbf{x}) + \delta_{\ell+1}(\mathbf{x})$$

where $A_i(0) = 0 = B_i(0)$ and $\deg(\delta_{\ell+1}(\mathbf{x})) < d$,

Proof Overview Of Our Result (contd.)

Step 2: Iteratively reduce to Base Case

In each iteration, reduce the number of layers till it becomes $(d+1)$ such that

- the number of layers is reduced by a constant fraction,
- the size does not increase,
- the polynomial being computed continues to look like

$$f_{\ell+1} = \sum_{i=1}^{n} x_i^d + \sum_{i=1}^{r_{\ell+1}} A_i(\mathbf{x}) \cdot B_i(\mathbf{x}) + \delta_{\ell+1}(\mathbf{x})$$

  where $A_i(0) = 0 = B_i(0)$ and $\deg(\delta_{\ell+1}(\mathbf{x})) < d$,

- number of error terms collected is small.

The Induction Step

ℓ-th step

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1}$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$

Number of layers $= d_\ell$

Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$

Number of layers $= d_{\ell+1}$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$
Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$

Number of layers $= d_\ell$

Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$

Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$

Number of error terms $= r_{\ell+1}$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$
Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$
Number of error terms $= r_{\ell+1} \leq r_\ell + \frac{s_\ell}{d_\ell/3}$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$
Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$
Number of error terms $= r_{\ell+1} \leq r_\ell + \frac{s_\ell}{d_\ell/3}$

Number of Steps: $\Theta(\log n)$

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

Size $= s_{\ell+1} \leq s_\ell$
Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$
Number of error terms $= r_{\ell+1} \leq r_\ell + \frac{s_\ell}{d_\ell/3}$

Number of Steps: $\Theta(\log n)$

Number of Error Terms: $\varepsilon \cdot n$ where ε can be chosen.

The Induction Step

ℓ-th step

Given: $\mathcal{A}_\ell$

Size $= s_\ell$
Number of layers $= d_\ell$
Number of error terms $= r_\ell$

Want to construct: $\mathcal{A}_{\ell+1}$

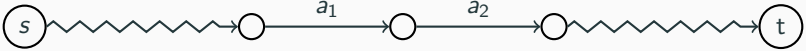
Size $= s_{\ell+1} \leq s_\ell$
Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$
Number of error terms $= r_{\ell+1} \leq r_\ell + \frac{s_\ell}{d_\ell/3}$

Number of Steps: $\Theta(\log n)$

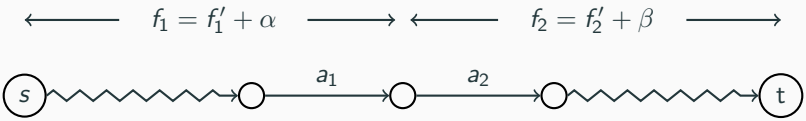
Number of Error Terms: $\varepsilon \cdot n$ where ε can be chosen.

The Lower Bound: $((n/2) - \varepsilon \cdot n) \cdot (d-1)$

Proof of the Induction Step

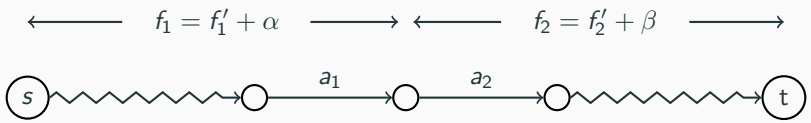


Proof of the Induction Step



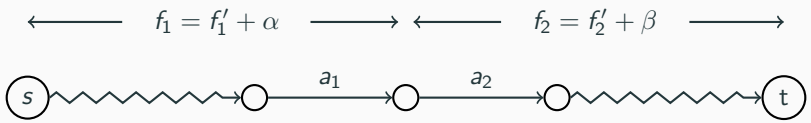
Proof of the Induction Step

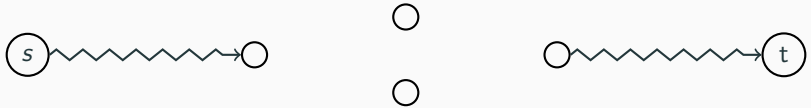
$\mathcal{A}_\ell = f_1 \cdot f_2$



Proof of the Induction Step

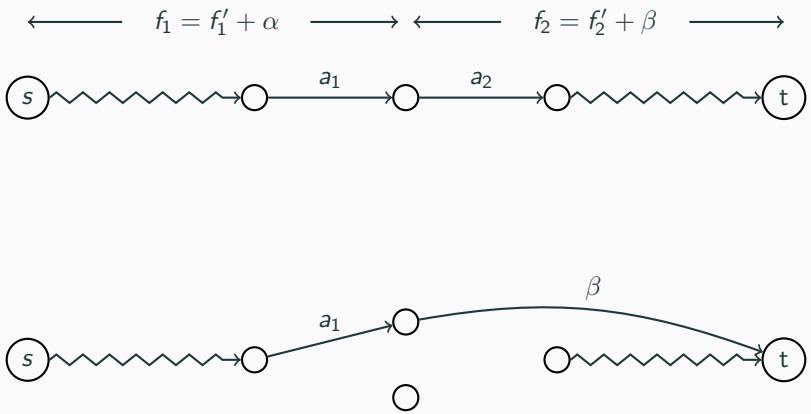
$\mathcal{A}_\ell = f_1 \cdot f_2$





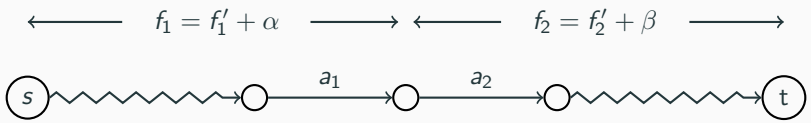
Proof of the Induction Step

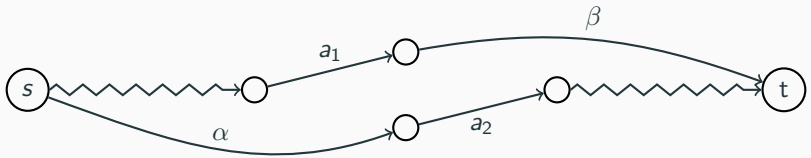
$\mathcal{A}_\ell = f_1 \cdot f_2$



Proof of the Induction Step

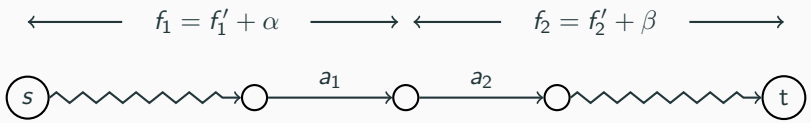
$\mathcal{A}_\ell = f_1 \cdot f_2$



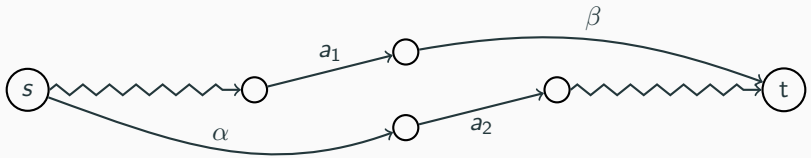


Proof of the Induction Step

$\mathcal{A}_\ell = f_1 \cdot f_2$

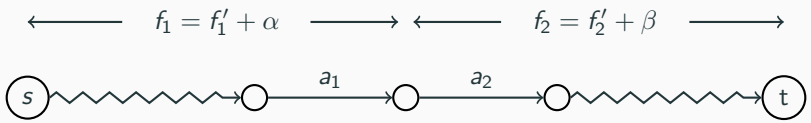


$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$

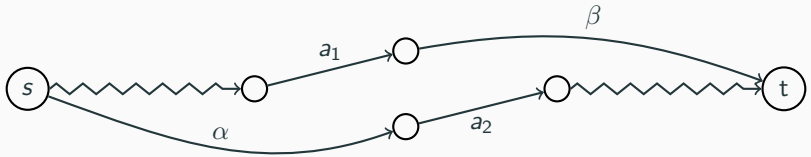


Proof of the Induction Step

$\mathcal{A}_\ell = f_1 \cdot f_2$

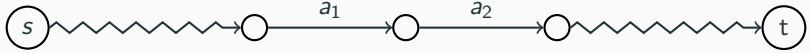


$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2 = \mathcal{A}_\ell - f_1' \cdot f_2' + \alpha \cdot \beta$

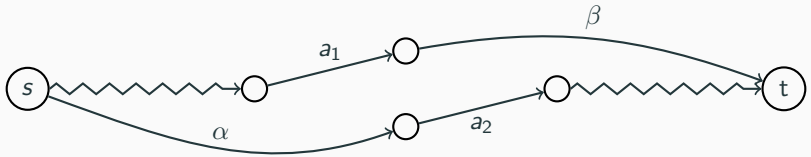


Proof of the Induction Step

$\mathcal{A}_\ell = f_1 \cdot f_2$

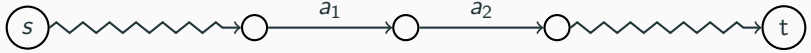


$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$



Proof of the Induction Step

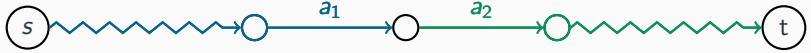
$\mathcal{A}_\ell = f_1 \cdot f_2$



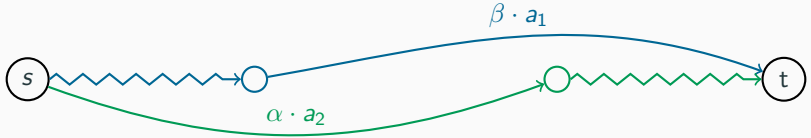
$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$

Proof of the Induction Step

$\mathcal{A}_\ell = f_1 \cdot f_2$



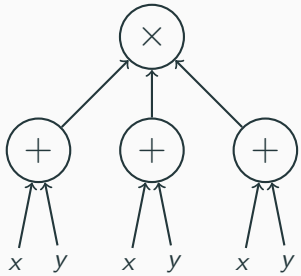
$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$



Algebraic Formulas And Lower Bounds Against Them

$$x^3 + 3x^2y + 3xy^2 + y^3 = (x + y)^3$$

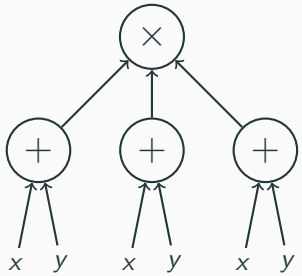
$$(x + y) \cdot (x + y) \cdot (x + y)$$



Algebraic Formulas And Lower Bounds Against Them

$$x^3 + 3x^2y + 3xy^2 + y^3 = (x+y)^3$$

$$(x+y)\cdot(x+y)\cdot(x+y)$$

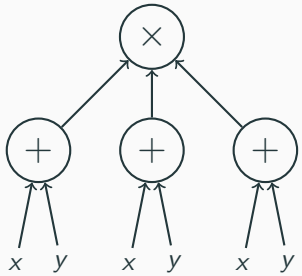


[Kalorkoti]: Any formula computing $\mathrm{Det}_{n\times n}(\mathbf{x})$ has atleast $\Omega(n^3)$ wires.

Algebraic Formulas And Lower Bounds Against Them

$$x^3 + 3x^2y + 3xy^2 + y^3 = (x+y)^3$$

$$(x+y)\cdot(x+y)\cdot(x+y)$$



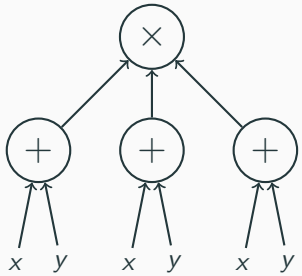
[Kalorkoti]: Any formula computing $\mathrm{Det}_{n\times n}(\mathbf{x})$ has atleast $\Omega(n^3)$ wires.

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

Algebraic Formulas And Lower Bounds Against Them

$$x^3 + 3x^2y + 3xy^2 + y^3 = (x+y)^3$$

$$(x+y)\cdot(x+y)\cdot(x+y)$$



[Kalorkoti]: Any formula computing $\mathrm{Det}_{n\times n}(\mathbf{x})$ has atleast $\Omega(n^3)$ wires.

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

Our Result: Any formula computing $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$ has atleast $\Omega(n^2)$ vertices, where

$$\mathrm{ESYM}_{n,0.1n}(\mathbf{x}) = \sum_{i_1<\cdots<i_{0.1n}\in[n]} \prod_{j=1}^{0.1n} x_{i_j}.$$

Some Subtelties: Why Multilinear?

[Shpilka-Yehudayoff]: Any formula computing $\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^j y_j$ requires $\Omega(n^2)$ wires.

Some Subtelties: Why Multilinear?

[Shpilka-Yehudayoff]: Any formula computing $\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^j y_j$ requires $\Omega(n^2)$ wires.

Why is Our Result Interesting Then?

- This is trivial to show since there are n variables that have *individual degree* n.

Some Subtelties: Why Multilinear?

[Shpilka-Yehudayoff]: Any formula computing $\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^j y_j$ requires $\Omega(n^2)$ wires.

Why is Our Result Interesting Then?

- This is trivial to show since there are n variables that have *individual degree* n.
- Interesting only when the bound is $\omega(\sum_{i \in [n]} d_i)$ when d_i is the individual degree of x_i.

Some Subtelties: Why Multilinear?

[Shpilka-Yehudayoff]: Any formula computing $\sum_{i=1}^{n}\sum_{j=1}^{n} x_i^j y_j$ requires $\Omega(n^2)$ wires.

Why is Our Result Interesting Then?

- This is trivial to show since there are n variables that have *individual degree* n.
- Interesting only when the bound is $\omega(\sum_{i\in[n]} d_i)$ when d_i is the individual degree of x_i.

Multilinear Polynomials Are An Interesting Subclass

- Individual Degree of every variable is 1.

Some Subtelties: Why Multilinear?

[Shpilka-Yehudayoff]: Any formula computing $\sum_{i=1}^{n}\sum_{j=1}^{n} x_i^j y_j$ requires $\Omega(n^2)$ wires.

Why is Our Result Interesting Then?

- This is trivial to show since there are n variables that have *individual degree* n.
- Interesting only when the bound is $\omega(\sum_{i\in[n]} d_i)$ when d_i is the individual degree of x_i.

Multilinear Polynomials Are An Interesting Subclass

- Individual Degree of every variable is 1.
- Multilinearisation of the SY polynomial gives an $\Omega(n^2/\log n)$ lower bound.

Some Subtelties: Why Multilinear?

[Shpilka-Yehudayoff]: Any formula computing $\sum_{i=1}^{n}\sum_{j=1}^{n} x_i^j y_j$ requires $\Omega(n^2)$ wires.

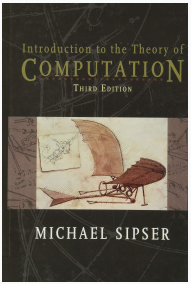
Why is Our Result Interesting Then?

- This is trivial to show since there are n variables that have *individual degree* n.
- Interesting only when the bound is $\omega(\sum_{i\in[n]} d_i)$ when d_i is the individual degree of x_i.

Multilinear Polynomials Are An Interesting Subclass

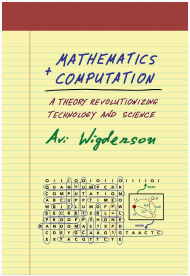
- Individual Degree of every variable is 1.
- Multilinearisation of the SY polynomial gives an $\Omega(n^2/\log n)$ lower bound.
- Kalorkoti's method can not give a better bound against multilinear polynomials.

Conclusion

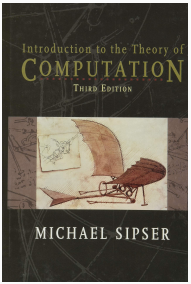


Complexity Theory is Cool!

You should do it too... :)

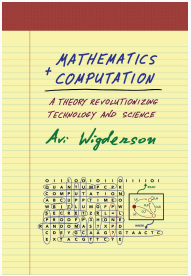


Conclusion



Complexity Theory is Cool!

You should do it too... :)



Thank You !

Webpage: preronac.bitbucket.io

Email: prerona.ch@gmail.com