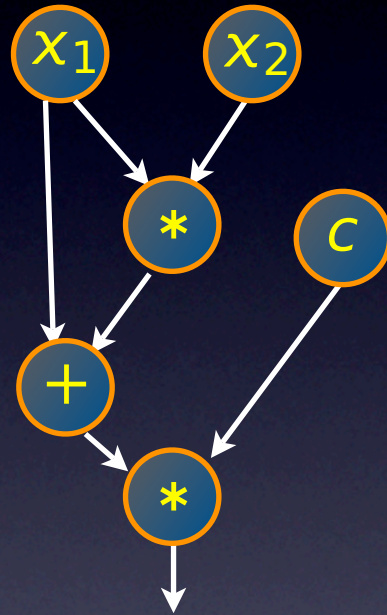


Arithmetic Circuit Identity Testing for Sparse Polynomials

Speaker: Moritz Hardt

Joint work with Markus Bläser,
Saarland University

Quick Reminder



Arithmetic Circuit

- division-free
- size = #gates

$$2x^4y + 2xy^4$$

Sparse Polynomial:
few nonzero monomials

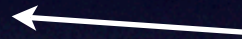
Arithmetic Circuit Identity Testing (ACIT)

Given an arithmetic circuit of size s computing a polynomial P , determine if P is identically zero.

Plan

1. Study the **univariate** case first.

Deterministic algorithm that exploits the ***sparsity*** of the input polynomial

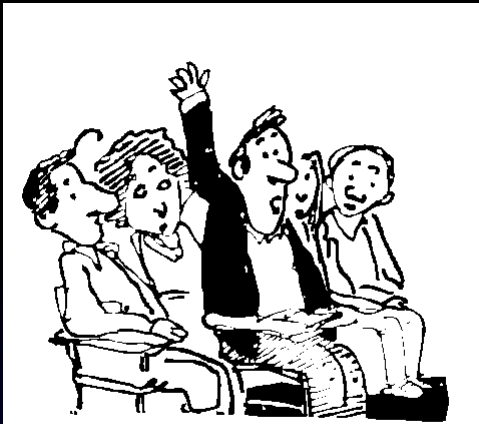


2. Randomize!

3. Reductions from **multivariate** to univariate.



How many roots can a *nonzero* univariate polynomial have?



How many roots can a *nonzero* univariate polynomial have?

d (degree)



How many roots can a *nonzero* univariate polynomial have?

d (degree)

⇒ Naive **$\text{poly}(d,s)$** -time algorithm



How many roots can a *nonzero* univariate polynomial have?

d (degree)

↪ Naive **$\text{poly}(d,s)$** -time algorithm

Can we say anything better?

Fact: Real-valued polynomial with at most m nonzero monomials has at most m positive roots.

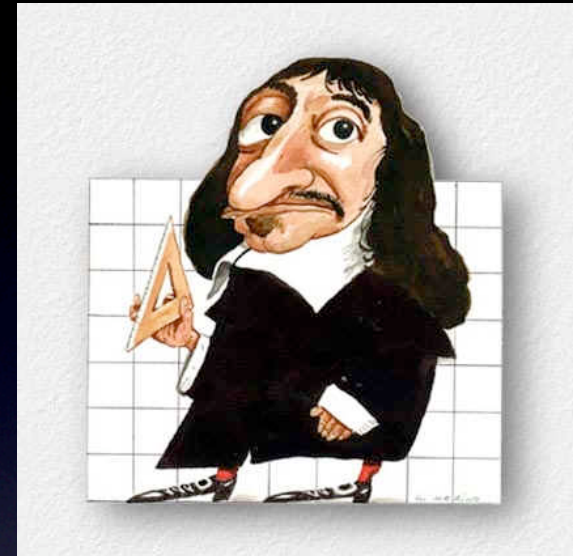
Fact: Real-valued polynomial with at most m nonzero monomials has at most m positive roots.

Proof: Rule of Signs.



Fact: Real-valued polynomial with at most m nonzero monomials has at most m positive roots.

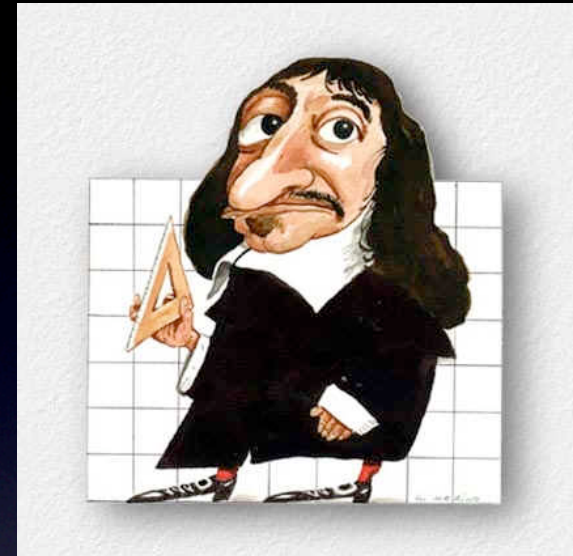
Proof: Rule of Signs.



Naive $\text{poly}(m,s)$ -time algorithm (over the reals)?

Fact: Real-valued polynomial with at most m nonzero monomials has at most m positive roots.

Proof: Rule of Signs.

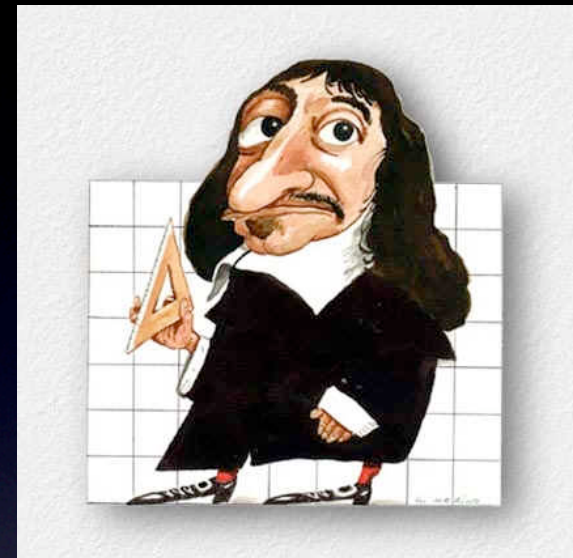


Naive $\text{poly}(m,s)$ -time algorithm (over the reals)?

No! Why not?

Fact: Real-valued polynomial with at most m nonzero monomials has at most m positive roots.

Proof: Rule of Signs.



Naive $\text{poly}(m,s)$ -time algorithm (over the reals)?

No! Why not?

Can we still do it?

Previous Work

Lipton, Vishnoi '03.

- $\text{poly}(m, n, \log d, H)$ runtime over the integers

Klivans, Spielman '01.

- $O(\log(mnd))$ random bits

Karpinski, Shparlinski '96

- $\text{poly}(m, n, \log d)$ runtime over finite fields,
(but either a lot of randomness or dependence on the characteristic)

Previous Work

Lipton, Vishnoi '03.

- $\text{poly}(m, n, \log d, H)$ runtime over the integers

Klivans, Spielman '01.

- $O(\log(mnd))$ random bits

Karpinski, Shparlinski '96

- $\text{poly}(m, n, \log d)$ runtime over finite fields,
(but either a lot of randomness or dependence on the characteristic)

Further related work on sparse polynomial interpolation...

Previous Work

Lipton, Vishnoi '03.

- $\text{poly}(m, n, \log d, H)$ runtime over the integers

Klivans, Spielman '01.

- $O(\log(mnd))$ random bits

Karpinski, Shparlinski '96

- $\text{poly}(m, n, \log d)$ runtime over finite fields,
(but either a lot of randomness or dependence on the characteristic)

Further related work on sparse polynomial interpolation...

Not related to sparsity:

Agrawal, Biswas '99

- Randomness-efficient test using arithmetic circuits

Our result

- Deterministic test using $\text{poly}(m,s)$ ring operations over any integral domain
 - ▶ amounts to runtime $\text{poly}(m,n,\log d,H)$ over integers or rationals, for instance
- Lower exponents in the runtime with fewer random bits
- Very simple algorithm

Algorithm

Given an arithmetic circuit computing a univariate polynomial P , verify

$$P(x) \equiv 0 \pmod{x^p - 1}$$

for *sufficiently* many primes p .

Given a univariate polynomial P , verify


$$P(x) \equiv 0 \pmod{x^p - 1}$$

for *sufficiently* many primes p .



What is *sufficient*?

Given a univariate polynomial P , verify
 $P(x) \equiv 0 \pmod{x^p - 1}$
for sufficiently many primes p .



What is *sufficient*?

Claim: Given polynomial $P(x)$, degree d , $m > 0$ nonzero monomials, then there are less than $m \log d$ primes p for which $P(x) \equiv 0 \pmod{x^p - 1}$.

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are less than $m \log d$ primes p for which $P(x) \equiv 0 \pmod{x^p - 1}$.

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are less than $m \log d$ primes p for which $P(x) \equiv 0 \pmod{x^p - 1}$.

1st Idea: We're reducing degrees mod p .

$$x^{kp+r} \equiv x^r \pmod{x^p - 1} \quad r < p, k \geq 0$$

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are **less than $m \log d$ primes p** for which $P(x) \equiv 0 \pmod{x^p - 1}$.

1st Idea: We're reducing degrees mod p .

$$x^{kp+r} \equiv x^r \pmod{x^p - 1} \quad r < p, k \geq 0$$

2nd Idea: Count prime factors!

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are **less than $m \log d$ primes p** for which $P(x) \equiv 0 \pmod{x^p - 1}$.

1st Idea: We're reducing degrees mod p .

$$x^{kp+r} \equiv x^r \pmod{x^p - 1} \quad r < p, k \geq 0$$

2nd Idea: Count prime factors!

Fix two monomials $x^d, x^{d'}$.

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are **less than $m \log d$ primes p** for which $P(x) \equiv 0 \pmod{x^p - 1}$.

1st Idea: We're reducing degrees mod p .

$$x^{kp+r} \equiv x^r \pmod{x^p - 1} \quad r < p, k \geq 0$$

2nd Idea: Count prime factors!

Fix two monomials $x^d, x^{d'}$.

If $d \equiv d' \pmod{p}$, we have $p | (d - d')$.

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are **less than $m \log d$ primes p** for which $P(x) \equiv 0 \pmod{x^p - 1}$.

1st Idea: We're reducing degrees mod p .

$$x^{kp+r} \equiv x^r \pmod{x^p - 1} \quad r < p, k \geq 0$$

2nd Idea: Count prime factors!

Fix two monomials $x^d, x^{d'}$.

If $d \equiv d' \pmod{p}$, we have $p | (d - d')$.

But, $|d - d'| \leq d$.

Claim: Given polynomial $P(x)$, degree $d, m > 0$ nonzero monomials, then there are **less than $m \log d$ primes p** for which $P(x) \equiv 0 \pmod{x^p - 1}$.

1st Idea: We're reducing degrees mod p .

$$x^{kp+r} \equiv x^r \pmod{x^p - 1} \quad r < p, k \geq 0$$

2nd Idea: Count prime factors!

Fix two monomials $x^d, x^{d'}$.

If $d \equiv d' \pmod{p}$, we have $p | (d - d')$.

But, $|d - d'| \leq d$.

At most **$\log d$ bad primes** per pair of monomials!

Given a univariate polynomial P , verify

$$P(x) \equiv 0 \pmod{x^p - 1}$$

for sufficiently many primes p .

How?

Given a univariate polynomial P , verify

$$P(x) \equiv 0 \pmod{x^p - 1}$$

for sufficiently many primes p .

How?

Claim: If P is given as an arithmetic circuit of size s ,
we can verify $P(x) \equiv 0 \pmod{x^k - 1}$
with $\tilde{O}(sk)$ ring operations (over an integral domain).

Claim: If P is given as an arithmetic circuit of size s ,
we can verify $P(x) \equiv 0 \pmod{x^k - 1}$
with $\tilde{O}(sk)$ ring operations (over an integral domain).

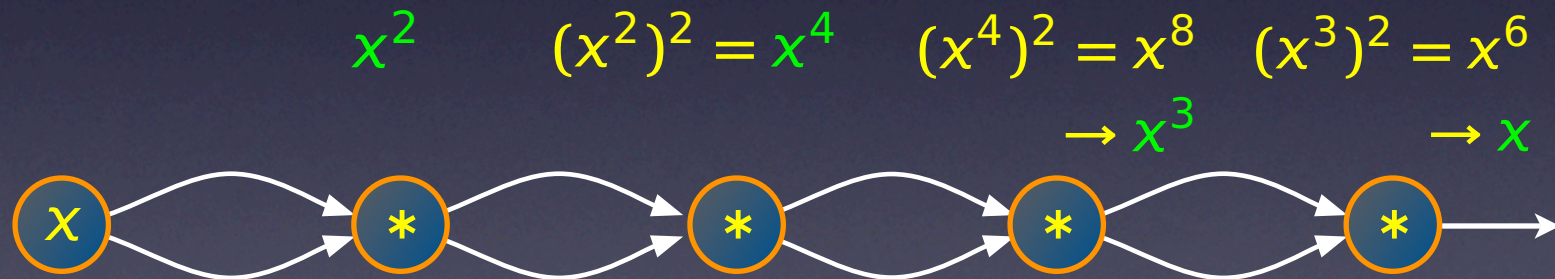
Claim: If P is given as an arithmetic circuit of size s ,
we can verify $P(x) \equiv 0 \pmod{x^k - 1}$
with $\tilde{O}(sk)$ ring operations (over an integral domain).

Idea: Compute the whole polynomial in its
reduced form.

Claim: If P is given as an arithmetic circuit of size s , we can verify $P(x) \equiv 0 \pmod{x^k - 1}$ with $\tilde{O}(sk)$ ring operations (over an integral domain).

Idea: Compute the whole polynomial in its reduced form.

Example ($k=5$)



Invariant: Degree k polynomial at each gate

So far: Deterministic algorithm

runtime $\text{poly}(m,s)$

Next step: **Randomization**

So far: Deterministic algorithm

runtime $\text{poly}(m, s)$

Next step: **Randomization**

Goal: $\tilde{O}(\log(ms))$ random bits,
speed up runtime as much as possible,
hopefully $\text{poly}(s)$.

↖
No m , here.

Given a univariate polynomial P , verify

$$P(x) \equiv 0 \pmod{x^p - 1}$$

for ~~sufficiently many~~ primes p .

a large enough **random** prime p .

Given a univariate polynomial P , verify

$$P(x) \equiv 0 \pmod{x^p - 1}$$

for ~~sufficiently many primes~~ p .

a large enough **random** prime p .

OK, $O(\log p) = \tilde{O}(m \log d)$ random bits.

Given a univariate polynomial P , verify

$$P(x) \equiv 0 \pmod{x^p - 1}$$

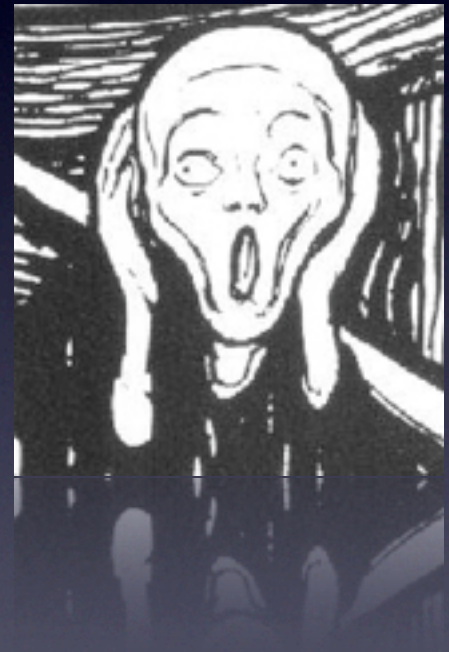
for a large enough random prime p .

Are we done?

Given a univariate polynomial P , verify
 $P(x) \equiv 0 \pmod{x^p - 1}$
for a large enough random prime p .

Are we done?

Runtime
“poly(p, s)”

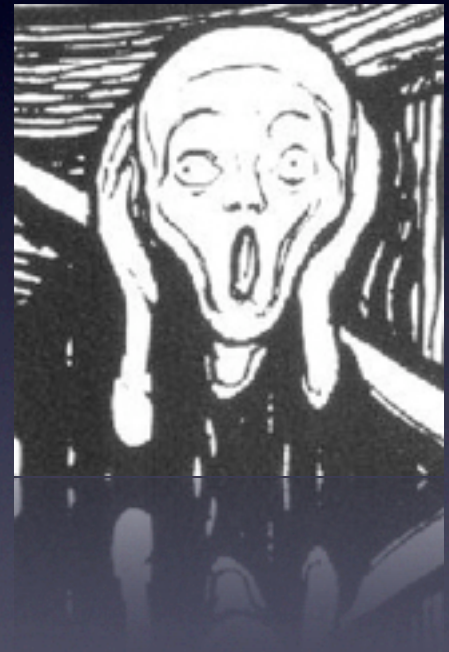


Given a univariate polynomial P , verify
 $P(x) \equiv 0 \pmod{x^p - 1}$
for a large enough random prime p .

Are we done?

Runtime
“poly(p, s)”

Not efficient general, but still nice speed up.



Open Problem

Can we verify $P(x) \equiv 0 \pmod{x^p - 1}$
with $\text{poly}(s)$ ring operations
and $O(\log p)$ random bits ?

From multivariate to univariate

Random bits	s	n	m	d
none	not much more	1	m	$n(d+1)^n$
$\tilde{O}(\log(mn \log d))$	not much more	1	$\leq m$ sound w.h.p.	$(mnd)^k$

Deterministic Reduction

Given:

$$P(x_1, x_2, \dots, x_n)$$

- max variable degree d

Deterministic Reduction

Given:

$$P(x_1, x_2, \dots, x_n)$$

- max variable degree d

Substitute

$$x_i := x^{(d+1)^{i-1}}$$

Deterministic Reduction

Given:


$$P(x_1, x_2, \dots, x_n)$$

- max variable degree d

Substitute

$$x_i := x^{(d+1)^{i-1}}$$

Circuit of size
 $n \log d$
repeated squaring



Randomized Reduction

- Uses the mapping by Klivans and Spielman '01
- Adds a step of chinese remaindering to it to further decrease # random bits:

$$O(\log(mnd)) \longrightarrow \tilde{O}(\log(mn \log d))$$

Results in Detail

Deterministic	Randomized
$\tilde{O}((mn \log d)^2 S)$ operations $S = s + n^2 \log d$	$\tilde{O}(m \log(mnd) T)$ operations with $\tilde{O}(\log(mn \log d))$ random bits $T = s + n \log(mnd)$

Conclusion

- What's the “significant” parameter in identity testing? **Degree** or **Sparsity**?
- Classical results say *degree*
- Our result argues for *sparsity*
- Efficient randomized analogon
($\tilde{O}(\log(ms))$ random bits) still missing!

Thank you.

Closer look at the open question

Suppose $P \in \mathbb{C}[x]$.

Fact: $P(x) \equiv 0 \pmod{x^p - 1}$

if and only if

$P(r) = 0$ for all p -th roots of unity r .

Closer look at the open question

Suppose $P \in \mathbb{C}[x]$.

Fact: $P(x) \equiv 0 \pmod{x^p - 1}$

if and only if

$P(r) = 0$ for all p -th roots of unity r .

What about picking a **random** root?

Closer look at the open question

Suppose $P \in \mathbb{C}[x]$.

Fact: $P(x) \equiv 0 \pmod{x^p - 1}$

if and only if

$P(r) = 0$ for all p -th roots of unity r .

What about picking a **random** root?

Idea: A polynomial that is zero everywhere except on one point cannot be **not sparse**!

“Uncertainty Principle”

Closer look at the open question

Suppose $P \in \mathbb{C}[x]$.

Fact: $P(x) \equiv 0 \pmod{x^p - 1}$

if and only if

$P(r) = 0$ for all p -th roots of unity r .

What about picking a **random** root?

Idea: A polynomial that is zero everywhere except on one point cannot be **not sparse**!

“Uncertainty Principle”

Unfortunately, parameters too weak per se.

But maybe, can mimic Indyk '07 on polynomials...