Algebraic complexity

EPIT 2023 : Le Kaléidoscope de la Complexité

Guillaume Malod

June 12–16 2023 Oléron Island (France)

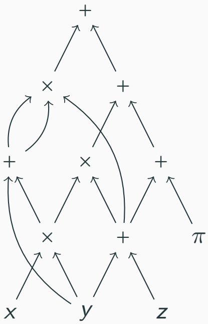
Outline

Outline

Representing multivariate polynomials

- Dense representation
- Sparse representation
- Arithmetic formulas: $(x_1 + y_1) \times \cdots \times (x_n + y_n)$
- Arithmetic circuits

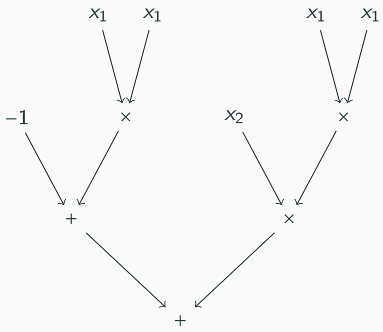
Arithmetic circuits



- Size of a circuit: number of gates or edges...
- Arithmetic circuit of size 12 computing
  $(xy+y)(xy+y)+(xy)(y+z)((y+z)+\pi)$
- Depth: length of a longest path from root to leaf

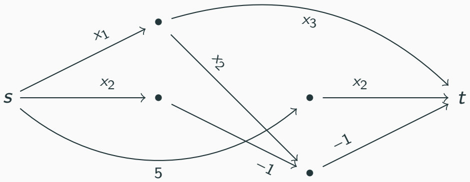
Arithmetic formulas

- Weak model: each subcomputation can be used only once.
- Underlying graph = tree.



Algebraic Branching Program (ABP)

- DAG from a source s to a sink t with arcs labelled by constants or variables.

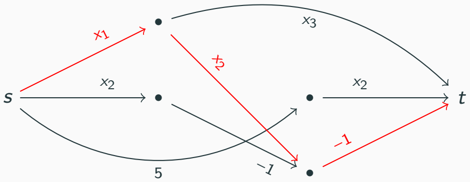


- Weight of a path = product of the labels.

- Polynomial computed by the ABP = sum of the weights of all paths from s to t.

Algebraic Branching Program (ABP)

- DAG from a source s
  to a sink t
  with arcs labelled by
  constants or variables.



- Weight of a path = product of the labels.

- Polynomial computed by the ABP =
  sum of the weights of all paths from s to t.

Important examples

$$\bar{z} = (z_{i,j})_{1 \le i,j \le n}$$

$$\det(\bar{z}) = \sum_{\sigma \in S_n} \epsilon(\sigma) \prod_{i=1}^{n} z_{i,\sigma(i)}$$

$$\operatorname{per}(\bar{z}) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} z_{i,\sigma(i)}$$

$$\operatorname{hc}(\bar{z}) = \sum_{\substack{\sigma \in S_n \\ \sigma \text{ is a cycle}}} \prod_{i=1}^{n} z_{i,\sigma(i)}$$

Small classes

- Only consider sequences of polynomials with polynomially bounded degree
- A sequence of polynomials $(f_n) \rightarrow$ existence of a "small" sequence (C_n) such that C_n computes f_n
- VP: sequences computable by a sequence of circuits of polynomially bounded size
- $\mathrm{VP_e}$: sequences computable by a sequence of formulas of polynomially bounded size
- VBP: sequences computable by a sequence of ABPs of polynomially bounded size
- $\mathrm{VP_e} \subseteq \mathrm{VBP} \subseteq \mathrm{VP}$

Big classes

- VNP: $(f_n) \in \mathrm{VNP}$ if $\exists (g_n) \in \mathrm{VP}$:

$$f_n(\bar{z}) = \sum_{\epsilon \in \{0,1\}^{q(n)}} g_n(\bar{z}, \epsilon)$$

- For the permanent:

$$per(\bar{z}) = \sum_{\bar{\epsilon} \in \{0,1\}^{n^2}} \mathrm{test}(\bar{\epsilon}) \cdot \prod_{i=1}^{n} \left(\sum_{j=1}^{n} \epsilon_{i,j} z_{i,j} \right)$$

Big classes

- VNP: $(f_n) \in \mathrm{VNP}$ if $\exists (g_n) \in \mathrm{VP}$:

$$f_n(\bar{z}) = \sum_{\epsilon \in \{0,1\}^{q(n)}} g_n(\bar{z}, \epsilon)$$

- For the permanent:

$$per(\bar{z}) = \sum_{\bar{\epsilon} \in \{0,1\}^{n^2}} \left(\prod_{\substack{1 \le i,j,k,l \le n \\ i=k \text{ iff } j \neq l}} (1 - \epsilon_{i,j}\epsilon_{k,l}) \right) \cdot \left(\prod_{i=1}^{n} \sum_{j=1}^{n} \epsilon_{i,j} \right) \cdot \prod_{i=1}^{n} \left(\sum_{j=1}^{n} \epsilon_{i,j} z_{i,j} \right)$$

Big classes

- VNP: $(f_n) \in \mathrm{VNP}$ if $\exists (g_n) \in \mathrm{VP}$:

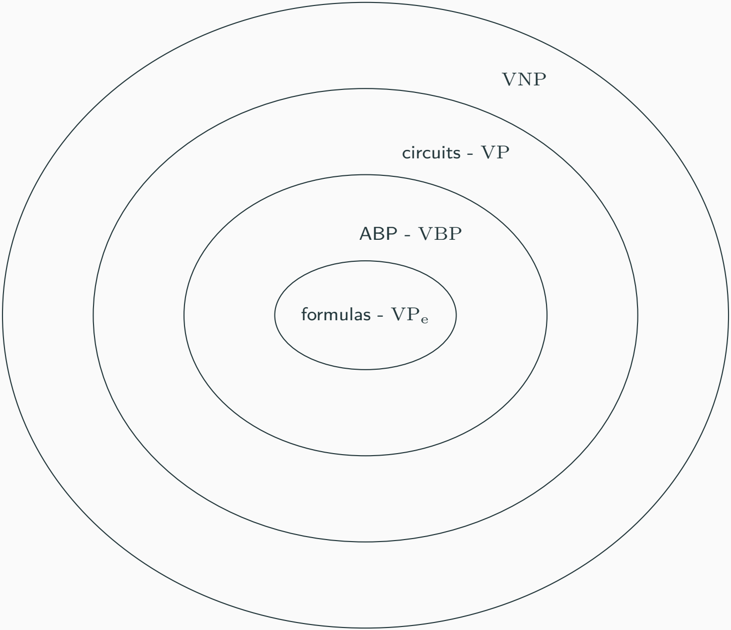
$$f_n(\bar{z}) = \sum_{\epsilon \in \{0,1\}^{q(n)}} g_n(\bar{z}, \epsilon)$$

- For the permanent:

$$per(\bar{z}) = \sum_{\bar{\epsilon} \in \{0,1\}^{n^2}} \left(\prod_{\substack{1 \le i,j,k,l \le n \\ i=k \text{ iff } j \ne l}} (1 - \epsilon_{i,j}\epsilon_{k,l}) \right) \cdot \left(\prod_{i=1}^{n} \sum_{j=1}^{n} \epsilon_{i,j} \right) \cdot \prod_{i=1}^{n} \left(\sum_{j=1}^{n} \epsilon_{i,j} z_{i,j} \right)$$

- Intuitively, all polynomials where the coefficient function is in GapP/poly
- Exercise: show that hc $\in \mathrm{VNP}$
- Bonus exercise: use dynamic programming to give an $O(n2^n)$ circuit for per; compare with Wikipedia (Ryser)

Classes



VP **versus** VNP

- Main open question: $\mathrm{VP} =? \mathrm{VNP}$
- Somewhat related to P =? NP

Theorem (P. Bürgisser)

Under (GRH), $\mathrm{VP} = \mathrm{VNP}$ *over* $\mathbb{C}$ *implies* $\mathrm{P/poly} = \mathrm{NP/poly}$.

- per is VNP-complete over fields of characteristic $\neq 2$
- hc is VNP-complete
- det is VBP-complete
- VBP vs VNP becomes det vs per

Reductions

- A polynomial f is a *projection* of a polynomial g if $f(\bar{x}) = g(a_1, \ldots, a_m)$, where the a_i are elements of the field or variables among $x_1, \ldots, x_n$
- A sequence (f_n) is a *p-projection* of a sequence (g_n) if there exists a polynomially bounded function $t(n)$ such that f_n is a projection of $g_{t(n)}$ for all n
- A sequence of polynomials $(f_n) \in \mathcal{C}$ is *$\mathcal{C}$-complete* if any sequence of polynomials $(g_n) \in \mathcal{C}$ is a p-projection of (f_n)

Valiant's theorem

Theorem

The sequence (per_n) *is* VNP*-complete over any field of characteristic different from* 2*.*

Corollary

Over any field of characteristic different from 2*,* $\mathrm{VP} = \mathrm{VNP}$ *iff* $\mathrm{per} \in \mathrm{VP}$*.*

Outline

Completeness proof strategy

1. $\mathrm{VNP_e} = \mathrm{VNP}$
2. The permanent is universal for formulas
3. The permanent can “eliminate” boolean sums

Outline

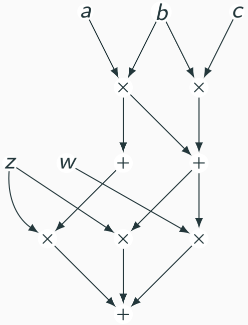
Classes defined via formulas

- $(f_n) \in \mathrm{VP_e}$ if there exists a sequence of formulas (F_n) of polynomially bounded size such that F_n computes f_n.
- $(f_n) \in \mathrm{VNP_e}$ if there exists a polynomial p and a sequence $g_n \in \mathrm{VP_e}$ such that:

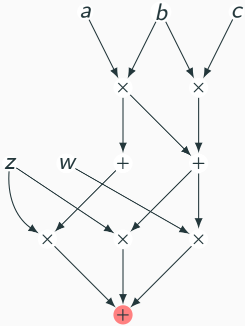
$$f_n(\bar{x}) = \sum_{\bar{\epsilon} \in \{0,1\}^{p(|\bar{x}|)}} g_n(\bar{x}, \bar{\epsilon}).$$

- $\mathrm{VP_e} \subseteq \mathrm{VP}$ and $\mathrm{VNP_e} \subseteq \mathrm{VNP}$
- Whether $\mathrm{VP_e} = \mathrm{VP}$ or not is still open
- Valiant showed that $\mathrm{VNP_e} = \mathrm{VNP}$
- Is it enough to show that $\mathrm{VP} \subseteq \mathrm{VNP_e}$
- Reduction of CircuitSAT to SAT

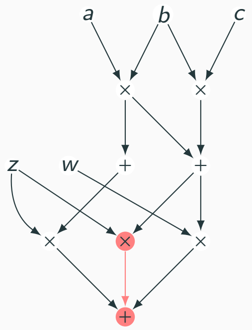
Parse trees



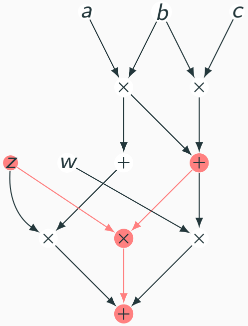
Parse trees



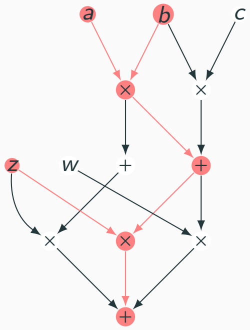
Parse trees



Parse trees



Parse trees



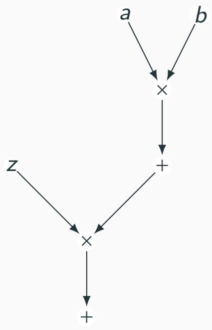


Figure 1: $val(T) = zab$

- Each parse tree computes a monomial.
- The polynomial $f(z)$ computed by the circuit is $\sum_T \text{val}(T)$
- $f(z) = \sum_{\bar{\epsilon} \in \{0,1\}^s} \text{test}(\bar{\epsilon}) \text{val}'(\epsilon, z)$

Outline

The permanent

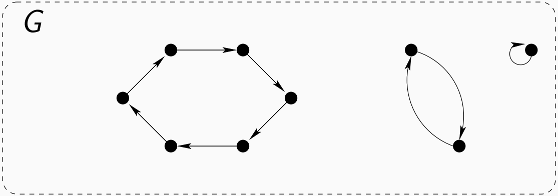
- If G is a bipartite graph, the permanent of its adjacency matrix counts the number of perfect matchings of G
- If G is a directed graph with a weight function on the edges, the permanent of its adjacency matrix is the sum of the weight of the cycle covers of G

The permanent

- If G is a bipartite graph, the permanent of its adjacency matrix counts the number of perfect matchings of G
- If G is a directed graph with a weight function on the edges, the permanent of its adjacency matrix is the sum of the weight of the cycle covers of G

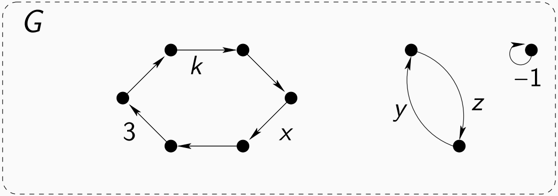
The permanent

- If G is a bipartite graph, the permanent of its adjacency matrix counts the number of perfect matchings of G
- If G is a directed graph with a weight function on the edges, the permanent of its adjacency matrix is the sum of the weight of the cycle covers of G



The permanent

- If G is a bipartite graph, the permanent of its adjacency matrix counts the number of perfect matchings of G
- If G is a directed graph with a weight function on the edges, the permanent of its adjacency matrix is the sum of the weight of the cycle covers of G



$VP_e \subseteq VBP$

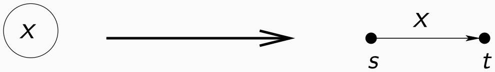
Lemma

If f is a polynomial computed by a formula of size e, then there exists an ABP G of size $e+1$ computing f.

Lemma

If f is a polynomial computed by a formula of size e, then there exists an ABP G of size $e+1$ computing f.

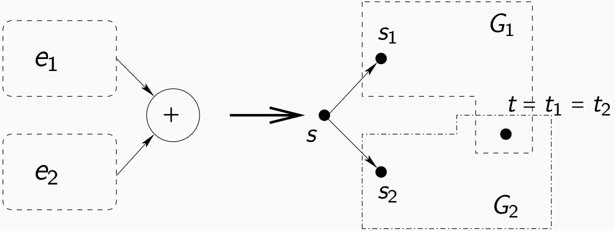
$e = 1$:



Lemma

If f is a polynomial computed by a formula of size e, then there exists an ABP G of size $e+1$ computing f.

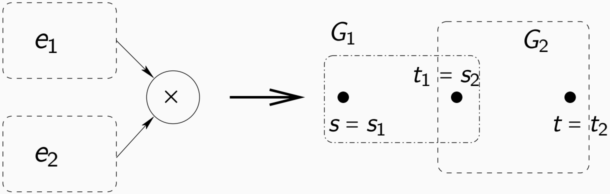
$e = e_1 + e_2$:



Lemma

If f is a polynomial computed by a formula of size e, then there exists an ABP G of size $e+1$ computing f.

$e = e_1 \times e_2$:



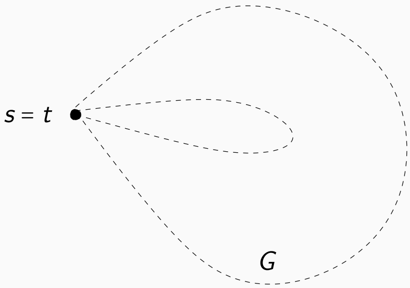
The permanent is universal for ABPs

Lemma

If f is a polynomial computed by a formula of size e, then there exists an $e \times e$ matrix M such that $f = \mathrm{per}(M)$.

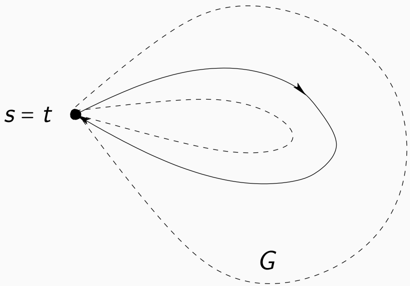
Lemma

If f is a polynomial computed by a formula of size e, then there exists an $e \times e$ matrix M such that $f = \mathrm{per}(M)$.



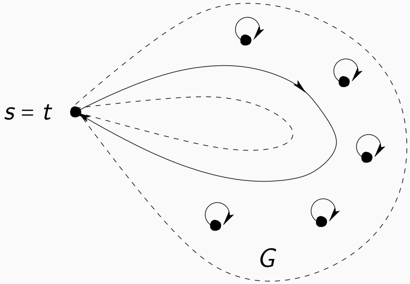
Lemma

If f is a polynomial computed by a formula of size e, then there exists an $e \times e$ matrix M such that $f = \mathrm{per}(M)$.



Lemma

If f is a polynomial computed by a formula of size e, then there exists an $e \times e$ matrix M such that $f = \mathrm{per}(M)$.



Outline

Eliminating sums

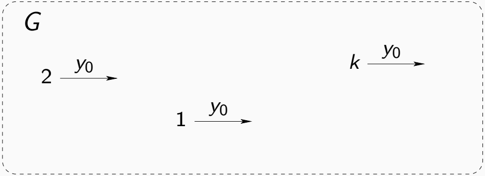
- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$

Eliminating sums

- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0

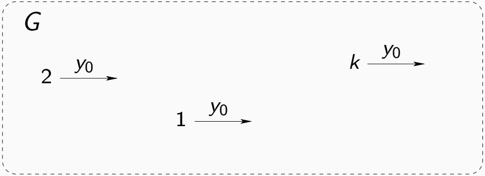
Eliminating sums

- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0
- $g_n(\bar{x}, y_0)$ is a permanent, so it is the weight of the cycle covers of a graph G



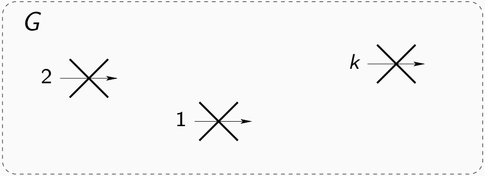
Eliminating sums

- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0
- $g_n(\bar{x}, y_0)$ is a permanent, so it is the weight of the cycle covers of a graph G
- We wish to compute $g_n(\bar{x}, 0) + g_n(\bar{x}, 1)$ as a permanent



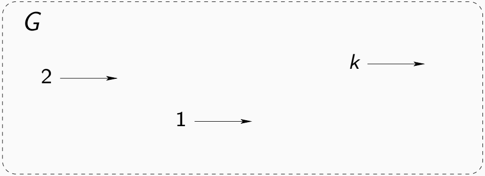
Eliminating sums

- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0
- $g_n(\bar{x}, y_0)$ is a permanent, so it is the weight of the cycle covers of a graph G
- We wish to compute $g_n(\bar{x}, 0) + g_n(\bar{x}, 1)$ as a permanent
- $g_n(\bar{x}, 0)$ is the sum of the weights of the cycle covers which *do not* use any of the edges



Eliminating sums

- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$,
  with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0
- $g_n(\bar{x}, y_0)$ is a permanent, so it is the weight of the cycle covers of a graph G
- We wish to compute $g_n(\bar{x}, 0) + g_n(\bar{x}, 1)$ as a permanent
- $g_n(\bar{x}, 0)$ is the sum of the weights of the cycle covers which *do not* use any of the edges
- For each subset $S \subseteq \{1, \ldots, k\}$, let W_S be the weight of the cycle covers using exactly the edges numbered in S
  Then: $g_n(\bar{x}, 1) = \sum_{S \subseteq \{1, \cdots, k\}} W_S$



Eliminating sums

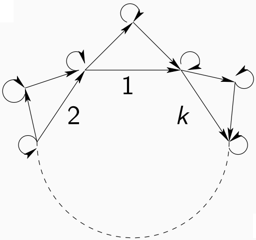
- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0
- $g_n(\bar{x}, y_0)$ is a permanent, so it is the weight of the cycle covers of a graph G
- We wish to compute $g_n(\bar{x}, 0) + g_n(\bar{x}, 1)$ as a permanent
- $g_n(\bar{x}, 0)$ is the sum of the weights of the cycle covers which *do not* use any of the edges
- For each subset $S \subseteq \{1, \ldots, k\}$, let W_S be the weight of the cycle covers using exactly the edges numbered in S
  Then: $g_n(\bar{x}, 1) = \sum_{S \subseteq \{1, \cdots, k\}} W_S$
- With this notation, $g_n(\bar{x}, 0)$ is $W_\varnothing$

Eliminating sums

- Suppose $(f_n) \in \mathrm{VNP}$, then $f_n(\bar{x}) = \sum_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$, with $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP_e}$
- Suppose there is only one variable y_0
- $g_n(\bar{x}, y_0)$ is a permanent, so it is the weight of the cycle covers of a graph G
- We wish to compute $g_n(\bar{x}, 0) + g_n(\bar{x}, 1)$ as a permanent
- $g_n(\bar{x}, 0)$ is the sum of the weights of the cycle covers which *do not* use any of the edges
- For each subset $S \subseteq \{1, \ldots, k\}$, let W_S be the weight of the cycle covers using exactly the edges numbered in S
  Then: $g_n(\bar{x}, 1) = \sum_{S \subseteq \{1, \cdots, k\}} W_S$
- With this notation, $g_n(\bar{x}, 0)$ is $W_\emptyset$
- And $g_n(\bar{x}, 0) + g_n(\bar{x}, 1) = 2W_\emptyset + \sum_{\substack{S \subseteq \{1, \cdots, k\} \\ S \neq \emptyset}} W_S$

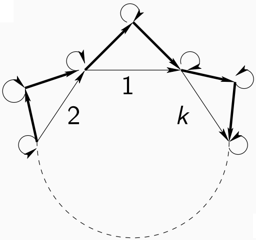
The Rosette gadget

- A directed graph with $2k$ vertices, $3k$ edges and $2k$ loops

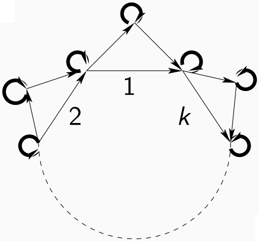


The Rosette gadget

- A directed graph with $2k$ vertices, $3k$ edges and $2k$ loops
- There are exactly two cycle covers which do not go through any of the edges $1, 2, \ldots, k$

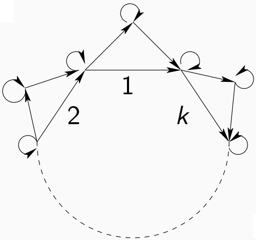


- A directed graph with $2k$ vertices, $3k$ edges and $2k$ loops
- There are exactly two cycle covers which do not go through any of the edges $1, 2, \ldots, k$

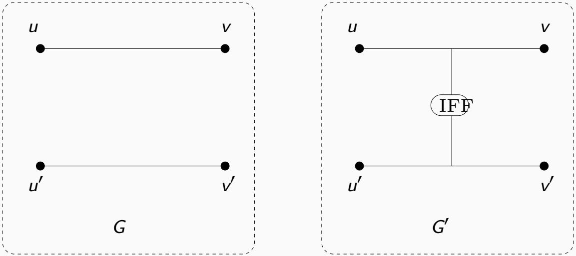


The Rosette gadget

- A directed graph with $2k$ vertices, $3k$ edges and $2k$ loops
- There are exactly two cycle covers which do not go through any of the edges $1, 2, \ldots, k$
- For each non-empty subset of $\{1, \ldots, k\}$ there is exactly one cycle cover which goes through exactly the specified edges



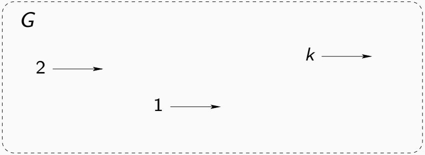
The IFF gadget

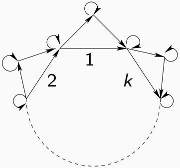


Lemma

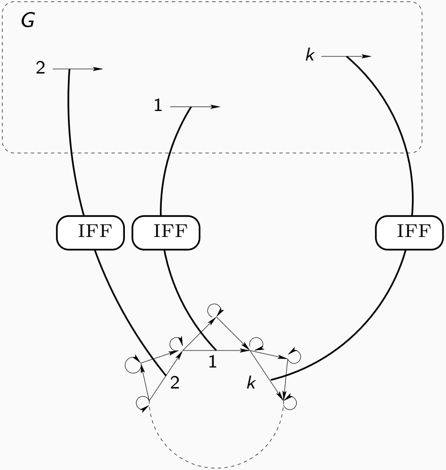
The permanent of G' is the sum of the weights of all cycle covers of G which contain both edges (u, v) and (u', v') or neither.

Bringing everything together

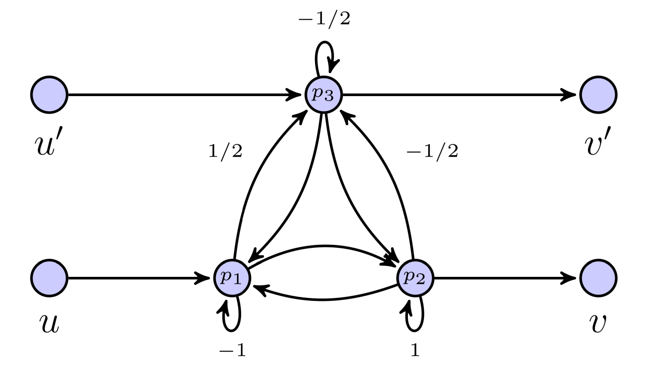




Bringing everything together



The IFF gadget

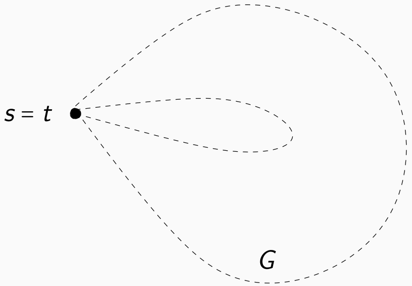


Source: Ramprasad Saptharishi

Outline

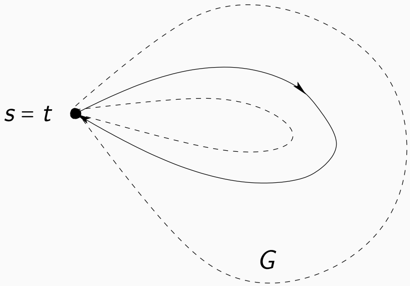
Universality for ABPs

- $\det(\bar{z}) = \sum_{\sigma \in S_n} \epsilon(\sigma)\ z_{1,\sigma(1)} \cdots z_{n,\sigma(n)}$
- Similar to the permanent: $\det(\bar{z}) = \sum_{\mathcal{C} \text{ a cycle cover}} \text{sign}(\mathcal{C}) \text{ weight}(\mathcal{C})$



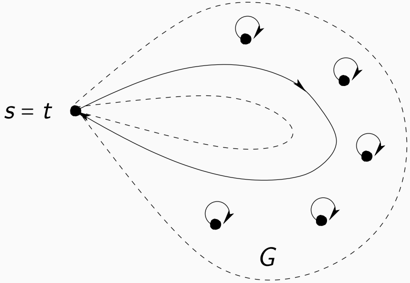
Universality for ABPs

- $\det(\bar{z}) = \sum_{\sigma \in S_n} \epsilon(\sigma)\ z_{1,\sigma(1)} \cdots z_{n,\sigma(n)}$
- Similar to the permanent: $\det(\bar{z}) = \sum_{\mathcal{C} \text{ a cycle cover}} \text{sign}(\mathcal{C})\ \text{weight}(\mathcal{C})$



Universality for ABPs

- $\det(\bar{z}) = \sum_{\sigma \in S_n} \epsilon(\sigma)\ z_{1,\sigma(1)} \cdots z_{n,\sigma(n)}$
- Similar to the permanent: $\det(\bar{z}) = \sum_{\mathcal{C} \text{ a cycle cover}} \text{sign}(\mathcal{C})\ \text{weight}(\mathcal{C})$
- Sign of a permutation decomposed in k cycles: $(-1)^{n+k}$
- Sign of a permutation with one main cycle of length p: $(-1)^{n\ +\ 1+(n-p)} = (-1)^{p-1}$
- Multiplicative sign coming from the -1 loops: $(-1)^{n-p}$
- Overall sign: $(-1)^{n-1}$



Computing the determinant

- Gaussian elimination
- Dynamic computation: too much information to keep track of, exponential size
- A cycle cannot loop before coming back to the first vertex
- Two cycles cannot have a common vertex

CLOW sequences

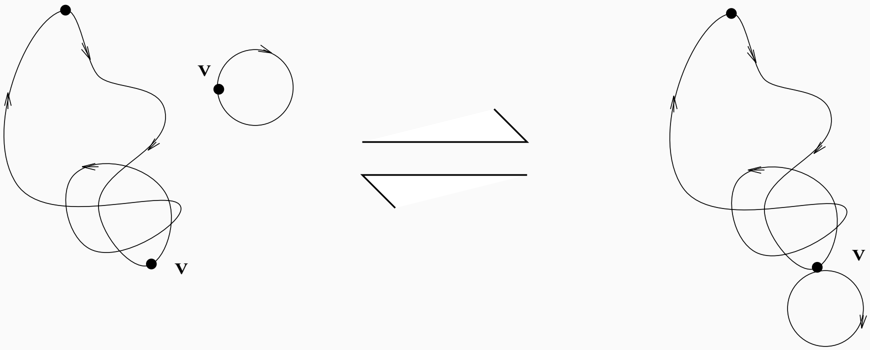
- A closed walk (CLOW) of length i is a sequence of vertices $c_1, c_2, \dots, c_i, c_1$ (generalization of a cycle)
- Its weight is the product of the weight of the edges
- A CLOW-sequence is a sequence $C_1, \dots, C_k$ of closed walks (generalization of a cycle cover)
- Its length is the sum of the lengths of the C_i
- Its weight is the product of the weights of the C_i
- Its sign is $(-1)^{n+k}$
- We know that: $\det(\bar{z}) = \sum_{\mathcal{C} \text{ a cycle cover}} \text{sign}(\mathcal{C}) \text{ weight}(\mathcal{C})$
- We will show that: $\det(\bar{z}) = \sum_{\substack{\mathcal{P} \text{ a CLOW sequence} \\ \text{of length } n}} \text{sign}(\mathcal{P}) \text{ weight}(\mathcal{P})$

Building an involution φ

- φ is the identity on cycle covers
- $\text{weight}(\varphi(P)) = \text{weight}(P)$ and $\text{sign}(\varphi(P)) = -\text{sign}(P)$, for a CLOW sequence $\mathcal{P}$ which is not a cycle cover

Building an involution φ

- φ is the identity on cycle covers
- weight($\varphi(P)$) = weight(P) and sign($\varphi(P)$) = −sign(P), for a CLOW sequence $\mathcal{P}$ which is not a cycle cover



Source: Mahajan & Vinay

Computing the determinant

- Compute the sum of the weights of the CLOW sequences of the complete directed graph
- $[l, c, c_0, s]$: sum of the weights of all partial CLOW sequences of length l, with current vertex c, with current CLOW starting point c_0 and with parity of the number of current completed CLOWs s.
- Build a graph with $2n^3$ vertices ($1 \leq l, c, c_0 \leq n$, $s \in \{-1.1\}$): one for each tuple $[l, c, c_0, s]$.
- Vertex $[l, c, c_0, s]$ sends an edge to vertex $[l+1, c', c_0, s]$, with weight $z_{cc'}$
- Vertex $[l, c, c_0, s]$ sends an edge to vertex $[l+1, c_0', c_0', -s]$ with weight z_{cc_0}
- Add a starting vertex, an end vertex, and relevant edges including for the sign

Slogan

- $\mathrm{VBP} = \mathrm{VNP}$ iff the permanent polynomial can be written as the determinant of a matrix of polynomially bounded size

Outline

Cancellations

- Cancellations are useful
- How much?
- Is it useful to produce non-multilinear monomials when computing a multilinear polynomial?
- Is it useful to compute higher-degree monomials and then cancel them out?
- Is is useful to produce non-homogeneous polynomials when computing an homogeneous polynomial?
- Answer may depend on the computation model (formula, ABP, circuit)

Outline

Homogenization of circuits

- A circuit C is said to be *homogeneous* if every gate in the circuit computes a homogeneous polynomial

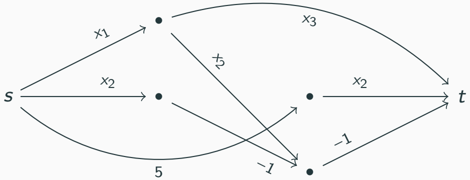
Lemma

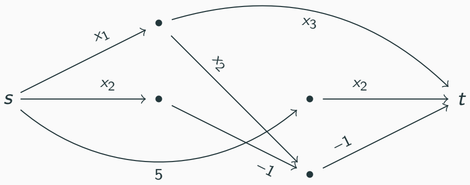
Let f be an n-variate degree d polynomial computed by a circuit C of size s. Then there is a homogeneous arithmetic circuit C', of size at most $O(sd^2)$, that computes the homogeneous components of f

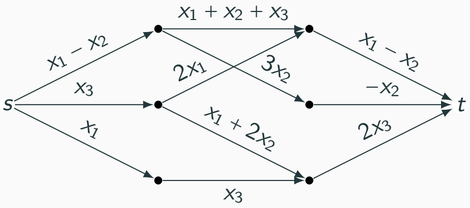
- For every gate $g \in C$, define $(d+1)$ gates $g^{(0)}, \ldots, g^{(d)}$
- We will build a new circuit C' such that $g^{(i)}$ computes the degree i homogeneous component of the polynomial computed at g.
- If a gate g has children h_1 and h_2 in C, then C' has the following connections depending on the type of g:

$$\begin{array}{lclcll} g = h_1 + h_2 & \implies & g^{(i)} & = & h_1^{(i)} + h_2^{(i)} & \text{for all } i \\ g = h_1 \times h_2 & \implies & g^{(i)} & = & \sum_{j=0}^{i} h_1^{(j)} h_2^{(i-j)} & \text{for all } i \end{array}$$

Homogenization of ABPs

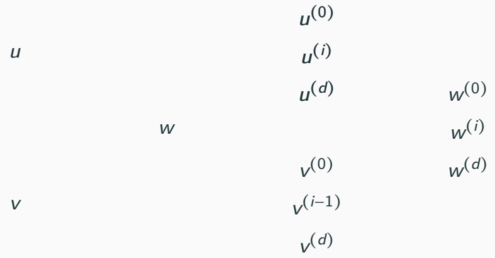



s
t
x1
x2
5
x3
x2
x2
−1
−1


s
t
x1 − x2
x3
x1
x1 + x2 + x3
2x1
3x2
x1 + 2x2
x3
x1 − x2
−x2
2x3

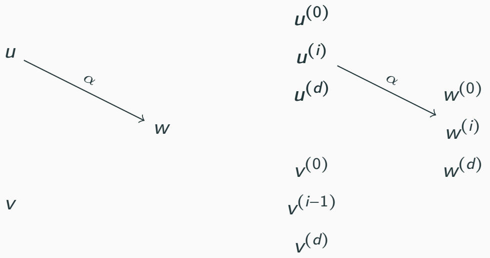
Homogenization of ABPs

- Similar idea applied to an ABP A
- Create a new ABP A' with vertices $u^{(0)}, \ldots, u^{(d)}$: $u^{(i)}$ will compute the homogeneous component of degree i of the polynomial computed at u in A
- Connect the new vertices by induction, starting from the source s



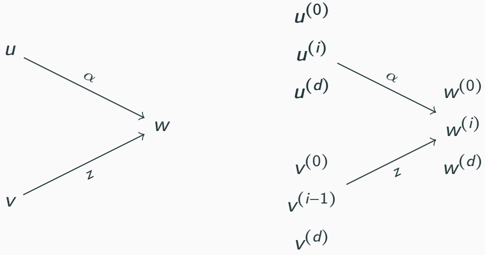
Homogenization of ABPs

- Similar idea applied to an ABP A
- Create a new ABP A' with vertices $u^{(0)}, \ldots, u^{(d)}$: $u^{(i)}$ will compute the homogeneous component of degree i of the polynomial computed at u in A
- Connect the new vertices by induction, starting from the source s
- If a vertex w receives an edge labeled with a constant α from the vertex u then $w^{(i)}$ must receive an edge labeled with α from the vertex $u^{(i)}$



Homogenization of ABPs

- Similar idea applied to an ABP A
- Create a new ABP A' with vertices $u^{(0)}, \dots, u^{(d)}$: $u^{(i)}$ will compute the homogeneous component of degree i of the polynomial computed at u in A
- Connect the new vertices by induction, starting from the source s
- If a vertex w receives an edge labeled with a constant α from the vertex u then $w^{(i)}$ must receive an edge labeled with α from the vertex $u^{(i)}$
- If a vertex w receives an edge labeled with a variable z from the vertex v then $w^{(i)}$ must receive an edge labeled with z from the vertex $u^{(i-1)}$

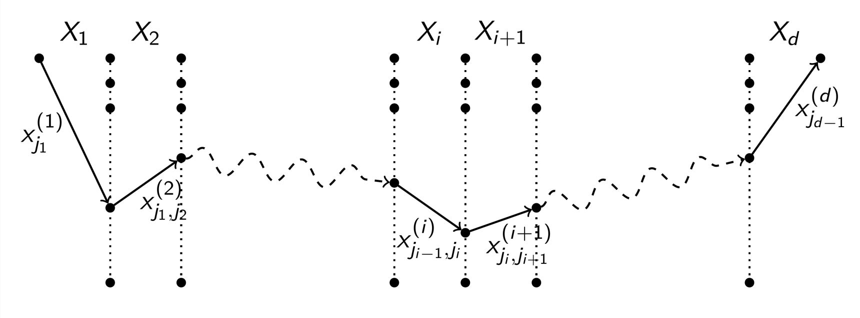


Homogenization of ABPs

- Arrange the vertices by "level": all vertices of the form w^i are on level i
- Edges can be inside a level or from one level to the next
- All the paths must visit at least one vertex by level
- The sum of the weights of the paths from a vertex in level i to a vertex in level $i+1$ is a linear form
- Delete all edges and add edges between levels, with linear forms

ABPs and iterated matrix multiplication

$$\mathrm{IMM}_{n,d} = \sum_{1 \le j_1, \ldots, j_{d-1} \le n} x^{(1)}_{j_1} x^{(2)}_{j_1,j_2} x^{(3)}_{j_2,j_3} \cdots x^{(d-1)}_{j_{d-2},j_{d-1}} x^{(d)}_{j_{d-1}}$$



Homogenization of formulas: interpolation

- If a circuit C computes a polynomial $P(x_1, \ldots, x_n, y)$ with $\deg_y P = d$
- $P(x_1, \ldots, x_n, y) = P_0(x_1, \ldots, x_n) + P_1(x_1, \ldots, x_n)y + \cdots + P_d(x_1, \ldots, x_n)y^d$
- Fix distinct scalars $\alpha_0, \ldots, \alpha_d \in \mathbb{F}$
- Each of the $P_i(x_1, \ldots, x_n)$ can be expressed as a linear combination of $\{P(x_1, \ldots, x_n, \alpha_0), \ldots, P(x_1, \ldots, x_n, \alpha_d)\}$
- If P is computable by a size s circuit from some class $\mathcal{C}$, then each P_i is computable by a size $O(sd)$ circuit from the class $\Sigma\mathcal{C}$
- If P is computable by a size s formula, then each P_i is computable by a size $O(sd)$ formula

Homogenization of formulas: interpolation

$$P(x_1, \ldots, x_n, y) = P_0(x_1, \ldots, x_n) + P_1(x_1, \ldots, x_n)y + \cdots + P_d(x_1, \ldots, x_n)y^d$$

$$\begin{bmatrix} 1 & \alpha_0 & \cdots & \alpha_0^d \\ 1 & \alpha_1 & \cdots & \alpha_1^d \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_d & \cdots & \alpha_d^d \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_d \end{bmatrix} = \begin{bmatrix} P(\alpha_0) \\ P(\alpha_1) \\ \vdots \\ P(\alpha_d) \end{bmatrix}$$

$$\begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_d \end{bmatrix} = \begin{bmatrix} \beta_{00} & \beta_{01} & \cdots & \beta_{0d} \\ \beta_{10} & \beta_{11} & \cdots & \beta_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{d0} & \beta_{d1} & \cdots & \beta_{dd} \end{bmatrix} \begin{bmatrix} P(\alpha_0) \\ P(\alpha_1) \\ \vdots \\ P(\alpha_d) \end{bmatrix}$$

$$P_i(x_1, \ldots, x_n) = \beta_0 P(x_1, \ldots, x_n, \alpha_0) + \cdots + \beta_d P(x_1, \ldots, x_n, \alpha_d).$$

Homogenization of formulas

- $P(x_1, \ldots, x_n) = Q_0 + \cdots + Q_d$
- Consider the polynomial $P'(x_1, \ldots, x_n, y) := P(yx_1, \ldots, yx_n)$
- Then: $P'(x_1, \ldots, x_n) = Q_0 + yQ_1 + \cdots + y^d Q_d$
- Computing higher degrees gives no advantage for formulas
- But it is unknown if homogeneous formulas are as powerful as general ones
- Exercise: Give a polynomial-size formula for:

$$\sum_{1 \le i_1 < i_2 < \cdots < i_d \le n} x_{i_1} \cdots x_{i_d} \quad \text{and} \quad \sum_{m \in \{\text{deg. } d \text{ monomials}\}} m$$

Lemma (Raz 2010)

Let Φ be a formula of size s computing an n-variate homogeneous polynomial f of degree d. Then there is an homogeneous formula Φ' computing f of size at most $\mathrm{poly}\left(s, \binom{d+\log s}{d}\right)$.

In particular, if $d = O(\log n)$ and $n = \mathrm{poly}(n)$ then we have $\mathrm{size}(\Phi') = \mathrm{poly}(n)$ as well.

Outline

Depth-reduction for formulas

Lemma (Brent 1974)

Let f be an n-variate degree d polynomial computed by an arithmetic formula Φ of size s. Then f can also be computed by a formula Φ' of size $s' = \text{poly}(s, n, d)$ and depth $O(\log s)$.

Depth-reduction for circuits and ABPs

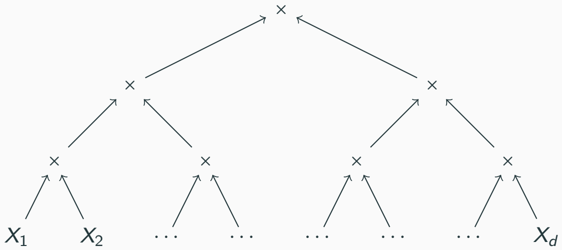
Theorem (Valiant, Skyum, Berkowitz, Rackoff 1983)

Let f be an n-variate degree d polynomial computed by an arithmetic circuit Φ of size s. Then there is an arithmetic circuit Φ' computing f of size $s' = \mathrm{poly}(s, n, d)$ and depth $O(\log d)$.

- In the resulting circuit the additions have unbounded fan-in
- It is enough to prove lower bounds for such log-depth circuits
- Easy to prove for ABPs

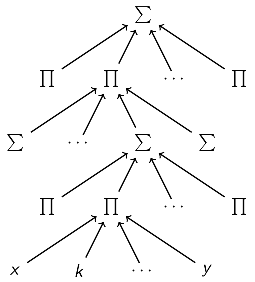
Depth-reduction for ABPs

- Compute IMM with a $\log d$-depth formula where gates compute matrix products
- Implementing each matrix-product gate with arithmetic operations can be done in constant depth if we use unbounded fan-in addition gates



Depth-reduction to constant depth

- If multiplication gates have fan-in 2, constant-depth circuits can only compute constant-degree polynomials
- We consider depth-4 circuits of a special form: $\Sigma\Pi\Sigma\Pi$



Theorem

Let f be an n-variate degree d polynomial computed by a size s arithmetic circuit. Then for any $0 < t \leq d$, f can be equivalently computed by a homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuit of top fan-in $s^{O(d/t)}$ and size $s^{O(t+d/t)}$.

Outline

Outline

A lower bound for general circuits

Theorem (Baur & Strassen 1983)

Any circuit computing simultaneously $x_1^d, \ldots, x_n^d$ *has size* $\Omega(n \log d)$

- Each gate $a \mapsto$ new variable z_a
- Collect the equations characterizing the local computation of each gate, e.g., $z_a - (z_b \cdot z_c) = 0$
- If z is an output gate, add the equation $z - 1 = 0$
- Solutions of this system:
  - Each x_i is mapped to a d-th root of unity
  - Other variables are set by the equations.
- **Bézout**: the number of common roots is at most the product of the degrees of the equations
- $d^n \leq 2^s$

A lower bound for general circuits

Lemma (Baur & Strassen 1983)

If f can be computed by a circuit of size s, then all first-order derivatives of f can be simultaneously computed by a circuit of size $O(s)$

- Simple proof by induction
- Same principle as backpropagation for neural networks

Theorem (Baur & Strassen 1983)

Any circuit computing $x_1^{d+1} + \cdots + x_n^{d+1}$ has size $\Omega(n \log d)$

Outline


ESSAYONS AVEC ÇA !..
NNNNNG!
PLUS PETIT...
CECI PEUT-ÊTRE ?..
BOF!
ÇA-Y-EST ! JE SUIS UN SURHOMME !!!
QUEL GRAND PRODIGE !...
34

Restricted computations

- Cancellations yield efficient computations
- Efficient computations produce “wrong” monomials which then cancel out
- Monotone computations: no cancellations, exponential lower bounds for circuit size
- Multilinear computations: only produce multilinear monomials, superpolynomial lower bounds for formula size
- Non-commutative lower bounds: cancelled monomials must be in the same order, exponential lower bounds for ABPs

Outline

Basic steps

For a given model:

1. Decomposition: show that any polynomial computed by such a model is a small sum of simple *building blocks* polynomials: $f = \sum_{i=1}^{s} f_i$
2. Measure: define a sub-additive measure $\mu : K[X] \to \mathbb{R}^+$
3. Simple blocks: show that if g is a building block, $\mu(g)$ is small
4. Explicit hard polynomial: find f such that $\mu(f)$ is big

   big $\leq \mu(f) \leq \mu(\sum_{i=1}^{s} f_i) \leq \sum_{i=1}^{s} \mu(f_i) \leq s \times$ small
5. ???
6. Profit

Lower bounds for depth-3 powering circuits

For $\Sigma \bigwedge \Sigma$ circuits:

1. Decomposition: $f = \sum_{i=1}^{s} L_i^{d_i}$, L_i a linear combination of the variables
2. Measure: $\mu_k(f)$: dimension of the space spanned by all partial derivatives of f of order k
3. Simple blocks: $\mu_k(L^d) = 1$, because any partial derivative is proportionnal to L^{d-k}
4. Explicit hard polynomial: per

   $\mu_k(\text{per}) = \binom{n}{k}^2$

   $2^n \leq \mu_{n/2}(f) \leq \mu_{n/2}(\sum_{i=1}^{s} L_i^{d_i}) \leq \sum_{i=1}^{s} \mu_{n/2}(L_i^{d_i}) \leq s$

Outline

Non-commutative setting

- $\mathbb{F}\langle x_1, \ldots, x_n \rangle$: ring of non-commutative polynomials
- Non-commutative: $x_i x_j \neq x_j x_i$. Need to order the children of the $\times$-gates.
- Non-commutative polynomial of degree $\leq d$:

$$f = \sum_{\substack{m \in \{x_1, \ldots, x_n\}^* \\ |m| \leq d}} \alpha_m . m \qquad (\alpha_m \in \mathbb{F})$$

Lower bounds for non-commutative ABPs

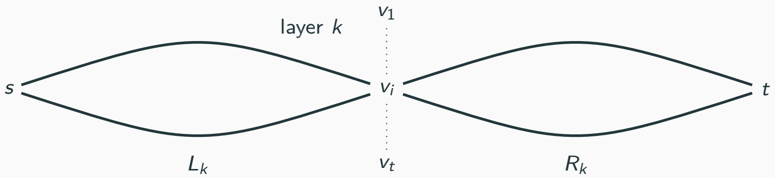
For homogeneous non-commutative ABPs:

1. Decomposition: $f = \sum_{i=1}^{w} l_i \cdot r_i$, cutting at layer k and partitioning depending on the intermediary vertex from layer k
2. Measure: $\mu_k(f)$: rank of the coefficient matrix with monomials of degree k for the lines and degree $n - k$ for the columns
3. Simple blocks: $\mu_k(l \cdot r) = 1$, because the coefficient matrix is then the product of two vectors
4. Explicit hard polynomial: Pal (or per or det)

   $\mu_k(\text{Pal}) = n^k$

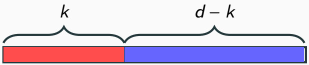
   $n^k \leq \mu_k(\text{Pal}) \leq \mu_k(\sum_{i=1}^{w} l_i \cdot r_i) \leq \sum_{i=1}^{w} \mu_k(l_i \cdot d_i) \leq s$

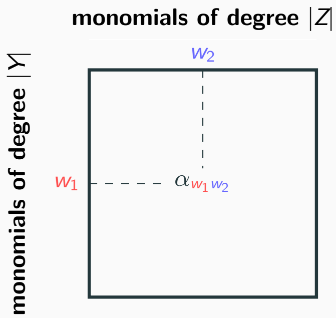
Decomposition



Measure: coefficient matrices

- $f = \sum_w \alpha_w . w$, homogeneous, degree d, n variables

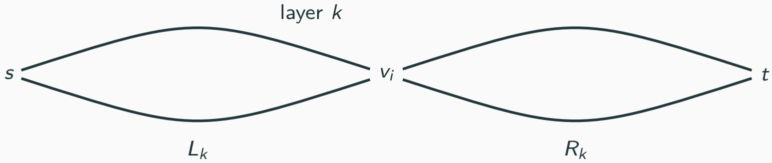




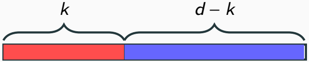
- Define matrix $M_k(f)$

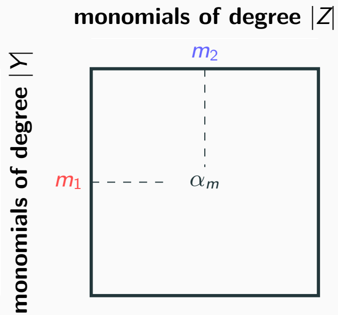
- Complexity measure : rank$(M_k(f))$.

Simple blocks



- $\Pi = (\{1, 2, \ldots, k\}, \{k+1, k+2, \ldots, d\})$





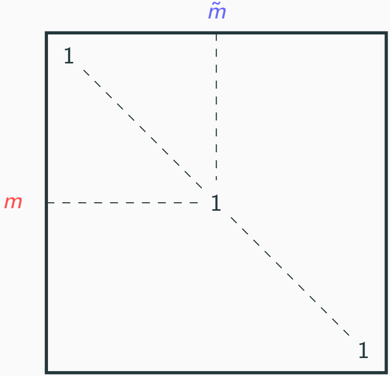
Explicit hard polynomial: the palindrome

- For $m \in \{x_1, \dots, x_n\}^*$, write $\tilde{m}$ for the word in reverse order

$$\mathrm{Pal}_d\, X = \sum_{m \in \{x_1, \dots, x_n\}^{d/2}} m \cdot \tilde{m}$$

- $\mathrm{Pal}_{d+1}\, X = \sum_{i=1}^{n} x_i \cdot \mathrm{Pal}_d\, X \cdot x_i$
- What is the matrix if we cut in the middle?

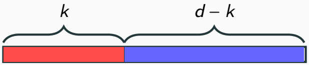
Explicit hard polynomial: the palindrome



$$n^{d/2} \leq \mu_{d/2}(\mathrm{Pal}) \leq \mu_{d/2}\Big(\sum_{i=1}^{w} l_i \cdot r_i\Big) \leq \sum_{i=1}^{w} \mu_{d/2}(l_i \cdot r_i) \leq s$$

Nisan's beautiful result

- $\Pi = (\{1, 2, \ldots, k\}, \{k+1, k+2, \ldots, d\})$



Theorem (Nisan, 1991)

For any homogeneous polynomial f of degree d, the size of a smallest homogeneous algebraic branching program for f is equal to

$$\sum_{k=0}^{d} \mathrm{rank}(M_k(f))$$

Corollary

Any homogeneous ABP computing the permanent has size $\geq 2^n$

Formal series on words and trees

- General results from the 70s imply Nisan's results and can be used to recover more recent extensions
- Provide a characterization of smallest circuit size for *non-associative* computations
- Does not seem to provide tools for non-commutative circuits

Now what?

- Many different open questions, in general and in restricted models
- New tools (measures)
- Completely new tools

References

- Completeness and Reduction in Algebraic Complexity Theory, Peter Bürgisser. Algorithms and Computation in Mathematics. Springer, 2000.
- Arithmetic Circuits: a survey of recent results and open questions, Amir Shpilka & Amir Yehudayoff. Foundations and Trends in Theoretical Computer Science 2010. https://www.cs.tau.ac.il/~shpilka/publications/SY10.pdf
- A survey of lower bounds in arithmetic circuit complexity. https://github.com/dasarpmar/lowerbounds-survey, maintained by Ramprasad Saptharishi
  Quite a few statements and examples borrowed!

- Please ask for detailed explanations...