Lecture 35

$\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$, PIT

$\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$

$\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$

Theorem: $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

$\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$

Theorem: $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

Proof:

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

ZPP = RP ∩ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: ZPP $\subseteq$ RP $\cap$ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

ZPP = RP $\cap$ coRP

Theorem: ZPP = RP $\cap$ coRP.

Proof: ZPP $\subseteq$ RP $\cap$ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

ZPP = RP ∩ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: ZPP $\subseteq$ RP $\cap$ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$:

ZPP = RP $\cap$ coRP

Theorem: ZPP = RP $\cap$ coRP.

Proof: ZPP $\subseteq$ RP $\cap$ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

$1)$ Runs M for $3.p(n)$ steps.

$2)$ If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$:
- When $x \in L$:

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$:

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$: Probability of $M'(x) = 0$

ZPP $=$ RP $\cap$ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: ZPP $\subseteq$ RP $\cap$ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$: Probability of $M'(x) = 0$ $=$ Probability that M runs for $> 3.p(n)$ steps

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$: Probability of $M'(x) = 0$ $=$ Probability that M runs for $> 3.p(n)$ steps

$$\leq \textbf{Pr}[T_x \geq 3.p(n)]$$

ZPP = RP $\cap$ coRP

Theorem: ZPP = RP $\cap$ coRP.

Proof: ZPP $\subseteq$ RP $\cap$ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$: Probability of $M'(x) = 0$ $=$ Probability that M runs for $> 3.p(n)$ steps

$$\leq \ \textbf{Pr}[T_x \geq 3.p(n)] \ \leq \ \textbf{Ex}(T_x) \ / \ 3.p(n)$$

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$: Probability of $M'(x) = 0$ = Probability that M runs for $> 3.p(n)$ steps

$$\leq \textbf{Pr}[T_x \geq 3.p(n)] \leq \textbf{Ex}(T_x) \,/\, 3.p(n) \leq 1/3$$

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: ZPP ⊆ RP ∩ coRP:

Let $L \in$ **ZPP** and M be a PTM that decides L and runs in expected polytime, say $p(n)$.

RP algorithm for L: Consider the PTM M' that on input x:

1) Runs M for $3.p(n)$ steps.

2) If M outputs something, then M' outputs the same. Else M' outputs 0.

Correctness analysis:

- When $x \notin L$: $M'(x) = 0$ with probability 1.
- When $x \in L$: Probability of $M'(x) = 0$ $=$ Probability that M runs for $> 3.p(n)$ steps

$$\leq \textbf{Pr}[T_x \geq 3.p(n)] \leq \textbf{Ex}(T_x) / 3.p(n) \leq 1/3$$

…

$\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$

Theorem: $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

Proof:

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

ZPP = RP $\cap$ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: RP $\cap$ coRP $\subseteq$ ZPP:

Let $L \in$ RP, coRP.

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP $=$ RP $\cap$ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: RP $\cap$ coRP $\subseteq$ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

ZPP $=$ RP $\cap$ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: RP $\cap$ coRP $\subseteq$ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

ZPP $=$ RP $\cap$ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: RP $\cap$ coRP $\subseteq$ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

ZPP $=$ RP $\cap$ coRP

Theorem: ZPP $=$ RP $\cap$ coRP.

Proof: RP $\cap$ coRP $\subseteq$ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

Probability of M stopping in $2.p(n)\,.\,i$ time

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

$$\text{Probability of } M \text{ stopping in } 2.p(n)\,.\,i \text{ time } = \frac{2}{3}\cdot\frac{1}{3}^{i-1}$$

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

Probability of M stopping in $2.p(n)\,.\,i$ time $= \frac{2}{3} \cdot \frac{1}{3}^{i-1}$

M's expected running time

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

Probability of M stopping in $2.p(n)\,.\,i$ time $= \dfrac{2}{3} \cdot \dfrac{1}{3}^{i-1}$

M's expected running time $= \displaystyle\sum_{i \geq 1} 2.p(n)\,.\,i\,.\left(\frac{2}{3} \cdot \frac{1}{3}^{i-1}\right)$

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

Probability of M stopping in $2.p(n)\,.\,i$ time $= \dfrac{2}{3} \cdot \dfrac{1}{3}^{i-1}$

M's expected running time $= \displaystyle\sum_{i \geq 1} 2.p(n)\,.\,i\,.\left(\frac{2}{3} \cdot \frac{1}{3}^{i-1}\right) = O(p(n))$

ZPP = RP ∩ coRP

Theorem: ZPP = RP ∩ coRP.

Proof: RP ∩ coRP ⊆ ZPP:

Let $L \in$ **RP**, **coRP**. Let A and B be L's **RP** and **coRP** machines with $p(n)$ runtime.

ZPP machine M for L on input x:

1) Runs A on x. If A outputs 1, then M also outputs 1.

2) Runs B on x. If B outputs 0, then M also outputs 0.

3) Otherwise, repeat steps 1 and 2.

M always gives the right answer.

Time analysis:

Probability of M stopping in $2.p(n)\,.\,i$ time $= \frac{2}{3} . \frac{1}{3}^{i-1}$

M's expected running time $= \sum_{i \geq 1} 2.p(n)\,.\,i\,.\left(\frac{2}{3} . \frac{1}{3}^{i-1}\right) = O(p(n))$ ■

Polynomial Identity Testing

Polynomial Identity Testing

ZEROP **Problem:**

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients,

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.

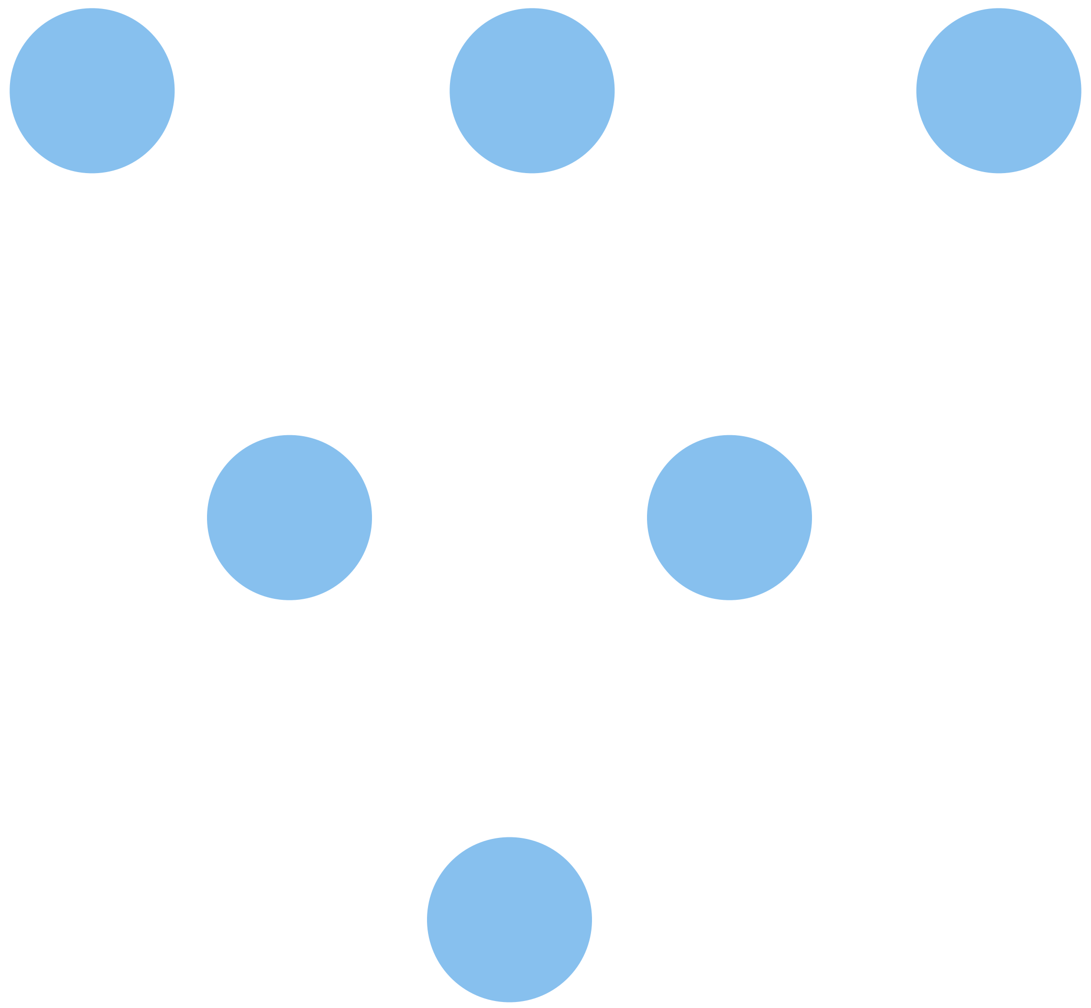
Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.



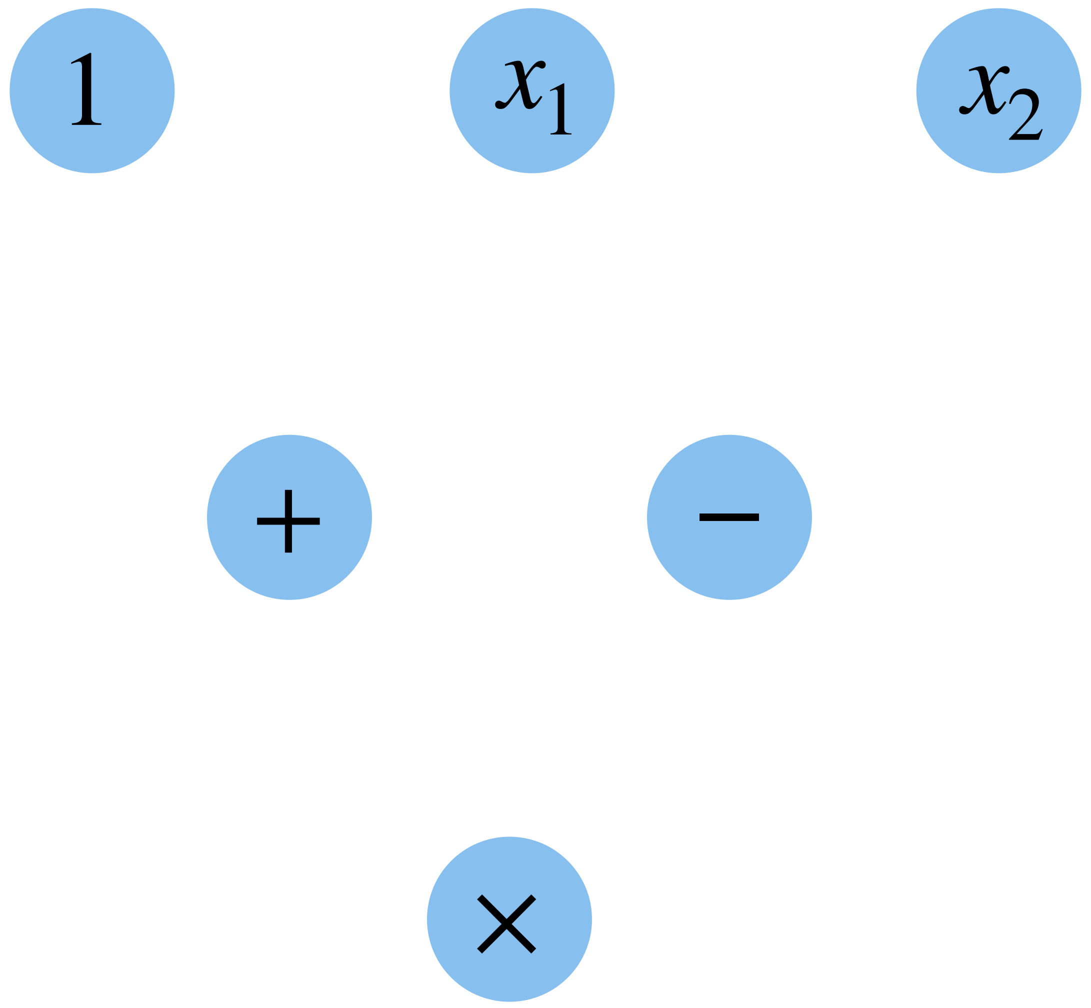
Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.



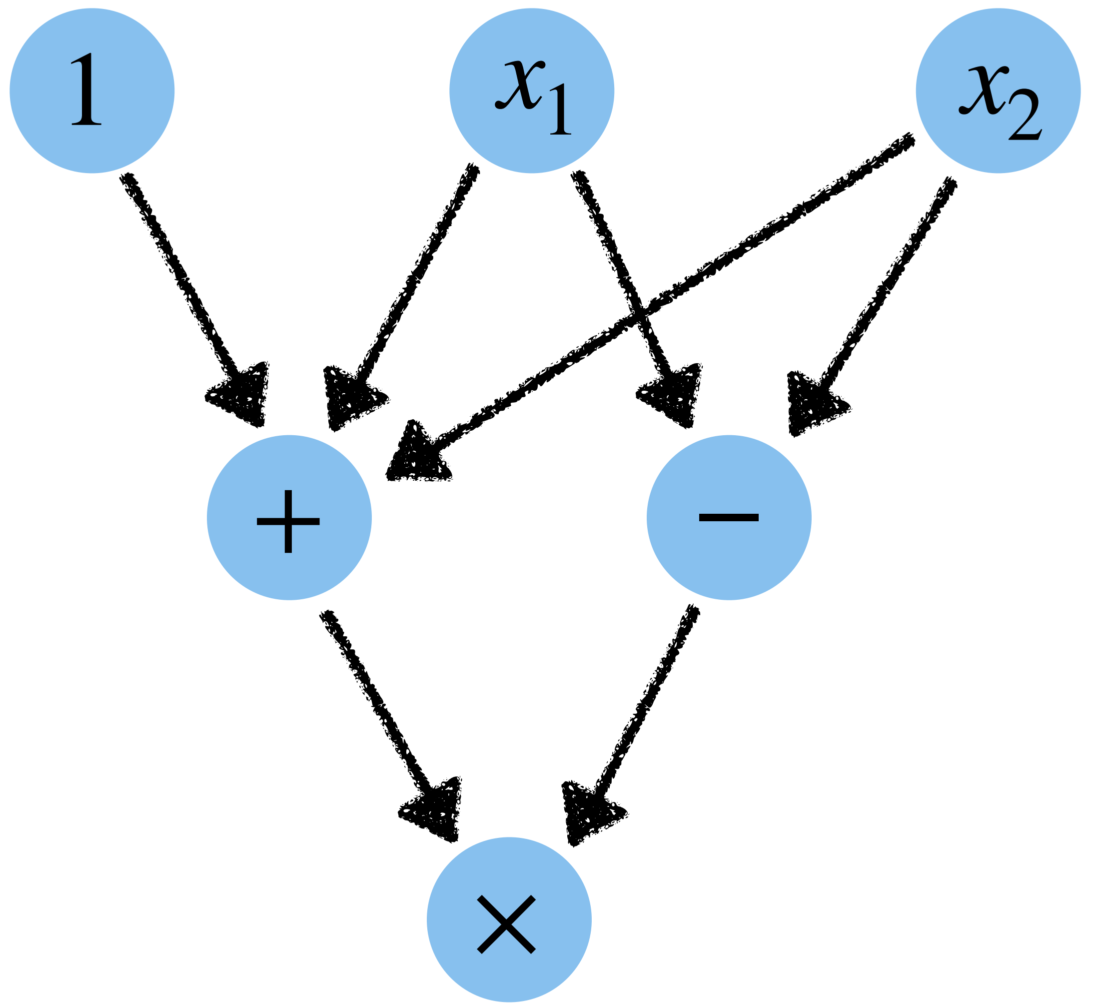
Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.



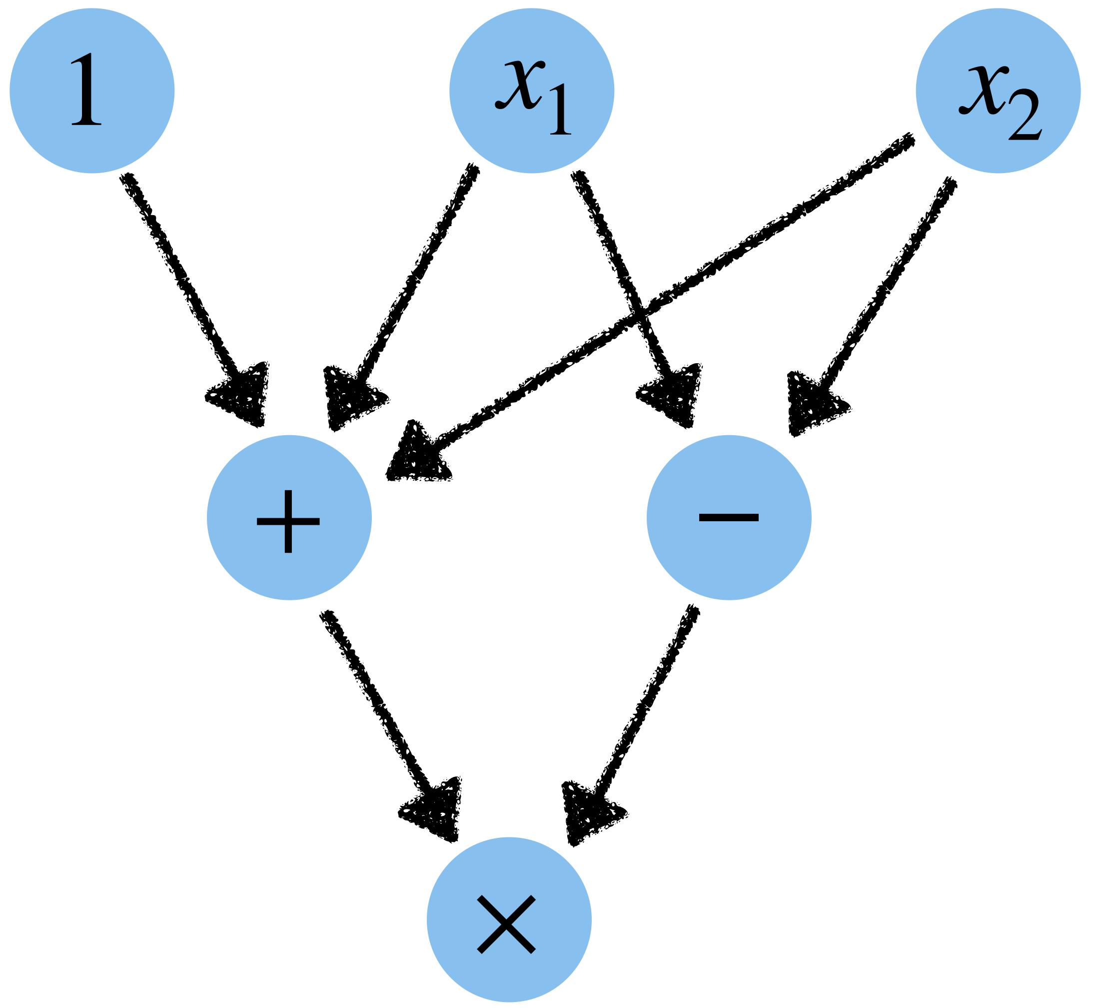
Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.



$$= \quad (x_1 + x_2 + 1)(x_1 - x_2)$$

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.

Simplest Approach for *ZEROP*:

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.

Simplest Approach for *ZEROP*:

- Given an algebraic circuit, compute the corresponding polynomial, say $p(x)$.

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.

Simplest Approach for *ZEROP*:

- Given an algebraic circuit, compute the corresponding polynomial, say $p(x)$.
- Output 1 iff all the coefficients of the corresponding polynomial are 0.

Polynomial Identity Testing

ZEROP **Problem:** Given a multivariate polynomial in an implicit form with integer coefficients, decide whether it is zero on all values.

Example: Is $x_1^2x_2 - x_1x_2 + x_2$ zero on all values of x_1 and x_2?

Fact: A polynomial is zero on all values if and only if each monomial has 0 as coefficient.

Implicit Form: Polynomial is given as an algebraic circuit that defines a polynomial from $\mathbb{Z}^n$ to $\mathbb{Z}$.

Simplest Approach for *ZEROP*:

- Given an algebraic circuit, compute the corresponding polynomial, say $p(x)$.
- Output 1 iff all the coefficients of the corresponding polynomial are 0.

Flaw: $O(n)$ size algebraic circuits can compute polynomials with 2^n many monomials, e.g., $\Pi_{i\in[n]}(1+x_i)$.

Polynomial Identity Testing

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d.

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d.

Let S be a finite set of integers.

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d.

Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\textbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\textbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis: If $C \in$ *ZEROP*, then A accepts with probability

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\textbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis: If $C \in$ *ZEROP*, then A accepts with probability

If $C \notin$ *ZEROP*, then A rejects with probability

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\textbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis: If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis: If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq 1 - \dfrac{2^m}{10.2^m}$

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\mathbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.

3) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis: If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq 1 - \dfrac{2^m}{10.2^m} = 9/10$

Polynomial Identity Testing

Schwartz-Zippel Lemma: Let $p(x_1, x_2, \ldots, x_m)$ be a non-zero polynomial with degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots, a_m$ are randomly chosen from S with replacement, then

$$\textbf{Pr}[p(a_1, a_2, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}$$

Algorithm A for *ZEROP* for a circuit C of size m that takes n inputs:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.
2) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) = y$.
3) If $y = 0$, then accept. Otherwise, reject.

Flaw: y can be as large as $(10.2^m)^{2^m}$.

Correctness Analysis: If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq 1 - \dfrac{2^m}{10.2^m} = 9/10$

Polynomial Identity Testing

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$.

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^{m})^{2^{m}}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof:

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \frac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}]$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

\# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}]$ $\geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}]$ $\geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$ | $|S|$

(from prime number theorem)

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

$|S| \leq \log y$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

$|S| \leq \log y \leq 5m2^m$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

$|S| \leq \log y \leq 5m2^m \leq \dfrac{2^{2m}}{4m}$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \frac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \frac{2^{2m}}{2m}$

(from prime number theorem)

$|S| \leq \log y \leq 5m2^m \leq \frac{2^{2m}}{4m}$

# of prime numbers in $[1,2^{2m}]$ that are not in S

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

$|S| \leq \log y \leq 5m2^m \leq \dfrac{2^{2m}}{4m}$

# of prime numbers in $[1,2^{2m}]$ that are not in $S \geq \dfrac{2^{2m}}{4m}$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}] \geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

$$|S| \leq \log y \leq 5m2^m \leq \frac{2^{2m}}{4m}$$

# of prime numbers in $[1,2^{2m}]$ that are not in S $\geq \dfrac{2^{2m}}{4m}$

Probability that k does not divide y $\geq \dfrac{2^{2m}/4m}{2^{2m}} = \dfrac{1}{4m}$

Polynomial Identity Testing

Lemma: Let $y \in [1,(10.2^m)^{2^m}]$ and a k is a number chosen randomly from $[1,2^{2m}]$. Then, with probability at least $\delta = \dfrac{1}{4m}$, k does not divide y.

Proof: Let $S = \{p_1, p_2, \ldots, p_l\}$ denote the set of all prime factors of y.

Sufficient to show that with probability $\geq \delta$, k will be a prime number number not in S.

# of prime numbers in $[1,2^{2m}]$ $\geq \dfrac{2^{2m}}{2m}$

(from prime number theorem)

$|S| \leq \log y \leq 5m2^m \leq \dfrac{2^{2m}}{4m}$

# of prime numbers in $[1,2^{2m}]$ that are not in S $\geq \dfrac{2^{2m}}{4m}$

Probability that k does not divide y $\geq \dfrac{2^{2m}/4m}{2^{2m}} = \dfrac{1}{4m}$ ■

Polynomial Identity Testing

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq (9/10)$

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq (9/10) \,.\, (1/4m)$

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq (9/10) \,.\, (1/4m)$

How can we improve the probability to a constant?

Polynomial Identity Testing

Algorithm B for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) If $y = 0$, then accept. Otherwise, reject.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A rejects with probability $\geq (9/10)\,.\,(1/4m)$

How can we improve the probability to a constant?

Repeat B $O(m)$ times and accept iff $y = 0$ on all iterations.

Polynomial Identity Testing

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1, 2, \ldots\ldots, 10.2^m]$.

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A accepts with probability

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n)$ mod $k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A accepts with probability $\leq \left(1 - \frac{9}{40m}\right)$

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A accepts with probability $\leq \left(1 - \frac{9}{40m}\right)^{40m/9}$

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A accepts with probability $\leq \left(1 - \dfrac{9}{40m}\right)^{40m/9} \leq 1/e$

Polynomial Identity Testing

Algorithm D for *ZEROP* for a circuit C of size m that takes n input:

1) Choose $x_1, x_2, \ldots, x_n$ randomly from $[1,2,\ldots\ldots,10.2^m]$.

2) Choose k randomly from $[1,2^{2m}]$.

3) Evaluate $C(x_1, x_2, \ldots\ldots, x_n) \bmod k = y$.

4) Repeat 2), 3) $40m/9$ times.

5) Accept iff $y = 0$ on all iterations.

Correctness Analysis:

If $C \in$ *ZEROP*, then A accepts with probability 1.

If $C \notin$ *ZEROP*, then A accepts with probability $\leq \left(1 - \dfrac{9}{40m}\right)^{40m/9} \leq 1/e$

■