Understanding the Power of Algebraic Models of Computation

Prerona Chatterjee
NISER, Bhubaneswar (OCC of Homi Bhabha National Institute, Mumbai)

July 10, 2026

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Class I v.s. Class II v.s. Class IV/V

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Class I v.s. Class II v.s. Class IV/V

- Why?

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Class I v.s. Class II v.s. Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Class I v.s. Class II v.s. Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition?

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Class I v.s. Class II v.s. Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

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

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

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

- design a computational model that captures the constraints

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

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

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

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d}$, how many '$+$'s and '$\times$'s does it take to compute f?

Complexity of Computing Polynomials

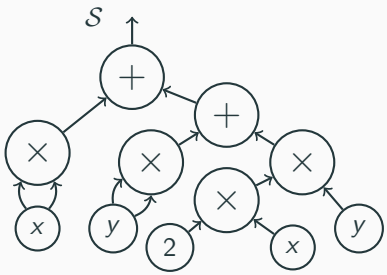
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d}$, how many '$+$'s and '$\times$'s does it take to compute f?

Representing Polynomials: $(x+y)^2 = (x+y) \cdot (x+y) = x^2 + 2xy + y^2$.

Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d}$, how many '$+$'s and '$\times$'s does it take to compute f?

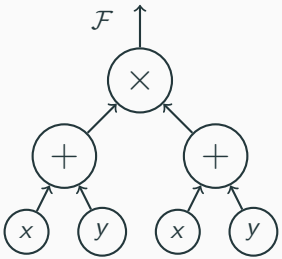
Representing Polynomials: $(x+y)^2 = (x+y) \cdot (x+y) = x^2 + 2xy + y^2$.

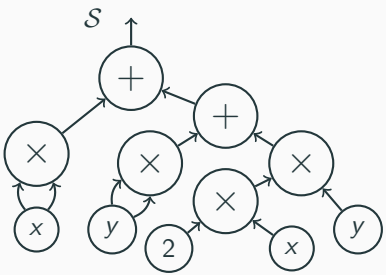


Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d}$, how many '$+$'s and '$\times$'s does it take to compute f?

Representing Polynomials: $(x+y)^2 = (x+y) \cdot (x+y) = x^2 + 2xy + y^2$.

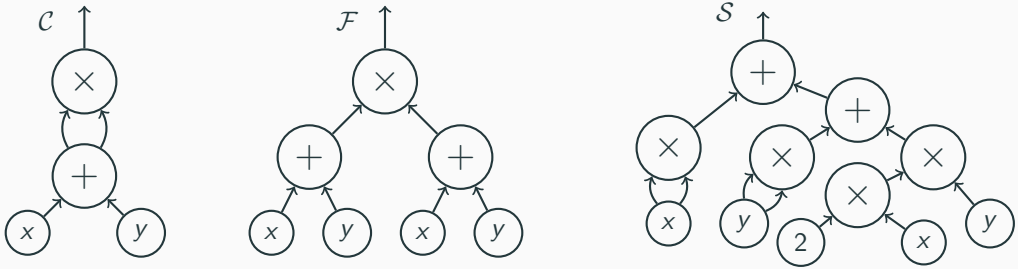




Complexity of Computing Polynomials

Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d}$, how many '$+$'s and '$\times$'s does it take to compute f?

Representing Polynomials: $(x+y)^2 = (x+y)\cdot(x+y) = x^2 + 2xy + y^2$.



Why Computer Scientists are Interested

- Many problems reduce to understanding properties of polynomials.
  Eg: Error Correcting Codes, Algorithm Design, Cryptography, Complexity Theory and many many more...

Why Computer Scientists are Interested

- Many problems reduce to understanding properties of polynomials.
  Eg: Error Correcting Codes, Algorithm Design, Cryptography, Complexity Theory and many many more...
- Certain polynomials show up repeatedly during algorithm design.
  Being able to succinctly compute these help improve the run-time of the algorithms.

Why Computer Scientists are Interested

- Many problems reduce to understanding properties of polynomials.
  Eg: Error Correcting Codes, Algorithm Design, Cryptography, Complexity Theory and many many more...
- Certain polynomials show up repeatedly during algorithm design.
  Being able to succinctly compute these help improve the run-time of the algorithms.
- Connection to $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$: Are all easily verfiable problems also easy to compute?

$$\text{All } \textit{explicit} \text{ polynomials are efficiently computable} \stackrel{\text{G.R.H.}}{\Longrightarrow} \mathrm{P} = \mathrm{NP}.$$

Why Computer Scientists are Interested

- Many problems reduce to understanding properties of polynomials.
  Eg: Error Correcting Codes, Algorithm Design, Cryptography, Complexity Theory and many many more...
- Certain polynomials show up repeatedly during algorithm design.
  Being able to succinctly compute these help improve the run-time of the algorithms.
- Connection to $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$: Are all easily verfiable problems also easy to compute?

  $$\text{All } \textit{explicit} \text{ polynomials are efficiently computable} \stackrel{\text{G.R.H.}}{\Longrightarrow} \mathrm{P} = \mathrm{NP}.$$

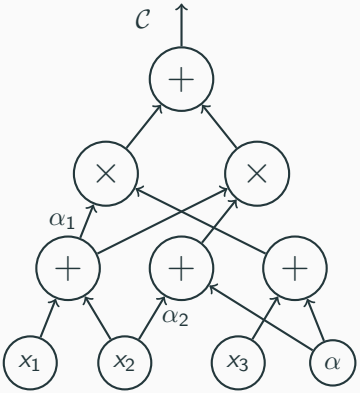
- **[Shamir 79, Lipton 94]**: If $h(x) = \prod_{i=1}^{d}(x - i)$ can be computed using $\mathrm{poly}(\log d)$ additions and multiplications, then integer factoring is easy for boolean circuits.

Algebraic Models of Computation

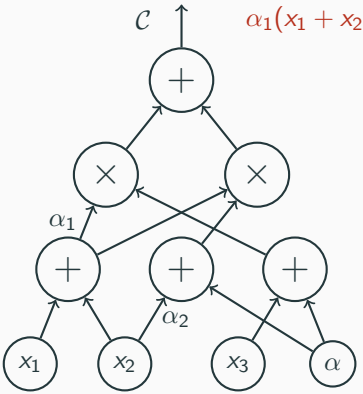
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?

Algebraic Models of Computation

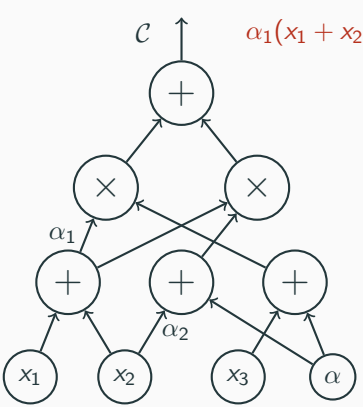
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?



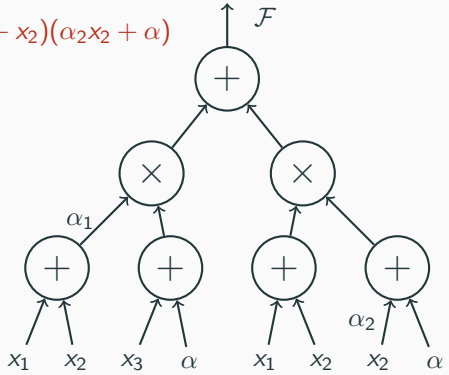
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?



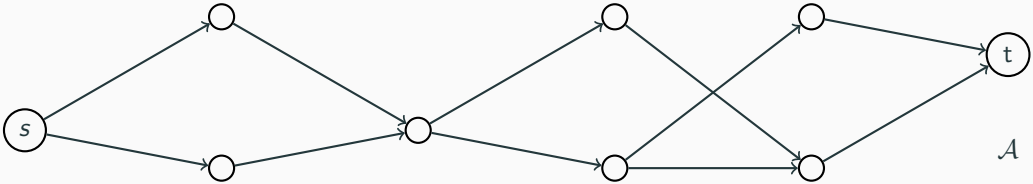
Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?



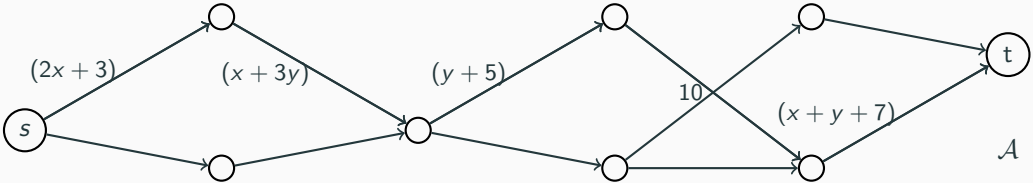
$$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$$



Algebraic Branching Programs

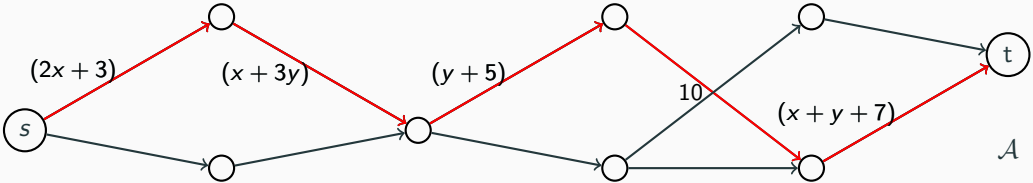


Algebraic Branching Programs



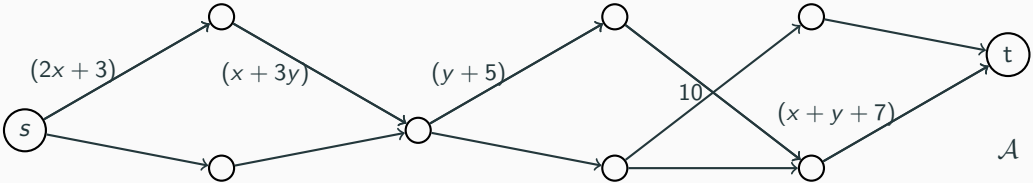
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$

Algebraic Branching Programs



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

Iterated Matrix Multiplication

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}^T \begin{bmatrix} \ell_{11}^{(1)} & \cdots & \ell_{1j}^{(1)} & \cdots & \ell_{1w}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(1)} & \cdots & \ell_{ij}^{(1)} & \cdots & \ell_{iw}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(1)} & \cdots & \ell_{wj}^{(1)} & \cdots & \ell_{ww}^{(1)} \end{bmatrix} \cdots\cdots \begin{bmatrix} \ell_{11}^{(d)} & \cdots & \ell_{1j}^{(d)} & \cdots & \ell_{1w}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(d)} & \cdots & \ell_{ij}^{(d)} & \cdots & \ell_{iw}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(d)} & \cdots & \ell_{wj}^{(d)} & \cdots & \ell_{ww}^{(d)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Iterated Matrix Multiplication

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}^T \begin{bmatrix} \ell_{11}^{(1)} & \cdots & \ell_{1j}^{(1)} & \cdots & \ell_{1w}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(1)} & \cdots & \ell_{ij}^{(1)} & \cdots & \ell_{iw}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(1)} & \cdots & \ell_{wj}^{(1)} & \cdots & \ell_{ww}^{(1)} \end{bmatrix} \cdots\cdots \begin{bmatrix} \ell_{11}^{(d)} & \cdots & \ell_{1j}^{(d)} & \cdots & \ell_{1w}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(d)} & \cdots & \ell_{ij}^{(d)} & \cdots & \ell_{iw}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(d)} & \cdots & \ell_{wj}^{(d)} & \cdots & \ell_{ww}^{(d)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Each $\ell_{i,j}^{(k)}$ is a linear polynomial over the variables $\{x_1, \ldots, x_n\}$.

Iterated Matrix Multiplication

$$\begin{bmatrix}1\\ \vdots\\ 1\\ \vdots\\ 1\end{bmatrix}^T \begin{bmatrix}\ell_{11}^{(1)} & \cdots & \ell_{1j}^{(1)} & \cdots & \ell_{1w}^{(1)}\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \ell_{i1}^{(1)} & \cdots & \ell_{ij}^{(1)} & \cdots & \ell_{iw}^{(1)}\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \ell_{w1}^{(1)} & \cdots & \ell_{wj}^{(1)} & \cdots & \ell_{ww}^{(1)}\end{bmatrix} \cdots\cdots \begin{bmatrix}\ell_{11}^{(d)} & \cdots & \ell_{1j}^{(d)} & \cdots & \ell_{1w}^{(d)}\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \ell_{i1}^{(d)} & \cdots & \ell_{ij}^{(d)} & \cdots & \ell_{iw}^{(d)}\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \ell_{w1}^{(d)} & \cdots & \ell_{wj}^{(d)} & \cdots & \ell_{ww}^{(d)}\end{bmatrix} \begin{bmatrix}1\\ \vdots\\ 1\\ \vdots\\ 1\end{bmatrix}$$

Each $\ell_{i,j}^{(k)}$ is a linear polynomial over the variables $\{x_1, \ldots, x_n\}$.

Canonical example of a polynomial that is computable by an ABP of width w and length d.

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Computable efficiently by ABPs.

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Symbolic Permanent

$$\mathrm{Perm}_n = \sum_{\sigma \in S_n} \prod_{i \in [n]} x_{i\sigma(i)}$$

Computable efficiently by ABPs.

Determinant and Permanent Polynomials

Symbolic Matrix :

$$\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix}$$

Symbolic Determinant

$$\mathrm{Det}_n = \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \cdot \prod_{i \in [n]} x_{i\sigma(i)}$$

Computable efficiently by ABPs.

Symbolic Permanent

$$\mathrm{Perm}_n = \sum_{\sigma \in S_n} \prod_{i \in [n]} x_{i\sigma(i)}$$

Complete for the class of all explicit polynomials.

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathrm{coeff}_{t^d} \left(\prod_{i \in [n]} (1 + t x_i) \right)$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathrm{coeff}_{t^d} \left(\prod_{i \in [n]} (1 + t x_i) \right)$$

$$\begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^j & \cdots & \alpha_0^n \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_i^0 & \cdots & \alpha_i^j & \cdots & \alpha_i^n \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_n^0 & \cdots & \alpha_n^j & \cdots & \alpha_n^n \end{bmatrix} \times \begin{bmatrix} \mathrm{coeff}_{t^0} \left(\prod_{i \in [n]} (1 + t x_i) \right) \\ \vdots \\ \mathrm{coeff}_{t^j} \left(\prod_{i \in [n]} (1 + t x_i) \right) \\ \vdots \\ \mathrm{coeff}_{t^n} \left(\prod_{i \in [n]} (1 + t x_i) \right) \end{bmatrix} = \begin{bmatrix} \prod_{i \in [n]} (1 + \alpha_0 x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_i x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_n x_i) \end{bmatrix}$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathsf{coeff}_{t^d} \left(\prod_{i \in [n]} (1 + t x_i) \right)$$

$$\begin{bmatrix} \mathsf{coeff}_{t^0} \left(\prod_{i \in [n]} (1 + t x_i) \right) \\ \vdots \\ \mathsf{coeff}_{t^j} \left(\prod_{i \in [n]} (1 + t x_i) \right) \\ \vdots \\ \mathsf{coeff}_{t^d} \left(\prod_{i \in [n]} (1 + t x_i) \right) \end{bmatrix} = \begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^j & \cdots & \alpha_0^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_i^0 & \cdots & \alpha_i^j & \cdots & \alpha_i^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_d^0 & \cdots & \alpha_d^j & \cdots & \alpha_d^d \end{bmatrix}^{-1} \times \begin{bmatrix} \prod_{i \in [n]} (1 + \alpha_0 x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_i x_i) \\ \vdots \\ \prod_{i \in [n]} (1 + \alpha_d x_i) \end{bmatrix}$$

Elementary Symmetric Polynomials

$$\mathrm{ESYM}_{n,d} = \sum_{S \subseteq [n]} \prod_{i \in S} x_i = \mathrm{coeff}_{t^d}\left(\prod_{i \in [n]} (1 + t x_i)\right)$$ is efficiently computable by $\Sigma\Pi\Sigma$ formulas.

$$\begin{bmatrix} \mathrm{coeff}_{t^0}\left(\prod_{i\in[n]}(1+tx_i)\right) \\ \vdots \\ \mathrm{coeff}_{t^j}\left(\prod_{i\in[n]}(1+tx_i)\right) \\ \vdots \\ \mathrm{coeff}_{t^d}\left(\prod_{i\in[n]}(1+tx_i)\right) \end{bmatrix} = \begin{bmatrix} \alpha_0^0 & \cdots & \alpha_0^j & \cdots & \alpha_0^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_i^0 & \cdots & \alpha_i^j & \cdots & \alpha_i^d \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_d^0 & \cdots & \alpha_d^j & \cdots & \alpha_d^d \end{bmatrix}^{-1} \times \begin{bmatrix} \prod_{i\in[n]}(1+\alpha_0 x_i) \\ \vdots \\ \prod_{i\in[n]}(1+\alpha_i x_i) \\ \vdots \\ \prod_{i\in[n]}(1+\alpha_d x_i) \end{bmatrix}$$

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.

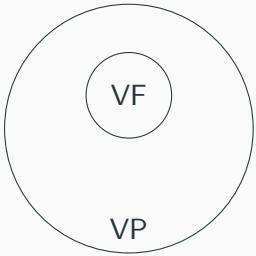
VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

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).



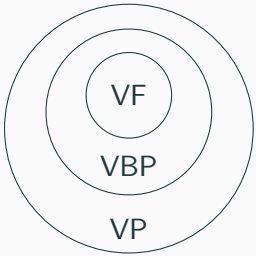
Lower Bounds in Algebraic Circuit Complexity

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

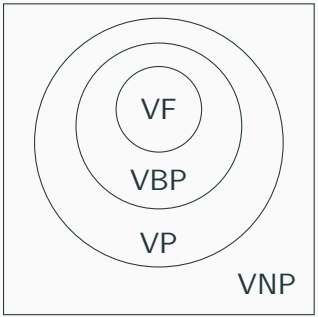
Objects of Study: Polynomials over n variables of degree d.

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

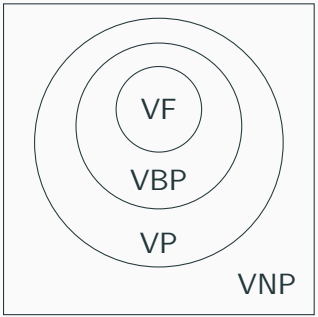
Objects of Study: Polynomials over n variables of degree d.

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



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

Lower Bounds in Algebraic Circuit Complexity

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

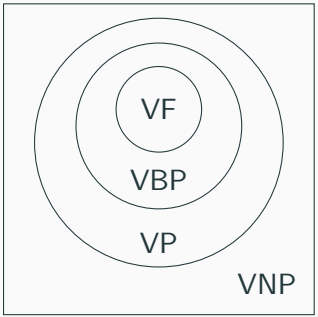
VF: Polynomials computable by formulas of size poly(n, d).

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

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

VNP: Explicit Polynomials

$$\text{VP} \stackrel{?}{=} \text{VNP}$$



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

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

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

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

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

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

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

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

[Shpilka-Yehudayoff 10] (using Kalorkoti's method): There is an n-variate multilinear polynomial such that any formula computing it requires $\Omega(n^2/\log n)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

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

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

[Shpilka-Yehudayoff 10] (using Kalorkoti's method): There is an n-variate multilinear polynomial such that any formula computing it requires $\Omega(n^2/\log n)$ wires.

[C-Kumar-She-Volk 22]: Any formula computing $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$ requires $\Omega(n^2)$ vertices.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial

that is

computable by a general model of size s.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial
that is
computable by a general model of size s.

implies

it is also computed by a structured model of size $\mathrm{func}(s, n, d)$ for some function func.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial that is computable by a general model of size s. implies it is also computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Step 2: **Study Structured Models**

Prove strong lower bounds against structured models computing f.

How does one make progress?

Step 1: **Structural Results**

Structured n-variate, degree-d polynomial that is computable by a general model of size s.

implies

it is also computed by a structured model of size $\text{func}(s, n, d)$ for some function func.

Step 2: **Study Structured Models**

Prove strong lower bounds against structured models computing f.

Ultimate Goal: Prove better than $\text{func}(s, n, d)$ lower bounds.

Super-Polynomial Lower Bound against Constant Dept Circuits

[Limaye-Srinivasan-Tavenas 21] (JACM 2025)

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

Super-Polynomial Lower Bound against Constant Dept Circuits

[Limaye-Srinivasan-Tavenas 21] (JACM 2025)

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

In particular, it shows that any depth-3 circuit computing $\mathrm{IMM}_{n,\frac{\log n}{100}}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega(\sqrt{d})}$.

Super-Polynomial Lower Bound against Constant Dept Circuits

[Limaye-Srinivasan-Tavenas 21] (JACM 2025)

Let d, n be such that $d \leq \frac{\log n}{100}$. For any $\Delta > 0$, any product-depth $\Delta \geq 1$ circuit computing $\mathrm{IMM}_{n,d}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega\left(d^{\frac{1}{\exp(\Delta)}}\right)}$.

In particular, it shows that any depth-3 circuit computing $\mathrm{IMM}_{n,\frac{\log n}{100}}$ over any field of characteristic zero or $\geq d$, requires size $n^{\Omega(\sqrt{d})}$.

[Forbes 24]: The theorem is true over all fields.

Set-Multilinearity

The variable set is divided into buckets.

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

Set-Multilinearity

The variable set is divided into buckets.

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

f is set-multilinear with respect to $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ if

every monomial in f has exactly one variable from $\mathbf{x}_i$ for each $i \in [d]$.

Proof Overview

Step 1: Convert any arbitrary product-depth Δ circuit into a homogeneous product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $2^{O(\sqrt{d})}\,\mathrm{poly}(s)$.

Proof Overview

Step 1: Convert any arbitrary product-depth Δ circuit into a homogeneous product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $2^{O(\sqrt{d})}\,\mathrm{poly}(s)$.

Step 2: Convert any homogeneous product-depth 2Δ circuit computing a set-multilinear polynomial into a set-multilinear product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $d^{O(d)}\,\mathrm{poly}(s)$.

Proof Overview

Step 1: Convert any arbitrary product-depth Δ circuit into a homogeneous product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $2^{O(\sqrt{d})}\,\mathrm{poly}(s)$.

Step 2: Convert any homogeneous product-depth 2Δ circuit computing a set-multilinear polynomial into a set-multilinear product-depth 2Δ circuit computing the same polynomial. Blow-up in size is $d^{O(d)}\,\mathrm{poly}(s)$.

Step 3: Prove a $n^{\Omega\left(\frac{d^{1/2^{\Delta}}-1}{\Delta}\right)}$ lower bound against set-multilinear constant depth circuits.

Questions?

Algebraic Branching Programs

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}^T \begin{bmatrix} \ell_{11}^{(1)} & \cdots & \ell_{1j}^{(1)} & \cdots & \ell_{1w}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(1)} & \cdots & \ell_{ij}^{(1)} & \cdots & \ell_{iw}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(1)} & \cdots & \ell_{wj}^{(1)} & \cdots & \ell_{ww}^{(1)} \end{bmatrix} \cdots\cdots \begin{bmatrix} \ell_{11}^{(d)} & \cdots & \ell_{1j}^{(d)} & \cdots & \ell_{1w}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(d)} & \cdots & \ell_{ij}^{(d)} & \cdots & \ell_{iw}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(d)} & \cdots & \ell_{wj}^{(d)} & \cdots & \ell_{ww}^{(d)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Each $\ell_{i,j}^{(k)}$ is a linear polynomial over the variables $\{x_1, \ldots, x_n\}$.

Algebraic Branching Programs

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}^T \begin{bmatrix} \ell_{11}^{(1)} & \cdots & \ell_{1j}^{(1)} & \cdots & \ell_{1w}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(1)} & \cdots & \ell_{ij}^{(1)} & \cdots & \ell_{iw}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(1)} & \cdots & \ell_{wj}^{(1)} & \cdots & \ell_{ww}^{(1)} \end{bmatrix} \cdots\cdots \begin{bmatrix} \ell_{11}^{(d)} & \cdots & \ell_{1j}^{(d)} & \cdots & \ell_{1w}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{i1}^{(d)} & \cdots & \ell_{ij}^{(d)} & \cdots & \ell_{iw}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \ell_{w1}^{(d)} & \cdots & \ell_{wj}^{(d)} & \cdots & \ell_{ww}^{(d)} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Each $\ell_{i,j}^{(k)}$ is a linear polynomial over the variables $\{x_1, \ldots, x_n\}$.

What is the smallest w for which the polynomial of interest can be computed as above?

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

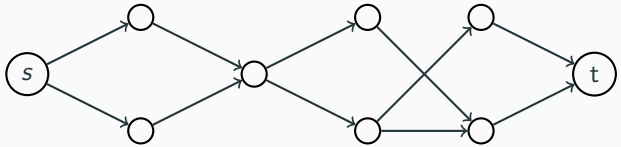
[Baur-Strassen]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

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

Our Result

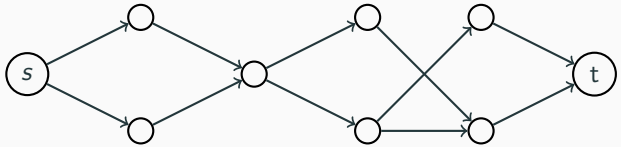
[C-Kumar-She-Volk]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

At most d matrices: Proof Overview



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

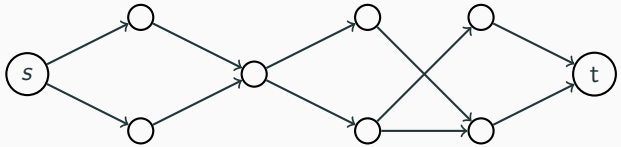
At most d matrices: Proof Overview



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

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

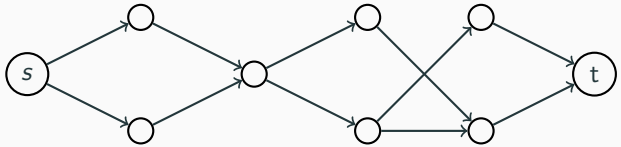
At most d matrices: Proof Overview



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

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

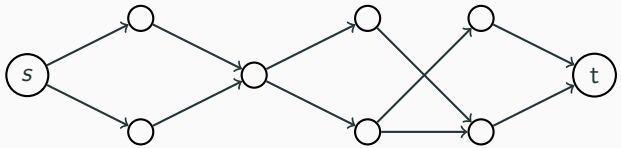
At most d matrices: Proof Overview



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

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

At most d matrices: Proof Overview

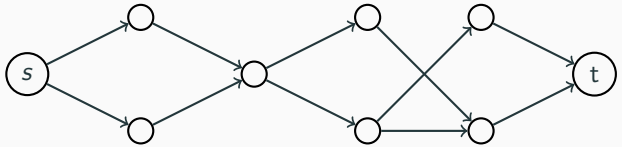


$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

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

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

At most d matrices: Proof Overview

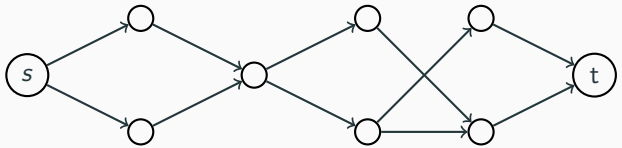


$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

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

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

At most d matrices: Proof Overview

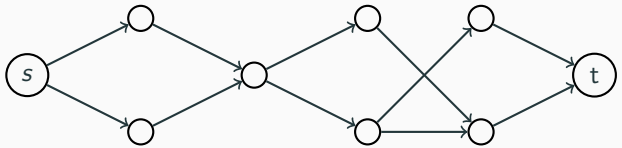


$P_{n,d}(\mathbf{x}) = \sum_{i=1}^{n} x_i^d$.

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

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

At most d matrices: 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 $\deg(R) \leq d-1$.

At most d matrices: Proof Overview (contd.)

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

At most d matrices: 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.

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

At most d matrices: 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.

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

At most d matrices: 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'

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

$R = 0$

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

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

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

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

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

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

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

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

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

At most d matrices: 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 $\left\{g_j', h_j'\right\}_{j\in[t_k]}$

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

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

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

At most d matrices: 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: $\mathcal{A}_{\ell+1}$

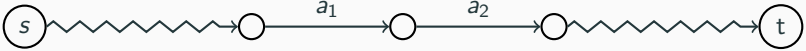
Size $= s_{\ell+1} \leq s_\ell$
Number of layers $= d_{\ell+1} \leq \frac{2}{3} d_\ell$
Number of error terms $= r_{\ell+1} \leq r_\ell + \frac{s_\ell}{d_\ell/3}$

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

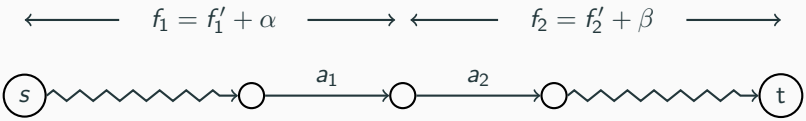
Number of Error Terms: $\varepsilon \cdot n$ where ε can be chosen.

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

Proof of the Induction Step

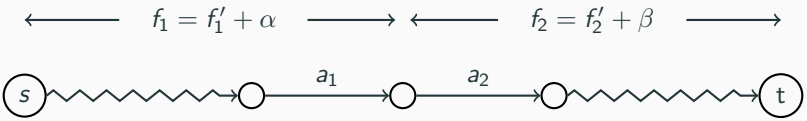


Proof of the Induction Step



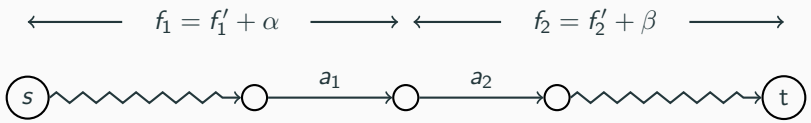
Proof of the Induction Step

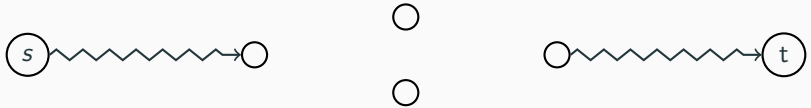
$\mathcal{A}_\ell = f_1 \cdot f_2$



Proof of the Induction Step

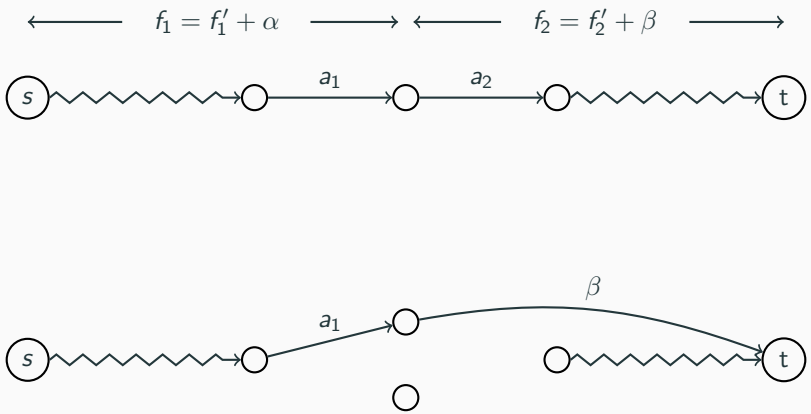
$\mathcal{A}_\ell = f_1 \cdot f_2$





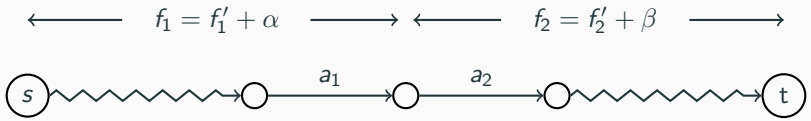
Proof of the Induction Step

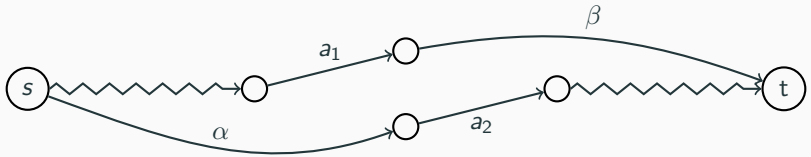
$\mathcal{A}_\ell = f_1 \cdot f_2$



Proof of the Induction Step

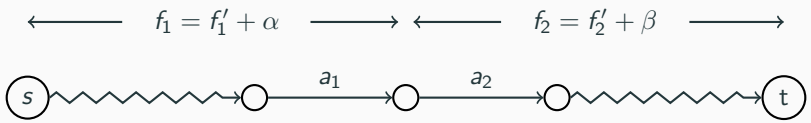
$\mathcal{A}_\ell = f_1 \cdot f_2$



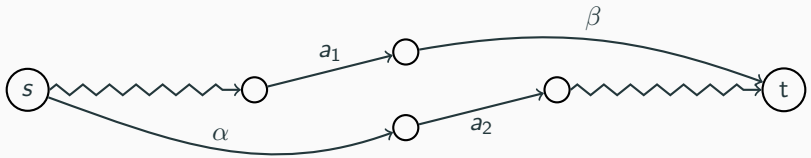


Proof of the Induction Step

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

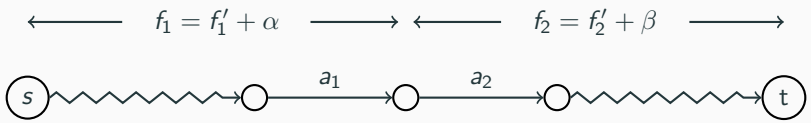


$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$

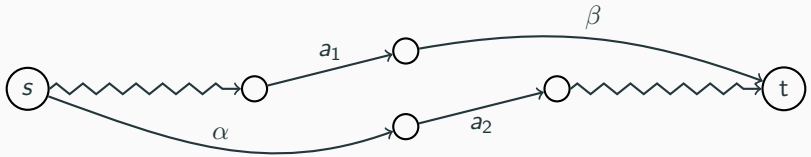


Proof of the Induction Step

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

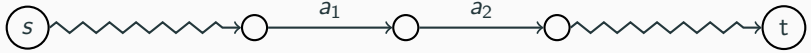


$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2 = \mathcal{A}_\ell - f_1' \cdot f_2' + \alpha \cdot \beta$

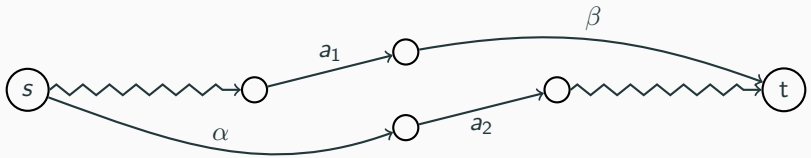


Proof of the Induction Step

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

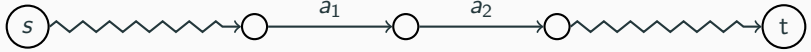


$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$



Proof of the Induction Step

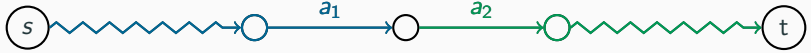
$\mathcal{A}_\ell = f_1 \cdot f_2$



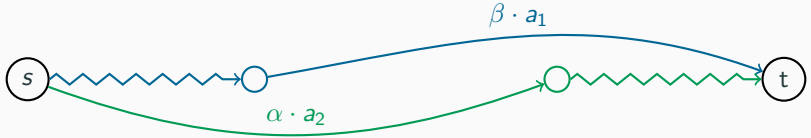
$\mathcal{A}_{\ell+1} = \beta \cdot f_1 + \alpha \cdot f_2$

Proof of the Induction Step

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



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



Thank you!!!