Complete Derandomization of Identity Testing of Read-Once Formulas

Daniel Minahan   Ilya Volkovich

University of Michigan

Presented by: Ramprasad Saptharishi
1 Introduction
Outline

1. Introduction
2. Previous Results
Outline

1. Introduction
2. Previous Results
3. Our Model
Outline

1. Introduction
2. Previous Results
3. Our Model
4. Read-Once Polynomials
Problem Definition

Definition (Polynomial Identity Testing (PIT))

Given a succinct representation of a polynomial $P$ in $n$ variables over a field $F$, decide efficiently if $P$ is identically 0.
Problem Definition

**Definition (Polynomial Identity Testing (PIT))**

Given a succinct representation of a polynomial $P$ in $n$ variables over a field $\mathbb{F}$, decide efficiently if $P$ is identically 0.

This requires a few things to be specified.
Problem Definition

Definition (Polynomial Identity Testing (PIT))

Given a succinct representation of a polynomial $P$ in $n$ variables over a field $\mathbb{F}$, decide efficiently if $P$ is identically 0.

This requires a few things to be specified.

- Method of representation
Problem Definition

Definition (Polynomial Identity Testing (PIT))
Given a succinct representation of a polynomial $P$ in $n$ variables over a field $F$, decide efficiently if $P$ is identically 0.

This requires a few things to be specified.

- Method of representation
- Efficiency
Definition (Polynomial Identity Testing (PIT))

Given a succinct representation of a polynomial $P$ in $n$ variables over a field $\mathbb{F}$, decide efficiently if $P$ is identically 0.

This requires a few things to be specified.

- Method of representation
- Efficiency
- Allowed actions
Succinct Representation

Definition

An arithmetic formula is a directed tree from variables \((x_1, \ldots, x_n)\) to an output. Each leaf is assigned a variable or field element and each internal node, or gate, is assigned an operation + or \(\times\).

Idea: Model small computational devices

Figure: Example of an Arithmetic Formula for \((x_1 + 1)(2x_2) + x_2 + x_3\)
Succinct Representation

**Definition**

An arithmetic formula is a directed tree from variables \((x_1, \ldots, x_n)\) to an output. Each leaf is assigned a variable or field element and each internal node, or gate, is assigned an operation \(+\) or \(\times\).

Idea: Model small computational devices

**Figure:** Example of an Arithmetic Formula for \((x_1 + 1)(2x_2) + x_2 + x_3\)
Black-Box Setting

- We cannot see the formula.
Black-Box Setting

- We cannot see the formula.
- We can only send inputs to the formula and receive an output.
Black-Box Setting

- We cannot see the formula.
- We can only send inputs to the formula and receive an output.
Black-Box Setting

- We cannot see the formula.
- We can only send inputs to the formula and receive an output.
Black-Box Setting

- **We cannot** see the formula.
- **We can only** send inputs to the formula and receive an output.

\[(x_1, \ldots, x_n) \rightarrow f \rightarrow P(x_1, \ldots, x_n)\]

- **We are allowed to query** the formula on a small extension field.
Black-Box Setting

- We cannot see the formula.
- We can only send inputs to the formula and receive an output.
- We are allowed to query the formula on a small extension field.

**Definition (Hitting Set)**

Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is called a hitting set for $C$ provided that for any $P \in C$ with $P \neq 0$, $\exists \bar{a} \in \mathcal{H}$ with $P(\bar{a}) \neq 0$. 
Black-Box Setting

- We cannot see the formula.
- We can only send inputs to the formula and receive an output.
- We are allowed to query the formula on a small extension field.

**Definition (Hitting Set)**

Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is called a hitting set for $C$ provided that for any $P \in C$ with $P \neq 0$, $\exists \bar{a} \in \mathcal{H}$ with $P(\bar{a}) \neq 0$.

- $\mathcal{H}$ gives an algorithm that runs in time $O(|\mathcal{H}|)$. 
We cannot see the formula.

We can only send inputs to the formula and receive an output.

We are allowed to query the formula on a small extension field.

**Definition (Hitting Set)**

Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is called a hitting set for $C$ provided that for any $P \in C$ with $P \neq 0$, $\exists \bar{a} \in \mathcal{H}$ with $P(\bar{a}) \neq 0$.

$\mathcal{H}$ gives an algorithm that runs in time $O(|\mathcal{H}|)$.

Goal: find a small hitting set for polynomials computed by small formulas.
Applications

Various application in Complexity Theory and Algorithm Design:
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]

- Graph Perfect Matching [MVV87]
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]
- Graph Perfect Matching [MVV87]
- Primality Testing [AKS04]
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]
- Graph Perfect Matching [MVV87]
- Primality Testing [AKS04]
- \(IP = PSPACE\) [Sha90]
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]

- Graph Perfect Matching [MVV87]

- Primality Testing [AKS04]

- IP = PSPACE [Sha90]

- PCP Theorem [AS98]
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]
- Graph Perfect Matching [MVV87]
- Primality Testing [AKS04]
- IP = PSPACE [Sha90]
- PCP Theorem [AS98]
- Geometric Complexity Theory [Mul12]
Applications

Various application in Complexity Theory and Algorithm Design:

- Fibonacci Identity:
  
  \[(a^2 + b^2)(c^2 + d^2) = (ac - bd)^2 + (ad + bc)^2\]

- Graph Perfect Matching [MVV87]
- Primality Testing [AKS04]
- IP = PSPACE [Sha90]
- PCP Theorem [AS98]
- Geometric Complexity Theory [Mul12]
- ...
Existing Algorithms

Suppose we can generate a random element of a field $\mathbb{F}$. Then we have a general PIT algorithm.
Existing Algorithms

- Suppose we can generate a random element of a field $\mathbb{F}$. Then we have a general PIT algorithm.

**Lemma (Schwartz-Zippel)**

Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ with total degree bounded by some $d \in \mathbb{N}$. Pick some finite $S \subseteq \mathbb{F}$. Then $\Pr_{\bar{x} \in S^n}[P(\bar{x}) = 0] \leq \frac{d}{|S|}$. 
Existing Algorithms

- Suppose we can generate a random element of a field $\mathbb{F}$. Then we have a general PIT algorithm.

**Lemma (Schwartz-Zippel)**

Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ with total degree bounded by some $d \in \mathbb{N}$. Pick some finite $S \subseteq \mathbb{F}$. Then $\Pr_{\bar{x} \in S^n}[P(\bar{x}) = 0] \leq \frac{d}{|S|}$.

- This does not provide a deterministic algorithm but suggests that one might exist.
Hardness Results

- [KI04]: Deterministic PIT $\iff$ super-polynomial circuit lower bounds: Boolean for NEXP or arithmetic for Permanent.
Hardness Results

- [KI04]: Deterministic PIT $\implies$ super-polynomial circuit lower bounds: Boolean for NEXP or arithmetic for Permanent.
- [Agr05]: Black-Box PIT $\implies$ Explicit exponential lower bounds for general arithmetic formulas.
Hardness Results

- [KI04]: Deterministic PIT \(\implies\) super-polynomial circuit lower bounds: Boolean for NEXP or arithmetic for Permanent.
- [Agr05]: Black-Box PIT \(\implies\) Explicit exponential lower bounds for general arithmetic formulas.
- [Agr05]: Black-Box PIT for multilinear \(\implies\) Explicit exponential lower bounds for multilinear arithmetic formulas.
[KI04]: Deterministic PIT $\implies$ super-polynomial circuit lower bounds: Boolean for NEXP or arithmetic for Permanent.

[Agr05]: Black-Box PIT $\implies$ Explicit exponential lower bounds for general arithmetic formulas.

[Agr05]: Black-Box PIT for multilinear $\implies$ Explicit exponential lower bounds for multilinear arithmetic formulas.

[AV08]: Black-Box PIT for depth-4 formulas $\implies$ Black-Box PIT for general formulas.
Hardness Results

- [KI04]: Deterministic PIT $\implies$ super-polynomial circuit lower bounds: Boolean for NEXP or arithmetic for Permanent.
- [Agr05]: Black-Box PIT $\implies$ Explicit exponential lower bounds for general arithmetic formulas.
- [Agr05]: Black-Box PIT for multilinear $\implies$ Explicit exponential lower bounds for multilinear arithmetic formulas.
- [AV08]: Black-Box PIT for depth-4 formulas $\implies$ Black-Box PIT for general formulas.

I have a truly marvelous proof of these lower bounds which this slide is too small to contain.
Hardness Results

- [KI04]: Deterministic PIT $\implies$ super-polynomial circuit lower bounds: Boolean for NEXP or arithmetic for Permanent.
- [Agr05]: Black-Box PIT $\implies$ Explicit exponential lower bounds for general arithmetic formulas.
- [Agr05]: Black-Box PIT for multilinear $\implies$ Explicit exponential lower bounds for multilinear arithmetic formulas.
- [AV08]: Black-Box PIT for depth-4 formulas $\implies$ Black-Box PIT for general formulas.

\emph{I have a truly marvelous proof of these lower bounds which this slide is too small to contain.}
Previous Results

- Sparse Polynomials: [BOT88, KS01, LV03]
Previous Results

- Sparse Polynomials: [BOT88, KS01, LV03]
- Depth-3 formulas: [DS07, KS07, KS09, SS13]
Previous Results

- Sparse Polynomials: [BOT88, KS01, LV03]
- Depth-3 formulas: [DS07, KS07, KS09, SS13]
- Depth-4 formulas: [SV11, ASSS12, KMSV13]
Previous Results

- Sparse Polynomials: [BOT88, KS01, LV03]
- Depth-3 formulas: [DS07, KS07, KS09, SS13]
- Depth-4 formulas: [SV11, ASSS12, KMSV13]
- Bounded-read formulas: [ASS12, AvMV15, SV15, FS13, AFS$^+16$]
Goal: Find an algorithm that works for certain classes of polynomials.
Multilinear and Read-Once Polynomials

Goal: Find an algorithm that works for certain classes of polynomials.

Definition (Multilinear Polynomial)
Each term of a multilinear polynomial has no variable with degree more than one. For example, $x_1x_2 + x_2x_3 + x_1x_3$. 
Multilinear and Read-Once Polynomials

Goal: Find an algorithm that works for certain classes of polynomials.

Definition (Multilinear Polynomial)
Each term of a multilinear polynomial has no variable with degree more than one. For example, $x_1x_2 + x_2x_3 + x_1x_3$.

Lemma (Hitting Set MP)
The set $\{0, 1\}^n$ is a hitting set for MPs.
Multilinear and Read-Once Polynomials

Goal: Find an algorithm that works for certain classes of polynomials.

**Definition (Multilinear Polynomial)**
Each term of a multilinear polynomial has no variable with degree more than one. For example, $x_1x_2 + x_2x_3 + x_1x_3$.

**Lemma (Hitting Set MP)**
*The set $\{0, 1\}^n$ is a hitting set for MPs.*

**Definition (Read-Once Polynomial)**
A polynomial that can be expressed by an arithmetic formula $f$ such that no variable appears more than once. For example, $x_1x_2 + x_1x_3$ but not $x_1x_2 + x_2x_3 + x_3x_1$. 
Read-Once Results

**Lemma (Random Hitting Set)**

A random set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $n^4$ is a hitting set for the class of read-once polynomials with high probability.
**Lemma (Random Hitting Set)**

A random set \( \mathcal{H} \subseteq \mathbb{F}^n \) of size \( n^4 \) is a hitting set for the class of read-once polynomials with high probability.

- This suggests that there does exist a hitting set for ROPs.
Read-Once Results

Lemma (Random Hitting Set)

A random set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $n^4$ is a hitting set the class of read-once polynomials with high probability.

- This suggests that there does exist a hitting set for ROPs.
- This does not suggest how to explicitly find such a set.
Lemma (Random Hitting Set)

A random set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $n^4$ is a hitting set the class of read-once polynomials with high probability.

- This suggests that there does exist a hitting set for ROPs.
- This does not suggest how to explicitly find such a set.
Read-Once Results

Lemma (Random Hitting Set)

A random set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $n^4$ is a hitting set for the class of read-once polynomials with high probability.

- This suggests that there does exist a hitting set for ROPs.
- This does not suggest how to explicitly find such a set.

Theorem ([SV09])

There exists an explicit hitting set for ROPs of size $n^{O(\log n)}$. 
Read-Once Results

Lemma (Random Hitting Set)

A random set \( \mathcal{H} \subseteq \mathbb{F}^n \) of size \( n^4 \) is a hitting set for the class of read-once polynomials with high probability.

- This suggests that there does exist a hitting set for ROPs.
- This does not suggest how to explicitly find such a set.

Theorem ([SV09])

There exists an explicit hitting set for ROPs of size \( n^{O(\log n)} \).

Theorem (Our Result)

There exists an explicit hitting set of size \( n^4 \) for ROPs.
Results and Implications

**Theorem (Main: Hitting Set for ROPs)**

There exists an explicit hitting set of size $n^4$ for ROPs.

**Theorem (Corollary 1 from [SV14, BHH95])**

There exists a deterministic algorithm that given a black-box access to a ROP outputs a ROF for it in polynomial time.

**Theorem (Corollary 2 from [SV15])**

For every $k \in \mathbb{N}$ there exists an explicit hitting set of size $n^{O(k)}$ for sums of $k$ ROPs.
Results and Implications

Theorem (Main: Hitting Set for ROPs)

There exists an explicit hitting set of size $n^4$ for ROPs.

Theorem (Corollary 1 from [SV14, BHH95])

There exists a deterministic algorithm that given a black-box access to a ROP outputs a ROF for it in polynomial time.
Results and Implications

Theorem (Main: Hitting Set for ROPs)

There exists an explicit hitting set of size $n^4$ for ROPs.

Theorem (Corollary 1 from [SV14, BHH95])

There exists a deterministic algorithm that given a black-box access to a ROP outputs a ROF for it in polynomial time.

Theorem (Corollary 2 from [SV15])

For every $k \in \mathbb{N}$ there exists an explicit hitting set of size $n^{O(k)}$ for sums of $k$ ROPs.
Definition (Generator)

Pick some $k \in \mathbb{N}$ with $k \ll n$. Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$. A generator for $C$ is a polynomial map $G : \mathbb{F}^k \rightarrow \mathbb{F}^n$ such that $\forall P \in C$, we have $P(G) \equiv 0 \iff P \equiv 0$. 

Lemma (Hitting Set for General Polynomials)

Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ where the degree of each variable in $P$ is bounded by some $d \in \mathbb{N}$. For any set $S \subseteq \mathbb{F}$ of size at least $d + 1$, $\exists \bar{a} \in S^n$ such that $P(\bar{a}) \neq 0$.

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $O(nd \cdot k)$. 
**Approach**

**Definition (Generator)**

Pick some $k \in \mathbb{N}$ with $k \ll n$. Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$. A generator for $C$ is a polynomial map $G : \mathbb{F}^k \to \mathbb{F}^n$ such that $\forall P \in C$, we have $P(G) \equiv 0 \iff P \equiv 0$.

**Lemma (Hitting Set for General Polynomials)**

Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ where the degree of each variable in $P$ is bounded by some $d \in \mathbb{N}$. For any set $S \subseteq \mathbb{F}$ of size at least $d + 1$, $\exists \bar{a} \in S^n$ such that $P(\bar{a}) \neq 0$. 
Approach

Definition (Generator)

Pick some $k \in \mathbb{N}$ with $k \ll n$. Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$. A generator for $C$ is a polynomial map $G : \mathbb{F}^k \to \mathbb{F}^n$ such that $\forall P \in C$, we have $P(G) \equiv 0 \iff P \equiv 0$.

Lemma (Hitting Set for General Polynomials)

Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ where the degree of each variable in $P$ is bounded by some $d \in \mathbb{N}$. For any set $S \subseteq \mathbb{F}$ of size at least $d + 1$, $\exists \bar{a} \in S^n$ such that $P(\bar{a}) \neq 0$.

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generator to Hitting Set)

Suppose the individual degrees of each component function of $G$ are bounded by some $d \in \mathbb{N}$. Then we can decide if $P(G) \equiv 0$ in time $(nd)^{O(k)}$. 

Lemma (From Generate...
Previous Generator

Definition (The Generator of [SV09])

Pick some \( n \in \mathbb{N} \). Pick distinct \( \alpha_1, \ldots, \alpha_n \in \mathbb{F} \) and let \( \mu_i(w) \) be the Lagrange Interpolation Polynomial that evaluates to 0 for \( \alpha_j \) with \( j \neq i \) and evaluates to 1 at \( \alpha_i \). Then we define:
Previous Generator

Definition (The Generator of [SV09])

Pick some $n \in \mathbb{N}$. Pick distinct $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ and let $\mu_i(w)$ be the Langrange Interpolation Polynomial that evaluates to 0 for $\alpha_j$ with $j \neq i$ and evaluates to 1 at $\alpha_i$. Then we define:

$G_{n,1} : \mathbb{F}^2 \rightarrow \mathbb{F}^n$ by $(y, z) \rightarrow (\mu_1(y)z, \ldots, \mu_n(y)z)$. 

Previous Generator

Definition (The Generator of [SV09])

Pick some \( n \in \mathbb{N} \). Pick distinct \( \alpha_1, \ldots, \alpha_n \in \mathbb{F} \) and let \( \mu_i(w) \) be the Langrange Interpolation Polynomial that evaluates to 0 for \( \alpha_j \) with \( j \neq i \) and evaluates to 1 at \( \alpha_i \). Then we define:

\[
G_{n,1} : \mathbb{F}^2 \to \mathbb{F}^n \text{ by } (y, z) \to (\mu_1(y)z, \ldots, \mu_n(y)z).
\]

For \( t \in \mathbb{N}, \ G_{n,t} : \mathbb{F}^{2t} \to \mathbb{F}^n \text{ as } G_{n,t} = \sum_{i=1}^{t} G_{n,1}(y_i, z_i)\]
Definition (The Generator of [SV09])

Pick some $n \in \mathbb{N}$. Pick distinct $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ and let $\mu_i(w)$ be the Langrange Interpolation Polynomial that evaluates to 0 for $\alpha_j$ with $j \neq i$ and evaluates to 1 at $\alpha_i$. Then we define:

$G_{n,1} : \mathbb{F}^2 \to \mathbb{F}^n$ by $(y, z) \to (\mu_1(y)z, \ldots, \mu_n(y)z)$.

For $t \in \mathbb{N}$, $G_{n,t} : \mathbb{F}^{2t} \to \mathbb{F}^n$ as $G_{n,t} = \sum_{i=1}^{t} G_{n,1}(y_i, z_i)$.

Theorem ([SV09] - Quasi-Polynomial PIT)

The polynomial map $G_{n, \log n}$ is a generator for ROPs on $n$ variables.
Previous Generator

**Definition (The Generator of [SV09])**

Pick some $n \in \mathbb{N}$. Pick distinct $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ and let $\mu_i(w)$ be the Langrange Interpolation Polynomial that evaluates to 0 for $\alpha_j$ with $j \neq i$ and evaluates to 1 at $\alpha_i$. Then we define:

$G_{n,1}: \mathbb{F}^2 \to \mathbb{F}^n$ by $(y, z) \mapsto (\mu_1(y)z, \ldots, \mu_n(y)z)$.

For $t \in \mathbb{N}$, $G_{n,t}: \mathbb{F}^{2t} \to \mathbb{F}^n$ as $G_{n,t} = \sum_{i=1}^{t} G_{n,1}(y_i, z_i)$

**Theorem ([SV09] - Quasi-Polynomial PIT)**

The polynomial map $G_{n,\log n}$ is a generator for ROPs on $n$ variables.

**Theorem (Our Result - Polynomial PIT)**

The polynomial map $G_{n,1}$ is a generator for ROPs on $n$ variables.
High Level Idea

**Definition (Homogenous Polynomial)**

A polynomial $P \in \mathbb{F}[x_1, \ldots, x_n]$ is called *homogeneous* if every term in the polynomial has the same total degree.

**Lemma (Generator for Homogeneous ROPs)**

$G_{n,1}$ is a generator for homogeneous ROPs.

**Lemma (From Homogeneous ROPs to General ROPs)**

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a ROP, then so is $H_{\deg(P)}(P)$.

**Corollary (From Homogeneous ROPs to ROPs)**

$G_{n,1}$ is a generator for (general) ROPs.
High Level Idea

Definition (Homogenous Polynomial)
A polynomial \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is called homogeneous if every term in the polynomial has the same total degree.

Lemma (Generator for Homogeneous ROPs)
\( G_{n,1} \) is a generator for homogeneous ROPs.

Lemma (From Homogeneous ROPs to General ROPs)
If \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is a ROP, then so is \( H_{\deg(P)}(P) \).

Corollary (From Homogeneous ROPs to ROPs)
\( G_{n,1} \) is a generator for (general) ROPs.
High Level Idea

**Definition (Homogenous Polynomial)**

A polynomial $P \in \mathbb{F}[x_1, \ldots, x_n]$ is called *homogeneous* if every term in the polynomial has the same total degree.

**Lemma (Generator for Homogeneous ROPs)**

$G_{n,1}$ is a generator for homogeneous ROPs.

**Lemma (From Homogeneous ROPs to General ROPs)**

*If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a ROP, then so is $H_{\deg(P)}(P)$.***

**Corollary (From Homogeneous ROPs to ROPs)**

$G_{n,1}$ is a generator for (general) ROPs.
### High Level Idea

#### Definition (Homogeneous Polynomial)
A polynomial \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is called *homogeneous* if every term in the polynomial has the same total degree.

#### Lemma (Generator for Homogeneous ROPs)
\( G_{n,1} \) is a generator for homogeneous ROPs.

#### Lemma (From Homogeneous ROPs to General ROPs)
If \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is a ROP, then so is \( H_{\deg(P)}(P) \).

#### Corollary (From Homogeneous ROPs to ROPs)
\( G_{n,1} \) is a generator for (general) ROPs.
### High Level Idea

**Definition (Homogenous Polynomial)**
A polynomial \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is called *homogeneous* if every term in the polynomial has the same total degree.

**Lemma (Generator for Homogeneous ROPs)**
\( G_{n,1} \) is a generator for homogeneous ROPs.

**Lemma (From Homogeneous ROPs to General ROPs)**
If \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is a ROP, then so is \( H_{\deg(P)}(P) \).

**Corollary (From Homogeneous ROPs to ROPs)**
\( G_{n,1} \) is a generator for (general) ROPs.
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{th}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$. 
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{th}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$.

**Lemma (Homogenous ROP Structural Lemma)**

*If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then $\exists P_1, P_2$ non-constant, variable-disjoint homogeneous ROPs s.t:*  

1. $P = P_1 \cdot P_2$  
2. $P = P_1 + P_2$
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{\text{th}}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$.

Lemma (Homogenous ROP Structural Lemma)

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then

$\exists P_1, P_2$ non-constant, variable-disjoint homogeneous ROPs s.t:

1. $P = P_1 \cdot P_2$
2. $P = P_1 + P_2$

By induction on $n$: 

...
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{\text{th}}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$.

**Lemma (Homogenous ROP Structural Lemma)**

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then $\exists P_1, P_2$ non-constant, variable-disjoint homogeneous ROPs s.t:

1. $P = P_1 \cdot P_2$
2. $P = P_1 + P_2$

By induction on $n$:

0. Linear Case (Base case): $P = c_1x_1 + \cdots + c_nx_n$. 
**Lemma (Homogenous ROP Structural Lemma)**

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then there exist non-constant, variable-disjoint homogeneous ROPs $P_1, P_2$ such that:

1. $P = P_1 \cdot P_2$
2. $P = P_1 + P_2$

By induction on $n$:

- **Linear Case (Base case):** $P = c_1x_1 + \cdots + c_nx_n$. Pick $i$ such that $c_i \neq 0$. Fix $y = \alpha_i$, then $P(G_{n,1}(\alpha_i, z)) = c_iz$.
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{th}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$.

**Lemma (Homogenous ROP Structural Lemma)**

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then there exist $P_1, P_2$ non-constant, variable-disjoint homogeneous ROPs such that:

1. $P = P_1 \cdot P_2$
2. $P = P_1 + P_2$

By induction on $n$:

0. **Linear Case (Base case):** $P = c_1 x_1 + \cdots + c_n x_n$. Pick $i$ such that $c_i \neq 0$. Fix $y = \alpha_i$, then $P(G_{n,1}(\alpha_i, z)) = c_i z$.

1. **Multiplicative Case:** $P = P_1 \cdot P_2$, where $P_i$ depends on $< n$ variables.
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{th}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$.

**Lemma (Homogenous ROP Structural Lemma)**

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then

$\exists P_1, P_2$ non-constant, variable-disjoint homogeneous ROPs s.t:

1. $P = P_1 \cdot P_2$
2. $P = P_1 + P_2$

By induction on $n$:

0. Linear Case (Base case): $P = c_1x_1 + \cdots + c_nx_n$. Pick $i$ such that $c_i \neq 0$. Fix $y = \alpha_i$, then $P(G_{n,1}(\alpha_i, z)) = c_i z$.

4. Multiplicative Case: $P = P_1 \cdot P_2$, where $P_i$ depends on $< n$ variables. By the inductive hypothesis, $P_1(G_{n,1}), P_2(G_{n,1}) \neq 0$. 
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick \( \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F} \). Let \( \mu_i(y) \) the \( i^{th} \) Lagrange Interpolation Polynomial. Let \( G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z) \).

**Lemma (Homogenous ROP Structural Lemma)**

*If \( P \in \mathbb{F}[x_1, \ldots, x_n] \) is a homogeneous ROP with \( n \geq 2 \), then \( \exists P_1, P_2 \) non-constant, variable-disjoint homogeneous ROPs s.t:*

1. \( P = P_1 \cdot P_2 \)
2. \( P = P_1 + P_2 \)

By induction on \( n \):

1. **Linear Case (Base case):** \( P = c_1 x_1 + \cdots + c_n x_n \). Pick \( i \) such that \( c_i \neq 0 \). Fix \( y = \alpha_i \), then \( P(G_{n,1}(\alpha_i, z)) = c_i z \).
2. **Multiplicative Case:** \( P = P_1 \cdot P_2 \), where \( P_i \) depends on \( < n \) variables. By the inductive hypothesis, \( P_1(G_{n,1}) \), \( P_2(G_{n,1}) \neq 0 \). Therefore, \( P(G_{n,1}) = P_1(G_{n,1}) \cdot P_2(G_{n,1}) \neq 0 \).
$G_{n,1}$ is a Generator for Homogeneous ROPs: Proof Sketch

Pick $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$. Let $\mu_i(y)$ the $i^{th}$ Lagrange Interpolation Polynomial. Let $G_{n,1}(y, z) = (\mu_1(y)z, \ldots, \mu_n(y)z)$.

**Lemma (Homogenous ROP Structural Lemma)**

If $P \in \mathbb{F}[x_1, \ldots, x_n]$ is a homogeneous ROP with $n \geq 2$, then

$\exists P_1, P_2$ non-constant, variable-disjoint homogeneous ROPs s.t:

1. $P = P_1 \cdot P_2$
2. $P = P_1 + P_2$

By induction on $n$:

1. **Linear Case (Base case):** $P = c_1x_1 + \cdots + c_nx_n$. Pick $i$ such that $c_i \neq 0$. Fix $y = \alpha_i$, then $P(G_{n,1}(\alpha_i, z)) = c_iz$.
2. **Multiplicative Case:** $P = P_1 \cdot P_2$, where $P_i$ depends on $< n$ variables. By the inductive hypothesis, $P_1(G_{n,1}), P_2(G_{n,1}) \neq 0$. Therefore, $P(G_{n,1}) = P_1(G_{n,1}) \cdot P_2(G_{n,1}) \neq 0$.
3. **Additive Case:** Main technical contribution of the paper.
Theorem (Black-Box PIT for ROFs)

There exists a polynomial-time black-box PIT algorithm for read-once formulas.

Open questions:
- Polynomial-time black-box PIT algorithm for read-$k$ formulas?
- [AvMV15]: quasi-polynomial-time black-box PIT algorithm for read-$k$ formulas.
- Our results + [SV15]: Polynomial-time black-box PIT algorithm for sum of two read-once formulas.
- Polynomial-time black-box PIT algorithm for read-twice formulas?
### Conclusion & Open Questions

<table>
<thead>
<tr>
<th>Theorem (Black-Box PIT for ROFs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>There exists a polynomial-time black-box PIT algorithm for read-once formulas.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theorem (Reconstruction for ROFs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>There exists a polynomial-time reconstruction algorithm for read-once formulas.</td>
</tr>
</tbody>
</table>
Conclusion & Open Questions

Theorem (Black-Box PIT for ROFs)

There exists a polynomial-time black-box PIT algorithm for read-once formulas.

Theorem (Reconstruction for ROFs)

There exists a polynomial-time reconstruction algorithm for read-once formulas.

Open questions:

- Polynomial-time black-box PIT algorithm for read-$k$ formulas?
Conclusion & Open Questions

Theorem (Black-Box PIT for ROFs)

There exists a polynomial-time black-box PIT algorithm for read-once formulas.

Theorem (Reconstruction for ROFs)

There exists a polynomial-time reconstruction algorithm for read-once formulas.

Open questions:

- Polynomial-time black-box PIT algorithm for read-$k$ formulas?
- [AvMV15]: quasi-polynomial-time black-box PIT algorithm for read-$k$ formulas.
Conclusion & Open Questions

Theorem (Black-Box PIT for ROFs)

There exists a polynomial-time black-box PIT algorithm for read-once formulas.

Theorem (Reconstruction for ROFs)

There exists a polynomial-time reconstruction algorithm for read-once formulas.

Open questions:

- Polynomial-time black-box PIT algorithm for read-$k$ formulas?
- \cite{AvMV15}: quasi-polynomial-time black-box PIT algorithm for read-$k$ formulas.
- Our results $+$ \cite{SV15}: Polynomial-time black-box PIT algorithm for sum of two read-once formulas.
Conclusion & Open Questions

Theorem (Black-Box PIT for ROFs)

There exists a polynomial-time black-box PIT algorithm for read-once formulas.

Theorem (Reconstruction for ROFs)

There exists a polynomial-time reconstruction algorithm for read-once formulas.

Open questions:

- Polynomial-time black-box PIT algorithm for read-\(k\) formulas?
- [AvMV15]: quasi-polynomial-time black-box PIT algorithm for read-\(k\) formulas.
- Our results + [SV15]: Polynomial-time black-box PIT algorithm for sum of two read-once formulas.
- Polynomial-time black-box PIT algorithm for read-twice formulas?
Thank you!
Identity testing and lower bounds for read-k oblivious algebraic branching programs.

M. Agrawal.
Proving lower bounds via pseudo-random generators.

M. Agrawal, N. Kayal, and N. Saxena.
Primes is in P.

S. Arora and S. Safra.
Probabilistic checking of proofs: A new characterization of NP.

M. Agrawal, C. Saha, and N. Saxena.


M. Agrawal and V. Vinay.

Derandomizing polynomial identity testing for multilinear constant-read formulae.
Introduction

Previous Results

Our Model

Read-Once Polynomials

Our Results


N. H. Bshouty, T. R. Hancock, and L. Hellerstein.
Learning arithmetic read-once formulas.

M. Ben-Or and P. Tiwari.
A deterministic algorithm for sparse multivariate polynomial interpolation.

Z. Dvir and A. Shpilka.
Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits.

M. Forbes and A. Shpilka.
Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs.
V. Kabanets and R. Impagliazzo.
Derandomizing polynomial identity tests means proving circuit lower bounds.

Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in.

A. Klivans and D. Spielman.
Randomness efficient identity testing of multivariate polynomials.

N. Kayal and N. Saxena.

N. Kayal and S. Saraf.

R. J. Lipton and N. K. Vishnoi.

K. Mulmuley.
Geometric complexity theory v: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of noether's normalization lemma.


N. Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits.
A. Shpilka and I. Volkovich.
Improved polynomial identity testing for read-once formulas.

S. Saraf and I. Volkovich.
Blackbox identity testing for depth-4 multilinear circuits.

A. Shpilka and I. Volkovich.
On reconstruction and testing of read-once formulas.

A. Shpilka and I. Volkovich.
Read-once polynomial identity testing.