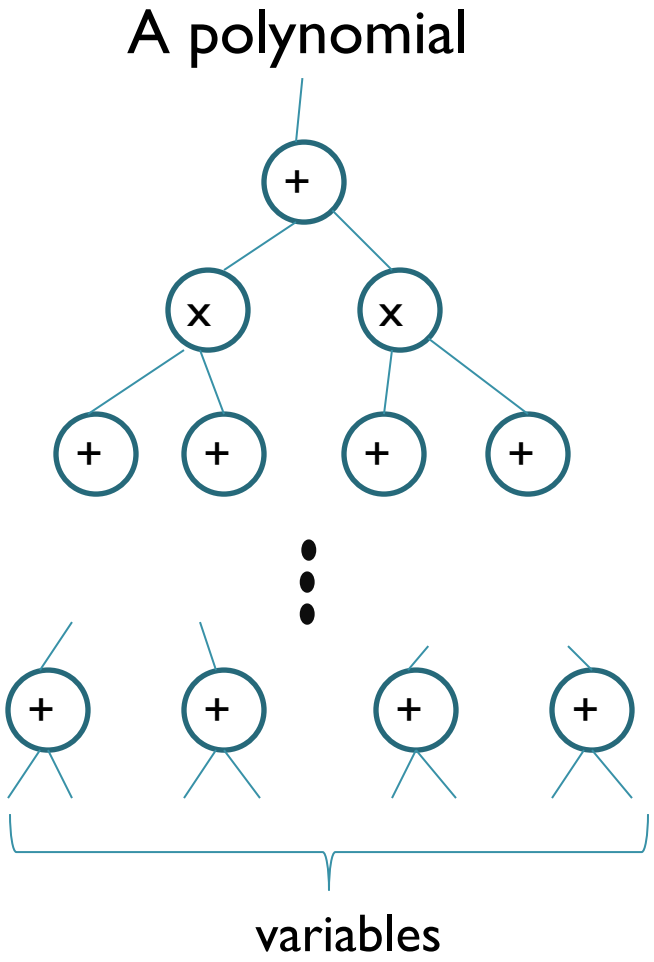
gct2022: School and Conference on Geometric Complexity Theory

# Arithmetic circuit reconstruction: Part I

Chandan Saha

Indian Institute of Science

# Arithmetic circuit



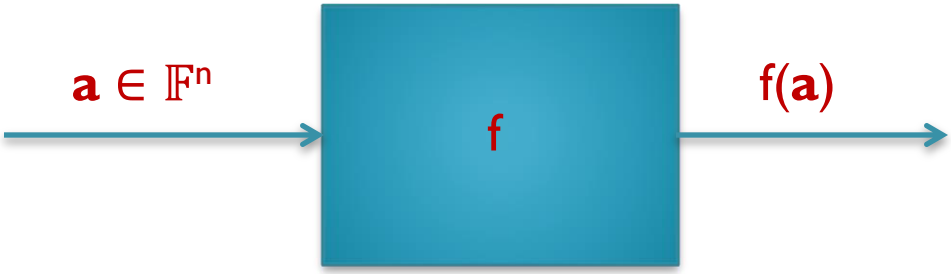

Size: Number of gates and wires

Depth: Length of the longest path from an input to the output node

# The reconstruction problem

- Let f(**x**) be a n-variate degree-d polynomial computed by a circuit of size s from a class *C*.

- Reconstruction problem for *C*. Given black-box access to f, output a *small* circuit computing f.




Black-box access to f
( membership query access to f )

# The reconstruction problem

- Let $f(\mathbf{x})$ be a $n$-variate degree-$d$ polynomial computed by a circuit of size $s$ from a class $C$.

- Reconstruction problem for $C$. Given black-box access to $f$, output a *small* circuit computing $f$.

- Size of the output circuit. Ideally, $\text{poly}(s)$.
- Proper learning. If the output circuit belongs to class $C$, then we say that the learning is proper.

# The reconstruction problem

- Let $f(\mathbf{x})$ be a $n$-variate degree-$d$ polynomial computed by a circuit of size $s$ from a class $C$.

- Reconstruction problem for $C$. Given black-box access to $f$, output a *small* circuit computing $f$.

- Efficiency. Ideally, $\text{poly}(d,s)$. But, even $N = \binom{n+d}{d}$ time reconstruction is non-trivial, for $n \ll s \ll N$, as exhaustive search over size-$s$ circuits takes $\exp(s)$ time.

# The reconstruction problem

- Let f(**x**) be a n-variate degree-d polynomial computed by a circuit of size s from a class *C*.

- Reconstruction problem for *C*. Given black-box access to f, output a *small* circuit computing f.

- Exact learning. By definition, reconstruction is an exact learning problem, i.e., the o/p circuit computes f exactly. This is unavoidable if we insist on outputting an arith. ckt. (as two different polys disagree almost everywhere).

# The reconstruction problem

- Let f(**x**) be a n-variate degree-d polynomial computed by a circuit of size s from a class *C*.

- Reconstruction problem for *C*. Given black-box access to f, output a *small* circuit computing f.

- PAC learning. It makes sense though to study PAC learnability of Boolean functions that match the output of arithmetic circuits on the Boolean hypercube [Klivans & Sherstov'09]. But this won't be the focus of the talk.

# The reconstruction problem

- Let f(**x**) be a n-variate degree-d polynomial computed by a circuit of size s from a class *C*.

- Reconstruction problem for *C*. Given black-box access to f, output a *small* circuit computing f.

- How hard is the reconstruction problem?

# Reconstruction implies lower bounds

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for $C$ implies super-polynomial lower bound for $C$.

- Naturally, existing reconstruction algorithms have focussed on circuit classes for which non-trivial lower bounds are known.

# Reconstruction implies lower bounds

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for $C$ implies super-polynomial lower bound for $C$.

- Naturally, existing reconstruction algorithms have focussed on circuit classes for which non-trivial lower bounds are known.

- Does lower bound imply reconstruction?

# Reconstruction is inherently hard

- Reconstruction is akin to approximating the minimum circuit size.

- Minimum Circuit Size Problem (MCSP). Given a truth-table $T$ of size $N = 2^n$ and an integer $s$, check if the function defined by $T$ has a circuit of size at most $s$.

# Reconstruction is inherently hard

- Reconstruction is akin to approximating the minimum circuit size.

- Minimum Circuit Size Problem (MCSP). Given a truth-table $T$ of size $N = 2^n$ and an integer $s$, check if the function defined by $T$ has a circuit of size at most $s$.

- Allender & Hirahara (2017): Approximating the minimum circuit size to within $N^{1-o(1)}$ factor is not in P, assuming the existence of one-way functions.

# Reconstruction is inherently hard

- Reconstruction is akin to approximating the minimum circuit size.

- Arithmetic MCSP. Given a coefficient vector T of size $N = \binom{n+d}{d}$ and an integer s, check if the polynomial defined by T has an arithmetic circuit of size at most s.

- Drawing analogy with the Boolean world, it is plausible that $N^{1-o(1)}$–approximate MCSP for arithmetic circuits cannot be done in poly(N) time.

# Reconstruction is inherently hard

- Reconstruction is akin to approximating the minimum circuit size.

- Arithmetic MCSP. Given a coefficient vector T of size $N = \binom{n+d}{d}$ and an integer s, check if the polynomial defined by T has an arithmetic circuit of size at most s.

- Drawing analogy with the Boolean world, it is plausible that $N^{1-o(1)}$–approximate MCSP for arithmetic circuits cannot be done in poly(N) time. (Open: Prove it.)

# Reconstruction is inherently hard

- Hardness results are known in some special cases.

- Set-multilinear depth-3 circuits (tensors). Computing tensor rank is NP-hard [Hastad'90].

- Approximating tensor rank to within $(1+\delta)$ factor for $\delta \approx 0.0005$ is also NP-hard [Swernofsky'18; Blaeser, Ikenmeyer, Jindal, Lysikov'18; Song, Woodruff, Zhong'19].

# Reconstruction is inherently hard

- Hardness results are known in some special cases.

- Depth-3 powering circuits (symmetric tensors). Computing Waring rank is NP-hard [Shitov'16].

- Open: Is there a $\delta > 0$ such that approximating Waring rank to within $(1+\delta)$ factor is NP-hard?

# Reconstruction for restricted classes

- Efficient reconstruction algorithms are known for several interesting circuit classes such as:

Constant depth

- Depth-2 circuits
- Depth-3 circuits with constant top fan-in
- Multilinear Depth-4 circuits with constant top fan-in

Constant read

- Read-Once Formulas
- Read–once Oblivious Algebraic Branching Programs

# Reconstruction for restricted classes

- Efficient reconstruction algorithms are known for several interesting circuit classes such as:

Constant depth

- Depth-2 circuits
- Depth-3 circuits with constant top fan-in
- Multilinear Depth-4 circuits with constant top fan-in

Constant read

- Read-Once Formulas
- Read–once Oblivious Algebraic Branching Programs

- This talk: Will discuss some of these *worst-case* algorithms.

# A convention

- We'll assume that the top gate of the circuit is a +-gate. If not, then apply black-box polynomial factorization algorithm to reduce to learning simpler polynomials.

- Black-box polynomial factoring [Kaltofen & Trager'90]. Given black-box access to a n-variate degree-d polynomial f, black-box access to the irreducible factors of f can be computed in randomized poly(n,d) time*.

- * Provided $|\mathbb{F}|$ and $\mathrm{char}(\mathbb{F})$ are sufficiently large.

# Depth-2 circuit reconstruction

# Depth-2 circuit reconstruction

- A depth-2 circuit of size s computes a sum of at most s monomials. A polynomial having at most s monomials is called a *s-sparse polynomial*.

- The reconstruction problem. Given black-box access to a n-variate degree-d s-sparse polynomial f, find the coefficient vector and the monomials of f.

- This is also known as the sparse polynomial interpolation problem.

# Depth-2 circuit reconstruction

- The problem has been intensely studied [Grigoriev & Karpinski'87; Ben-Or & Tiwari'88; Clausen, Dress, Grabmeier & Karpinski'91; Werther'94; Grigoriev, Karpinski & Singer'94; Karpinski & Shparlinski'96; Klivans & Spielman'01].

- Klivans & Spielman (2001): Sparse polynomial interpolation can be solved in deterministic poly(n,d,s) time.

# Depth-2 circuit reconstruction

- Let $f = \sum_{i \in [s]} c_i \cdot \mathbf{x}^{\mathbf{e_i}}$, where $c_i$ and $\mathbf{e_i}$ are unknown.

  (Assume $\mathbb{F} = \mathbb{Q}$.)

- Input: Black-box access to $f$.

  Output: The coefficients and monomials of $f$.

- Algorithm: (high-level)
  - Step 1. Recover the coefficients.
  - Step 2. Recover the monomials (exponent vectors).

# Step 1: Recovering the coefficients

- Step 1. Recover the coefficients $c_1, \ldots, c_s$.
  1. Pick a prime $p > (nds)^2$.
  2. For $t \in [ns^2]$
     - Let $g_t(x) = f(x, x^{t \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $g_t(x)$.
  3. Select a $g_t(x)$ that has the maximum sparsity.

  Output $t$ and $g_t(x)$.

# Step 1: Recovering the coefficients

- Step 1. Recover the coefficients $c_1, \ldots, c_s$.
  1. Pick a prime $p > (nds)^2$.
  2. For $t \in [ns^2]$
     - Let $g_t(x) = f(x, x^{t \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $g_t(x)$. ← $\deg(g_t) \le pd$
  3. Select a $g_t(x)$ that has the maximum sparsity.

  Output $t$ and $g_t(x)$.

# Step 1: Recovering the coefficients

- Step 1. Recover the coefficients $c_1, \ldots, c_s$.
  1. Pick a prime $p > (nds)^2$.
  2. For $t \in [ns^2]$
     - Let $g_t(x) = f(x, x^{t \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $g_t(x)$.
  3. Select a $g_t(x)$ that has the maximum sparsity.

     Output $t$ and $g_t(x)$.
- Let $\mathbf{e_i} = (e_{i,0}, \ldots, e_{i,n-1})$ and $E_i(y) = e_{i,0} + e_{i,1}y + \ldots + e_{i,n-1}y^{n-1}$.

# Step 1: Recovering the coefficients

- Step 1. Recover the coefficients $c_1, \ldots, c_s$.
  1. Pick a prime $p > (nds)^2$.
  2. For $t \in [ns^2]$
     - Let $g_t(x) = f(x, x^{t \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $g_t(x)$.
  3. Select a $g_t(x)$ that has the maximum sparsity.

     Output $t$ and $g_t(x)$.
- Let $\mathbf{e_i} = (e_{i,0}, \ldots, e_{i,n-1})$ and $E_i(y) = e_{i,0} + e_{i,1}y + \ldots + e_{i,n-1}y^{n-1}$.
- As $E_i(y) \neq E_j(y)$ in $\mathbb{F}_p[y]$, $\exists t \in [ns^2]$ s.t. $E_i(t) \neq_p E_j(t)$ for every pair $(i, j)$. (As $p > \max\{ns^2, d\}$)

# Step 1: Recovering the coefficients

- Step 1. Recover the coefficients $c_1, \ldots, c_s$.
  1. Pick a prime $p > (nds)^2$.
  2. For $t \in [ns^2]$
     - Let $g_t(x) = f(x, x^{t \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $g_t(x)$.
  3. Select a $g_t(x)$ that has the maximum sparsity.

     Output $t$ and $g_t(x)$.
- Let $D_i(t) = e_{i,0} + e_{i,1}(t \bmod p) + \ldots + e_{i,n-1}(t^{n-1} \bmod p)$.
- Note, $D_i(t) =_p E_i(t)$.

# Step 1: Recovering the coefficients

- Step 1. Recover the coefficients $c_1, \ldots, c_s$.
  1. Pick a prime $p > (nds)^2$.
  2. For $t \in [ns^2]$
     - Let $g_t(x) = f(x, x^{t \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $g_t(x)$.
  3. Select a $g_t(x)$ that has the maximum sparsity.

     Output $t$ and $g_t(x)$.
- Let $D_i(t) = e_{i,0} + e_{i,1}(t \bmod p) + \ldots + e_{i,n-1}(t^{n-1} \bmod p)$.
- Note, $D_i(t) =_p E_i(t)$. So, $\exists t \in [ns^2]$ s.t. $D_i(t) \neq D_j(t)$ for $i \neq j$, and $g_t(x) = \sum_{i \in [s]} c_i \cdot x^{D_i(t)}$ has the maximum sparsity.

# Step 2: Recovering the exponents

- Step 2. Recover $\mathbf{e_i} = (e_{i,0}, \ldots, e_{i,n-1})$.
  1. Let $t$ and $g_t(x)$ be as returned by Step 1.
  2. For $j \in \{0,\ldots,n-1\}$
     - Let $h_j(x) = f(x, x^{t \bmod p}, \ldots, 2x^{t^j \bmod p}, \ldots, x^{t^{n-1} \bmod p})$.
     - Interpolate $h_j(x)$.
     - Compare $h_j(x)$ and $g_t(x)$ to find $e_{i,j}$ for all $i \in [s]$.

- Observe, $h_j(x) = \sum_{i \in [s]} c_i \cdot 2^{e_{i,j}} \cdot x^{D_i(t)}$ and $g_t(x) = \sum_{i \in [s]} c_i \cdot x^{D_i(t)}$.

# Can we reconstruct Depth-3 circuits?

# Depth-3 circuit reconstruction

- A depth-3 circuit computes a sum of products of linear polynomials, i.e., f can be expressed as

$$f = \ell_{1,1} \cdot \ell_{1,2} \cdot \ldots \cdot \ell_{1,m} + \ldots + \ell_{s,1} \cdot \ell_{s,2} \cdot \ldots \cdot \ell_{s,m},$$

  where $\ell_{i,j}$ has degree 1.

- The reconstruction problem. Given black-box access to a depth-3 circuit computing f, output a small circuit for f.

- How hard is the above problem?

# Depth-3 circuit reconstruction

- A depth-3 circuit computes a sum of products of linear polynomials, i.e., f can be expressed as

$$f = \ell_{1,1} \cdot \ell_{1,2} \cdot \ldots \cdot \ell_{1,m} + \ldots + \ell_{s,1} \cdot \ell_{s,2} \cdot \ldots \cdot \ell_{s,m},$$

  where $\ell_{i,j}$ has degree 1.

- Depth reduction. It turns out that poly-time reconstruction of depth-3 circuits implies sub-exponential time reconstruction of poly-size general circuits. [Gupta, Kamath, Kayal & Saptharishi'13; Tavenas'13; Koiran'12; Agrawal & Vinay'08]

# Depth-3 circuit reconstruction

- A depth-3 circuit computes a sum of products of linear polynomials, i.e., f can be expressed as

$$f = \ell_{1,1} \cdot \ell_{1,2} \cdot \ldots \cdot \ell_{1,m} + \ldots + \ell_{s,1} \cdot \ell_{s,2} \cdot \ldots \cdot \ell_{s,m} ,$$

  where $\ell_{i,j}$ has degree 1.

- But, what if $\ell_{1,1} = \ell_{1,2} = \ldots = \ell_{1,m}$ ? Can we reconstruct depth-3 powering circuits efficiently?

# Depth-3 powering circuits

- A depth-3 powering circuit (a.k.a $\Sigma\wedge\Sigma$ circuit) computes a sum of powers of linear polynomials, i.e.,

$$f = \ell_1^{d_1} + \ldots + \ell_s^{d_s} ,$$

  where $\ell_i$ has degree $1$.

- Fischer (1994): The monomial $x_1 \cdot x_2 \cdot \ldots \cdot x_n$ can be expressed as a sum of $2^{n-1}$ powers of linear forms.

- The model is complete, i.e., every polynomial can be computed by a $\Sigma\wedge\Sigma$ circuit.

# Depth-3 powering circuits

- Strong lower bounds are known for $\sum\wedge\sum$ circuits.

- Waring rank. If f is a degree-d form, then the smallest s such that f can be expressed as a sum of s many d-th powers of linear forms is called the *Waring rank* of f.

- Carlini, Catalisano & Geramita (2012); Gashkov & Shavgulidze (2014): The Waring rank of $x_1^{e_1}\cdot x_2^{e_2}\cdot\ldots\cdot x_n^{e_n}$ (over $\mathbb{C}$) is $(e_2 + 1)\cdot\ldots\cdot(e_n + 1)$, where $e_1 \le e_2 \ldots \le e_n$.

# Learning depth-3 powering circuits

- The reconstruction problem. Given black-box access to a $\sum\wedge\sum$ circuit computing f, output a small circuit for f.

- Proper learning seems hard as computing Waring rank is NP-hard [Shitov'16].
- Proper learning $\sum\wedge\sum$ circuits is also known as the *symmetric tensor decomposition* problem.

# Learning depth-3 powering circuits

- The reconstruction problem. Given black-box access to a $\sum\wedge\sum$ circuit computing f, output a small circuit for f.

- Proper learning seems hard as computing Waring rank is NP-hard [Shitov'16].

- What about improper learning?

- We need an alternative circuit representation for polynomials computable by small $\sum\wedge\sum$ circuits.

# Read-once Oblivious ABP

- Let $h_i = x_1 + x_2 + \ldots + x_i$ for $i \in [n]$. Observe that

$$(1 \;\; h_i \;\; h_i^2 \ldots h_i^d) \cdot (x_{i+1}^j \;\; jx_{i+1}^{j-1} \;\; \ldots \;\; 1 \;\; 0 \ldots 0)^T = h_{i+1}^j$$

$$(1 \;\; h_i \;\; h_i^2 \ldots h_i^d) \cdot N(x_{i+1}) = (1 \;\; h_{i+1} \;\; h_{i+1}^2 \ldots h_{i+1}^d),$$

where $N(x_{i+1})$ is a $(d+1) \times (d+1)$ matrix whose entries are univariate polynomials in the $x_{i+1}$ variable.

- $h_{i+1}^j = x_{i+1}^j + jx_{i+1}^{j-1} \cdot h_i + \ldots + h_i^j$.

# Read-once Oblivious ABP

- Let $h_i = x_1 + x_2 + \ldots + x_i$ for $i \in [n]$. Observe that

$$(1 \;\; h_i \;\; h_i^2 \ldots h_i^d) \cdot (x_{i+1}^j \;\; jx_{i+1}^{j-1} \;\; \ldots \;\; 1 \;\; 0 \ldots 0)^T = h_{i+1}^j$$

$$(1 \;\; h_i \;\; h_i^2 \ldots h_i^d) \cdot N(x_{i+1}) = (1 \;\; h_{i+1} \;\; h_{i+1}^2 \ldots h_{i+1}^d),$$

where $N(x_{i+1})$ is a $(d+1) \times (d+1)$ matrix whose entries are univariate polynomials in the $x_{i+1}$ variable.

- Obs. If $f$ is computable by a $\Sigma\wedge\Sigma$ circuit, then

$$f = M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $(d+1)s \times (d+1)s$ matrix with entries in $\mathbb{F}[x_i]$.

# Read-once Oblivious ABP

- Let $h_i = x_1 + x_2 + \ldots + x_i$ for $i \in [n]$. Observe that

$$(1 \;\; h_i \;\; h_i^2 \ldots h_i^d) \cdot (x_{i+1}^j \;\; jx_{i+1}^{j-1} \;\; \ldots \;\; 1 \;\; 0 \ldots 0)^T = h_{i+1}^j$$

$$(1 \;\; h_i \;\; h_i^2 \ldots h_i^d) \cdot N(x_{i+1}) = (1 \;\; h_{i+1} \;\; h_{i+1}^2 \ldots h_{i+1}^d),$$

where $N(x_{i+1})$ is a $(d+1) \times (d+1)$ matrix whose entries are univariate polynomials in the $x_{i+1}$ variable.

- Obs. If $f$ is computable by a $\Sigma\wedge\Sigma$ circuit, then

Row vector ← Column vector

$$f = M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $(d+1)s \times (d+1)s$ matrix with entries in $\mathbb{F}[x_i]$.

# Read-once Oblivious ABP

- ROABP. An n-variate Read-once Oblivious Algebraic Branching Program R is a product of the form

$$M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $w_{i-1} \times w_i$ matrix with entries in $\mathbb{F}[x_{i+1}]$.

- $w_0 = w_n = 1$. Size(R) := $\sum_i w_i$ .
- $(w_1, \ldots, w_{n-1})$ is the *width sequence* of R.

# Read-once Oblivious ABP

- ROABP. An n-variate Read-once Oblivious Algebraic Branching Program R is a product of the form

$$M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $w_{i-1} \times w_i$ matrix with entries in $\mathbb{F}[x_{i+1}]$.

- $w_0 = w_n = 1$. Size(R) := $\sum_i w_i$ .
- $(w_1, \ldots, w_{n-1})$ is the *width sequence* of R.
- Obs. The polynomial computed by R has a poly-sized circuit as matrix multiplication has a poly-sized circuit.

# Read-once Oblivious ABP

- ROABP. An n-variate Read-once Oblivious Algebraic Branching Program R is a product of the form

$$M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $w_{i-1} \times w_i$ matrix with entries in $\mathbb{F}[x_{i+1}]$.

- $w_0 = w_n = 1$. Size(R) := $\sum_i w_i$ .
- $(w_1, \ldots, w_{n-1})$ is the *width sequence* of R.

- ROABPs capture several other interesting circuit models.

# Read-once Oblivious ABP

- ROABP. An n-variate Read-once Oblivious Algebraic Branching Program R is a product of the form

$$M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $w_{i-1} \times w_i$ matrix with entries in $\mathbb{F}[x_{i+1}]$.

- The reconstruction problem. Given black-box access to a ROABP computing f, output a small circuit for f.

# Read-once Oblivious ABP

- ROABP. An n-variate Read-once Oblivious Algebraic Branching Program R is a product of the form

$$M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n),$$

where $M_i(x_i)$ is a $w_{i-1} \times w_i$ matrix with entries in $\mathbb{F}[x_{i+1}]$.

- The reconstruction problem. Given black-box access to a ROABP computing f, output a small circuit for f.

- Proper learning is known!

# ROABP reconstruction

# ROABP reconstruction

- Beimel, Bergadano, Bshouty, Kushilevitz & Varricchio (2000): ROABP reconstruction can be solved in randomized polynomial time.

- Forbes & Shpilka (2013). ROABP reconstruction can be solved in deterministic quasi-polynomial time.
- Uses efficient construction of hitting sets for ROABPs .(FS'13; Agrawal, Gurjar, Korwar & Saxena'15)

# ROABP reconstruction

- Beimel, Bergadano, Bshouty, Kushilevitz & Varricchio (2000): ROABP reconstruction can be solved in randomized polynomial time.

- Forbes & Shpilka (2013). ROABP reconstruction can be solved in deterministic quasi-polynomial time.
- Uses efficient construction of hitting sets for ROABPs .(FS'13; Agrawal, Gurjar, Korwar & Saxena'15)

- We will give an overview of the randomized algorithm.

# Randomized ROABP reconstruction

- Let $M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n)$ be a <u>minimal</u> ROABP computing $f$, and $(w_1, \ldots, w_{n-1})$ be its width sequence.
- Input: Black-box access to $f$.

Output: An ROABP $N_1(x_1) \cdot N_2(x_2) \cdot \ldots \cdot N_n(x_n)$ for $f$ that has width sequence $(w_1, \ldots, w_{n-1})$.

# Randomized ROABP reconstruction

- Let $M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n)$ be a minimal ROABP computing $f$, and $(w_1, \ldots, w_{n-1})$ be its width sequence.
- Input: Black-box access to $f$.

  Output: An ROABP $N_1(x_1) \cdot N_2(x_2) \cdot \ldots \cdot N_n(x_n)$ for $f$ that has width sequence $(w_1, \ldots, w_{n-1})$.

- Obs. Minimal ROABP for $f$ is not unique. If $M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n)$ is a minimal ROABP, then so is $M_1(x_1)A \cdot A^{-1}M_2(x_2) \cdot \ldots \cdot M_n(x_n)$ for any $A \in GL(w_1, \mathbb{F})$.
- The algorithm outputs some minimal ROABP.

# Randomized ROABP reconstruction

- Let $M_1(x_1) \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n)$ be a minimal ROABP computing $f$, and $(w_1, \ldots, w_{n-1})$ be its width sequence.
- Input: Black-box access to $f$.

  Output: An ROABP $N_1(x_1) \cdot N_2(x_2) \cdot \ldots \cdot N_n(x_n)$ for $f$ that has width sequence $(w_1, \ldots, w_{n-1})$.

- Algorithm: (high-level)
  - Step 1. Find the width sequence $(w_1, \ldots, w_{n-1})$.
  - Step 2. Construct $N_1(x_1), N_2(x_2), \ldots$ sequentially.

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x} \backslash \mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Obs 1. $g_1, \ldots g_w$ and $h_1, \ldots, h_w$ are $\mathbb{F}$-linearly independent.
- Proof~. If not, then there's a $A \in GL(w, \mathbb{F})$ s.t.

  $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) \cdot A = (g_1(\mathbf{x}_i) \ldots g_{w'}(\mathbf{x}_i)\ 0 \ldots 0)$,

  where $w' < w$. This will violate minimality of size.

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$.
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$.
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- Fact *. If $f_1,\ldots,f_t \in \mathbb{F}[\mathbf{x}]$ are l.i., and $\mathbf{a}_1,\ldots,\mathbf{a}_t \in_r \mathbb{F}^{|\mathbf{x}|}$, then $\operatorname{rank}(f_k(\mathbf{a}_j))_{k,j\in[t]} = t$ w.h.p. (use the Schwartz-Zippel lemma)

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$. $f(\mathbf{x}_i, \mathbf{b}_j) = g_1(\mathbf{x}_i)h_1(\mathbf{b}_j) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{b}_j)$
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- Proof~. $(f(\mathbf{x}_i, \mathbf{b}_1) \ \ldots \ f(\mathbf{x}_i, \mathbf{b}_t)) = (g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) \cdot H$ .

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$. $f(\mathbf{x}_i, \mathbf{b}_j) = g_1(\mathbf{x}_i)h_1(\mathbf{b}_j) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{b}_j)$
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- Proof~. By Obs 1, $H = (h_k(\mathbf{b}_j))_{k\in[w],\, j\in[t]}$ has rank $w$ w.h.p. (use Fact *)

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$. $f(\mathbf{x}_i, \mathbf{b}_j) = g_1(\mathbf{x}_i)h_1(\mathbf{b}_j) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{b}_j)$
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- Proof~. $(f(\mathbf{x}_i, \mathbf{b}_1) \ldots f(\mathbf{x}_i, \mathbf{b}_t)) = (g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) \cdot H$ .

↑ rank w

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$.
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- Clm 0. Given b.b.a. to $f_1,\ldots, f_t \in \mathbb{F}[\mathbf{x}]$, we can compute b.b.a. to a basis of $\langle f_1,\ldots, f_t\rangle$ in randomized poly-time.

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$.
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- $\text{Evaldim}_{|\mathbf{y}_i|}(f) := \dim\langle f(\mathbf{x}_i, \mathbf{b}) : \mathbf{b} \in \mathbb{F}^{|\mathbf{y}_i|}\rangle = w$ (Clm 1).

# Step 1: Finding the width sequence

$$f = [M_1(x_1)\cdot\ldots\cdot M_i(x_i)] \cdot [M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_n)]$$

- Let $w = w_i$, $\mathbf{x}_i = x_1 \uplus \ldots \uplus x_i$ and $\mathbf{y}_i = \mathbf{x}\backslash\mathbf{x}_i$ .
- Let $(g_1(\mathbf{x}_i) \ldots g_w(\mathbf{x}_i)) = M_1(x_1)\cdot\ldots\cdot M_i(x_i)$ and
  $(h_1(\mathbf{y}_i) \ldots h_w(\mathbf{y}_i))^T = M_{i+1}(x_{i+1})\cdot\ldots\cdot M_n(x_i)$ .
- Then, $f = g_1(\mathbf{x}_i)h_1(\mathbf{y}_i) + \ldots + g_w(\mathbf{x}_i)h_w(\mathbf{y}_i)$ .

- Pick $\mathbf{b}_1,\ldots,\mathbf{b}_t \in_r \mathbb{F}^{|\mathbf{y}_i|}$.
- Clm 1. If $t \geq w$ then $\dim\langle f(\mathbf{x}_i, \mathbf{b}_1), \ldots, f(\mathbf{x}_i, \mathbf{b}_t)\rangle = w$ w.h.p.
- By Clm 0 & 1, $\text{Evaldim}_{|\mathbf{y}_i|}(f)$ is efficiently computable.

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- It means not only can we find $\text{Evaldim}_{|\mathbf{y}_i|}(f)$ efficiently (Clm 0 & 1), but we can also learn a decomposition for f (as above) efficiently.

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- Proof~. Pick $\mathbf{b}_1, \ldots, \mathbf{b}_w \in_r \mathbb{F}^{|\mathbf{y}_i|}$. Set $g'_j(\mathbf{x}_i) = f(\mathbf{x}_i, \mathbf{b}_j)$.

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- Proof~. Pick $\mathbf{b}_1, \ldots, \mathbf{b}_w \in_r \mathbb{F}^{|\mathbf{y}_i|}$. Set $g'_j(\mathbf{x}_i) = f(\mathbf{x}_i, \mathbf{b}_j)$.

Observe that

$$(f(\mathbf{x}_i, \mathbf{b}_1) \ \ldots \ f(\mathbf{x}_i, \mathbf{b}_w)) = (g_1(\mathbf{x}_i) \ \ldots \ g_w(\mathbf{x}_i)) \cdot H ,$$

where $H = (h_k(\mathbf{b}_j))_{k,j\in[w]}$ has rank $w$ w.h.p. Also,

$$f = \underbrace{(g_1(\mathbf{x}_i) \ \ldots \ g_w(\mathbf{x}_i)) \cdot H}_{\mathbf{g'}} \cdot \underbrace{H^{-1} \cdot (h_1(\mathbf{y}_i) \ \ldots \ h_w(\mathbf{y}_i))^T}_{\mathbf{h'}} .$$

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- Proof~. From b.b.a. to $g'_1, \ldots, g'_w$ we compute b.b.a. to $h'_1, \ldots, h'_w$ as follows: Pick $\mathbf{a}_1, \ldots, \mathbf{a}_w \in_r \mathbb{F}^{|\mathbf{x}_i|}$. Observe,

$$(f(\mathbf{a}_1, \mathbf{y}_i) \ \ldots \ f(\mathbf{a}_w, \mathbf{y}_i))^T = G \cdot (h'_1(\mathbf{y}_i) \ \ldots \ h'_w(\mathbf{y}_i))^T ,$$

where $G = (g'_k(\mathbf{a}_j))_{j,k\in[w]}$ has rank $w$ w.h.p. Set

$$(h'_1(\mathbf{y}_i) \ \ldots \ h'_w(\mathbf{y}_i))^T = G^{-1} \cdot (f(\mathbf{a}_1, \mathbf{y}_i) \ \ldots \ f(\mathbf{a}_w, \mathbf{y}_i))^T .$$

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- For simplicity, assume $w_1 = \ldots = w_{n-1} = w$ henceforth.

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- Constructing $N_1$. Set $N_1(x_1) = (g'_1(x_1) \ldots g'_w(x_1))$ (Clm 2)

$$= M_1(x_1) \cdot H ,$$

where $H \in GL(w, \mathbb{F})$.

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- Constructing $N_1$. Set $N_1(x_1) = (g'_1(x_1) \ldots g'_w(x_1))$ (Clm 2)

$$= M_1(x_1) \cdot H ,$$

where $H \in GL(w, \mathbb{F})$.

- Note. $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = H^{-1} \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n)$

$$= M'_2(x_2) \cdot \ldots \cdot M_n(x_n) .$$

# Step 2: Constructing the matrices

- Clm 2. In randomized poly-time, we can compute b.b.a. to $g'_1(\mathbf{x}_i), \ldots, g'_w(\mathbf{x}_i)$ and $h'_1(\mathbf{y}_i), \ldots, h'_w(\mathbf{y}_i)$ such that

$$f = g'_1(\mathbf{x}_i)h'_1(\mathbf{y}_i) + \ldots + g'_w(\mathbf{x}_i)h'_w(\mathbf{y}_i) .$$

- Constructing $N_1$. Set $N_1(x_1) = (g'_1(x_1) \ldots g'_w(x_1))$ (Clm 2)

$$= M_1(x_1) \cdot H ,$$

where $H \in GL(w, \mathbb{F})$.

- Note. $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = H^{-1} \cdot M_2(x_2) \cdot \ldots \cdot M_n(x_n)$

$$= M'_2(x_2) \cdot \ldots \cdot M_n(x_n) .$$

- Next, we try to recover $M'_2$. But, $M'_2$ is not unique!

# Step 2: Constructing the matrices

- Canonical $M'_2$. There's a $A \in GL(w, \mathbb{F})$ s.t. $M'_2A$ looks like:
  - 1st row : $\overbrace{P_{1,1} \cdots P_{1,s1}}^{\text{l.i.}}$ $0 \cdots$ $\overbrace{P_{2,s1+1} \cdots P_{2,s2}}^{\text{l.i.}}$ $0 \cdots$ $\overbrace{P_{3,s2+1} \cdots P_{3,s3}}^{\text{l.i.}}$ $0 \cdots$
  - 2nd row: $P_{2,1} \cdots P_{2,s1}$
  - 3rd row: $P_{3,1} \cdots P_{3,s2}$
  - …

- A is simply a column operation matrix.

- We will recover a $N_2$ in the above form.

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A \in GL(w, \mathbb{F})$ s.t. $M'_2 A$ looks like:
  - 1st row : $P_{1,1} \cdots P_{1,s1}$ $0 \ldots$
  - 2nd row: $P_{2,1} \cdots P_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T = A^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A \in GL(w, \mathbb{F})$ s.t. $M'_2 A$ looks like:
  - 1st row : $P_{1,1} \cdots P_{1,s1}$ $0 \ldots$
  - 2nd row: $P_{2,1} \cdots P_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T = A^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2 A \cdot (q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T$ .

Known from the construction of $N_1$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A \in GL(w, \mathbb{F})$ s.t. $M'_2A$ looks like:
  - 1st row : $p_{1,1} \cdots p_{1,s1}$ $0 \ldots$
  - 2nd row: $p_{2,1} \cdots p_{2,s1}$ $p_{2,s1+1} \cdots p_{2,s2}$ $0 \ldots$
- Let $(q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T = A^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A \cdot (q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T$ .
- $h'_1(\mathbf{y}_1) = p_{1,1}(x_1)\, q_1(\mathbf{y}_2) + \ldots + p_{1,s1}(x_1)\, q_{s1}(\mathbf{y}_2)$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A \in GL(w, \mathbb{F})$ s.t. $M'_2 A$ looks like:
  - 1st row : $p_{1,1} \cdots p_{1,s1}$ $0 \ldots$
  - 2nd row: $p_{2,1} \cdots p_{2,s1}$ $p_{2,s1+1} \cdots p_{2,s2}$ $0 \ldots$
- Let $(q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T = A^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2 A \cdot (q_1(\mathbf{y}_2) \ldots q_w(\mathbf{y}_2))^T$ .
- $h'_1(\mathbf{y}_1) = p'_{1,1}(x_1)\, q'_1(\mathbf{y}_2) + \ldots + p'_{1,s1}(x_1)\, q'_{s1}(\mathbf{y}_2)$ (Clm 2)

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_1(\mathbf{y}_1) = P'_{1,1}(x_1)\, q'_1(\mathbf{y}_2) + \ldots + P'_{1,s1}(x_1)\, q'_{s1}(\mathbf{y}_2)$ (Clm 2)

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_1(\mathbf{y}_1) = P'_{1,1}(x_1)\, q'_1(\mathbf{y}_2) + \ldots + P'_{1,s1}(x_1)\, q'_{s1}(\mathbf{y}_2)$ (Clm 2)
- Next, we try to discover the 2nd row.

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) = (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) = (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$
- Pick $\mathbf{b}_1, \ldots, \mathbf{b}_{s1} \in_r \mathbb{F}^{|\mathbf{y}_2|}$. Then,

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) = (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$
- $(h'_2(x_2, \mathbf{b}_1) \ldots h'_2(x_2, \mathbf{b}_{s1})) = (P'_{2,1} \cdots P'_{2,s1}) \cdot Q +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot P$

$\in GL(s1, \mathbb{F})$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) = (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$
- $(h'_2(x_2, \mathbf{b}_1) \ldots h'_2(x_2, \mathbf{b}_{s1})) \cdot Q^{-1} = (P'_{2,1} \cdots P'_{2,s1}) +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot P \cdot Q^{-1}$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) = (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$
- $(h'_2(x_2, \mathbf{b}_1) \ldots h'_2(x_2, \mathbf{b}_{s1})) \cdot Q^{-1} = (P'_{2,1} \cdots P'_{2,s1}) +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot P \cdot Q^{-1}$

Take this as $(P'_{2,1} \cdots P'_{2,s1})$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) = (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T +$
  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) - (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T =$

  $(P_{2,s1+1} \cdots P_{2,s2}) \cdot (q'_{s1+1}(\mathbf{y}_2) \ldots q'_{s2}(\mathbf{y}_2))^T$

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A' \in GL(w, \mathbb{F})$ s.t. $M'_2A'$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ $0 \ldots$
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1}$ $P_{2,s1+1} \cdots P_{2,s2}$ $0 \ldots$
- Let $(q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T = A'^{-1} \cdot M_3(x_3) \cdot \ldots \cdot M_n(x_n)$ .
- Then, $(h'_1(\mathbf{y}_1) \ldots h'_w(\mathbf{y}_1))^T = M'_2A' \cdot (q'_1(\mathbf{y}_2) \ldots q'_w(\mathbf{y}_2))^T$.
- $h'_2(\mathbf{y}_1) - (P'_{2,1} \cdots P'_{2,s1}) \cdot (q'_1(\mathbf{y}_2) \ldots q'_{s1}(\mathbf{y}_2))^T =$
  $(P'_{2,s1+1} \cdots P'_{2,s2}) \cdot (q''_{s1+1}(\mathbf{y}_2) \ldots q''_{s2}(\mathbf{y}_2))^T$

(Applying Clm 2 again)

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A'' \in GL(w, \mathbb{F})$ s.t. $M'_2A''$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ 0 …
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1} P'_{2,s1+1} \cdots P'_{2,s2}$ 0 …

- Continue like this to discover the other rows of $N_2$.

# Step 2: Constructing the matrices

- Canonical $M'_2$. $\exists A'' \in GL(w, \mathbb{F})$ s.t. $M'_2A''$ looks like:
  - 1st row : $P'_{1,1} \cdots P'_{1,s1}$ 0 …
  - 2nd row: $P'_{2,1} \cdots P'_{2,s1} P'_{2,s1+1} \cdots P'_{2,s2}$ 0 …

- Continue like this to discover the other rows of $N_2$.

- Apply the same strategy to construct $N_3$, $N_4$, …

# Read-once formula reconstruction

- Read-once formulas. A formula is *read-once* if every variable labels exactly one leaf node. (ROFs $\subset$ ROABPs)

- Bshouty, Hancock & Hellerstein (1995): ROF reconstruction can be done in randomized poly-time.

- Shpilka & Volkovich (2014); Minahan & Volkovich (2018). ROF reconstruction in deterministic poly-time.
- Uses efficient construction of hitting sets for ROFs .
- These algorithms are proper.

# Back to $\sum\wedge\sum$ circuit reconstruction

- ROABP reconstruction gives an improper learning algorithm for $\sum\wedge\sum$ circuits.

- Recall, computing Waring rank is NP-hard.

- Can we proper learn $\sum\wedge\sum$ circuits computing polynomials that have constant Waring rank?

# Back to $\sum\wedge\sum$ circuit reconstruction

- ROABP reconstruction gives an improper learning algorithm for $\sum\wedge\sum$ circuits.
- Recall, computing Waring rank is NP-hard.
- Can we proper learn $\sum\wedge\sum$ circuits computing polynomials that have constant Waring rank?
- Yes, we can!

# $\Sigma\Pi\Sigma(k)$ circuit reconstruction

- $\Sigma\Pi\Sigma(k)$ ≡ depth-3 circuits with top fan-in $k = O(1)$.
- Shpilka (2007); Karnin & Shpilka (2009): Randomized quasi-poly-time reconstruction over *poly-sized fields*.
- Sinha (2016, 2020): Randomized poly-time for $k = 2$.
- Bhargava, Saraf & Volkovich (2021). Randomized poly-time for $\Sigma\wedge\Sigma(k)$ and *multilinear* $\Sigma\Pi\Sigma(k)$ circuits.
- These algorithms are (mostly) proper.

# Depth-4 circuit reconstruction

- $\Sigma\Pi\Sigma\Pi(k)$ ≡ depth-4 circuits with top fan-in $k = O(1)$.

- Gupta, Kayal & Lokam (2012). Randomized poly-time for *multilinear* $\Sigma\Pi\Sigma\Pi(2)$ circuits.

- Bhargava, Saraf & Volkovich (2019). Randomized quasi-poly-time reconstruction for *multilinear* $\Sigma\Pi\Sigma\Pi(k)$ circuits over *poly-size fields*.

- These algorithms are proper.

# Proper learning $\sum\wedge\sum(k)$ circuits

# Proper learning $\sum\wedge\sum(k)$ circuits

- $\sum\wedge\sum(k)$ circuit computing f:

  $f = \ell_1^{d_1} + \ldots + \ell_k^{d_k}$, where $\ell_i$ has degree 1.

- Input: Black-box access to f.

  Output: A $\sum\wedge\sum(k)$ circuit for f.

- Algorithm: (high-level)
  - Step 1. Remove redundant variables.
  - Step 2. Solve a polynomial system in $k^2$ variables.

# Essential & redundant variables

- Essential & redundant vars: The smallest $k$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$ for a $A \in GL(n, \mathbb{F})$ is the number of *essential* variables of $f$. $\{x_{k+1}, \ldots, x_n\}$ are the *redundant* vars.

# Essential & redundant variables

- Essential & redundant vars: The smallest $k$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$ for a $A \in GL(n, \mathbb{F})$ is the number of *essential* variables of $f$. $\{x_{k+1}, \ldots, x_n\}$ are the *redundant* vars.

- Let $\partial_i f$ = derivative of $f$ w.r.t. $x_i$.
- Clm. $k :=$ #essential vars of $f$ $= \dim\langle \partial_1 f \ldots \partial_n f \rangle$.
- Proof~. Chain rule + examining $\langle \partial_1 f \ldots \partial_n f \rangle^{\perp}$.

# Essential & redundant variables

- Essential & redundant vars: The smallest $k$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$ for a $A \in GL(n, \mathbb{F})$ is the number of *essential* variables of $f$. $\{x_{k+1}, \ldots, x_n\}$ are the *redundant* vars.

- Let $\partial_i f$ = derivative of $f$ w.r.t. $x_i$.
- Clm. $k :=$ #essential vars of $f = \dim\langle \partial_1 f \ldots \partial_n f \rangle$.

- Carlini (2006); Kayal (2011): Given b.b.a. to $f$, we can compute a $A \in GL(n, \mathbb{F})$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$ in randomized poly-time.

# Step 1: Removing redundant variables

- $\sum\wedge\sum(k)$ circuit computing $f$:

$$f = \ell_1^{d_1} + \ldots + \ell_k^{d_k}\ ,\ \text{where } \ell_i \text{ has degree } 1.$$

- Obs. $\dim\langle \partial_1 f \ldots \partial_n f \rangle \leq k$ .

# Step 1: Removing redundant variables

- $\Sigma\wedge\Sigma(k)$ circuit computing $f$:

$$f = \ell_1^{d_1} + \ldots + \ell_k^{d_k} \text{ , where } \ell_i \text{ has degree } 1.$$

- Obs. $\dim\langle \partial_1 f \ldots \partial_n f \rangle \leq k$ .
- Compute $A$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$ ,

$$f(A\mathbf{x}) = g_1^{d_1} + \ldots + g_k^{d_k} \text{ , where } g_i = [\ell_i(A\mathbf{x})]_{\mathbf{x}\setminus\mathbf{x}_k = 0} \text{ .}$$

# Step 1: Removing redundant variables

- $\Sigma\wedge\Sigma(k)$ circuit computing $f$:

$$f = \ell_1^{d_1} + \ldots + \ell_k^{d_k} \text{ , where } \ell_i \text{ has degree } 1.$$

- Obs. $\dim\langle \partial_1 f \ldots \partial_n f \rangle \leq k$ .
- Compute $A$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$ ,

$$f(A\mathbf{x}) = g_1^{d_1} + \ldots + g_k^{d_k} \text{ , where } g_i = [\ell_i(A\mathbf{x})]_{\mathbf{x}\setminus\mathbf{x}_k = 0} \text{ .}$$

- Learn the depth-2 circuit for $f(A\mathbf{x})$.

# Step 2: Solving a polynomial system

- $\Sigma\wedge\Sigma(k)$ circuit computing $f$:

$$f = \ell_1^{d_1} + \ldots + \ell_k^{d_k}, \text{ where } \ell_i \text{ has degree } 1.$$

- Obs. $\dim\langle \partial_1 f \ldots \partial_n f \rangle \leq k$.
- Compute $A$ s.t. $f(A\mathbf{x}) \in \mathbb{F}[x_1,\ldots,x_k]$,

$$f(A\mathbf{x}) = g_1^{d_1} + \ldots + g_k^{d_k}, \text{ where } g_i = [\ell_i(A\mathbf{x})]_{\mathbf{x}\setminus\mathbf{x}_k = 0}.$$

- Learn the depth-2 circuit for $f(A\mathbf{x})$.
- Set up a polynomial system in $k^2$ variables by taking the coefficients of the $g_i$'s as variables and equating the coefficients of the LHS and the RHS. Solve the system.

# Step 2: Solving a polynomial system

- Complexity of solving a polynomial system. A polynomial system in O(1) variables can be solved in randomized poly-time over finite fields [Huang & Wong (1999)], and in deterministic poly-time over complex [Ierardi (1989)] and real numbers [Grigoriev & Vorobjov (1988)].

# Summary

- Hardness of *worst-case* reconstruction.
- A survey of known results on *worst-case* reconstruction.

- Depth-2 circuit reconstruction.
- $\sum\wedge\sum$ circuit reconstruction
  - Improper: ROABP reconstruction
  - Proper: Waring decomposition for $\sum\wedge\sum(k)$ circuits.

# Summary

- Hardness of *worst-case* reconstruction.
- A survey of known results on *worst-case* reconstruction.

- Depth-2 circuit reconstruction.
- $\sum\wedge\sum$ circuit reconstruction
  - Improper: ROABP reconstruction
  - Proper: Waring decomposition for $\sum\wedge\sum(k)$ circuits.

- In Part 2, we will discuss *average-case* reconstruction.

Thanks!