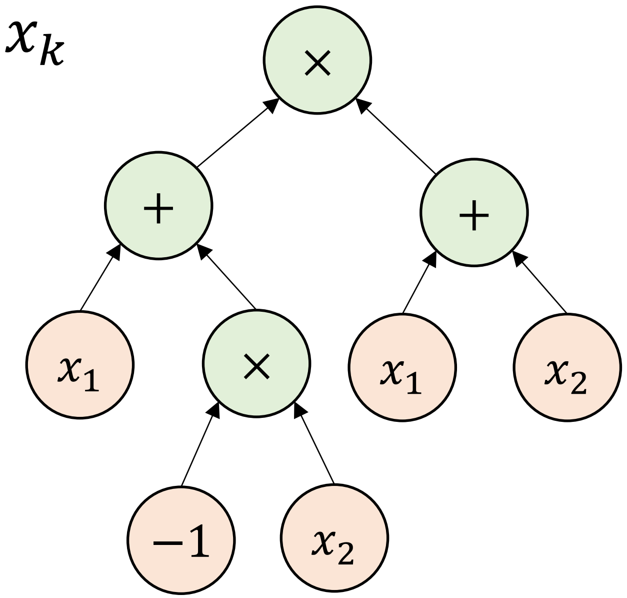
CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

Arithmetic formulas

- **Definition:** A k-variate arithmetic formula is a rooted binary tree
  - Each internal node is labeled with $+$ or $\times$
  - Each leaf is labeled with 0, 1, -1, or a variable among $x_1, \ldots, x_k$



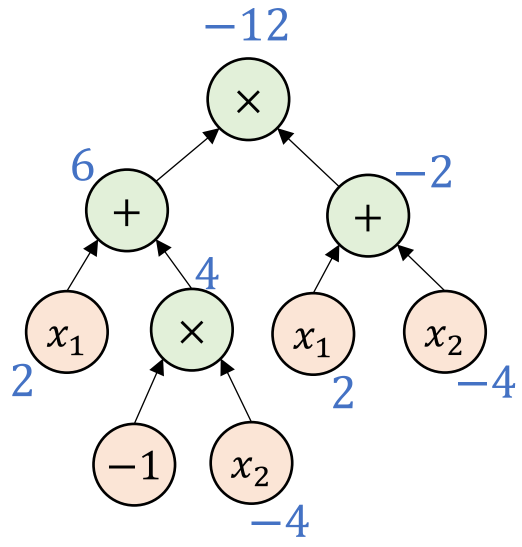
Polynomial identity testing

- **Problem:** Given an arithmetic formula F, determine whether $F \equiv 0$
- **As a language:** $\text{PIT} = \{\langle F \rangle : F \text{ is an arithmetic formula and } F \equiv 0\}$
- **Open Question:** Is $\text{PIT} \in \text{P}$?
- Next 10 slides: We will prove $\text{PIT} \in \text{BPP}$

Evaluating an arithmetic formula

- Let F be a k-variate arithmetic formula and let $\vec{x} \in \mathbb{Z}^k$

Lemma: Given $\langle F, \vec{x} \rangle$, one can compute $F(\vec{x}) \in \mathbb{Z}$ in polynomial time.



$x_1 = 2$

$x_2 = -4$

$F(x_1, x_2) = -12$ ✔

Possible concern: How big can these numbers get?

Bound on the magnitude of the output

- Let $M = \max(|x_1|, |x_2|, \dots, |x_k|, 2)$ and let d be the number of leaves
- **Claim:** $|F(\vec{x})| \leq M^d$. Proof by induction:
  - Base case: $d = 1$: trivial ✔
  - If $F(\vec{x}) = F_L(\vec{x}) \cdot F_R(\vec{x})$, then $|F(\vec{x})| = |F_L(\vec{x})| \cdot |F_R(\vec{x})| \leq M^{d_L} \cdot M^{d_R} = M^d$
  - If $F(\vec{x}) = F_L(\vec{x}) + F_R(\vec{x})$, then $|F(\vec{x})| \leq |F_L(\vec{x})| + |F_R(\vec{x})| \leq M^{d_L} + M^{d_R} \leq M^d$

Evaluating an arithmetic formula

- Let F be a k-variate arithmetic formula and let $\vec{x} \in \mathbb{Z}^k$

Lemma: Given $\langle F, \vec{x} \rangle$, one can compute $F(\vec{x}) \in \mathbb{Z}$ in polynomial time.

- **Proof sketch:** Evaluate the nodes one by one, starting at the leaves
- $M \leq 2^n$ and $d \leq n$, so each node outputs y such that $|y| \leq M^d \leq 2^{n^2}$
- In other words, y is an $O(n^2)$-bit integer
- There are $O(n)$ nodes, and we can do arithmetic in polynomial time

Note on standards of rigor

- Going forward, when we analyze specific algorithms, we will often assert that they run in polynomial time without a rigorous proof
  - In each case, one can rigorously prove the time bound by describing a TM implementation and reasoning about the motions of the heads…
  - But this is tedious
  - Note: We still prove correctness whenever it is nontrivial, just not efficiency
- You should follow this convention on exercise 14 and beyond

Polynomial identity testing

- We are given $\langle F \rangle$, where F is an arithmetic formula
- Goal: Figure out whether $F \equiv 0$
- If $F \equiv 0$, then $F(\vec{x}) = 0$ for all $\vec{x}$ 🥲
- Even if $F \not\equiv 0$, there still might be some $\vec{x}$ such that $F(\vec{x}) = 0$ ☹️
- How often can this occur?

Counting ro

How many roots can a nonzero degree-d two-variable polynomial have?

A: Up to d

B: Up to d^2

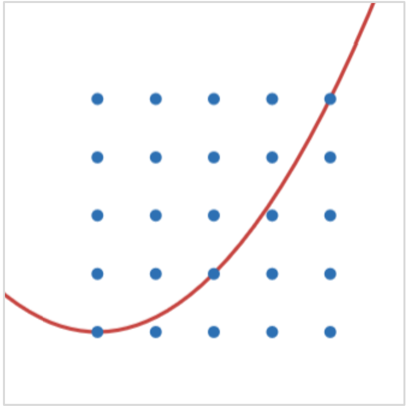
C: It might have infinitely many

D: Only finitely many, but there is no bound in terms of d

Respond at PollEv.com/whoza or text “whoza” to 22333

- **Fundamental Theorem of Algebra** ⇒ Every nonzero degree-d univariate polynomial has at most d real roots
- What about a multivariate polynomial?

Polynomial Identity Lemma



$F = y - x^2$

- Even if $F \not\equiv 0$, it might have infinitely many roots 😦
- Intuition: Roots are nevertheless "rare"
- Let $F : \mathbb{R}^k \to \mathbb{R}$ be a multivariate polynomial of degree at most d in each variable individually
- Let $S \subseteq \mathbb{R}$ and assume S is finite

Polynomial Identity Lemma: If $F \not\equiv 0$, then $\left|F^{-1}(0) \cap S^k\right| \leq dk \cdot |S|^{k-1}$

Polynomial Identity Lemma: If $F \not\equiv 0$, then $\left|F^{-1}(0) \cap S^k\right| \le dk \cdot |S|^{k-1}$

Proof when $k = 2$: Write $F(x,y) = \sum_{i=0}^{d} F_i(x) \cdot y^i$ and suppose $F_\ell \not\equiv 0$

$$|F^{-1}(0) \cap S^2| = \sum_{x \in S} |\{y \in S : F(x,y) = 0\}|$$

$$= \sum_{F_\ell(x)=0} |\{y \in S : F(x,y) = 0\}| + \sum_{F_\ell(x)\neq 0} |\{y \in S : F(x,y) = 0\}|$$

$$\le d \cdot |S| + |S| \cdot d$$

$$= 2d \cdot |S|$$

Fundamental Theorem of Algebra

Theorem: $\mathrm{PIT} \in \mathrm{BPP}$

Given F with k variables and d leaves:

1. Let $S = \{1, \dots, 3dk\}$
2. Pick $\vec{x} \in S^k$ uniformly at random
3. Compute $F(\vec{x}) \in \mathbb{Z}$
4. If $F(\vec{x}) = 0$, accept, otherwise reject

- Polynomial time ✔
- **Correctness proof**:
- Degree $\leq d$ (prove by induction)
- If $F \equiv 0$, then $\Pr[\text{accept}] = 1$
- If $F \not\equiv 0$, then by the Polynomial Identity Lemma, we have

$$\Pr[\text{accept}] = \Pr[F(\vec{x}) = 0] = \frac{\left|F^{-1}(0) \cap S^k\right|}{|S^k|} \leq \frac{dk \cdot |S|^{k-1}}{|S|^k} = \frac{dk}{3dk} = \frac{1}{3}$$

Polynomial identity testing: Recap

- It is an open question whether $\mathrm{PIT} \in \mathrm{P}$
- We proved $\mathrm{PIT} \in \mathrm{BPP}$
- Does that mean we should consider PIT “tractable?”

The complexity class BPP

- **Definition:** BPP is the set of languages $Y \subseteq \{0,1\}^*$ such that there exists a randomized polynomial-time Turing machine that decides Y with error probability $1/3$

Amplification lemma

- Suppose a language $Y \subseteq \{0,1\}^*$ can be decided by a time-T Turing machine M_0 with error probability $1/3$
- Let $k \in \mathbb{N}$ be any constant

Amplification Lemma: There exists a randomized time-T' Turing machine that decides Y with error probability 3^{-n^k}, where $T'(n) = O\left(T(n) \cdot n^k\right)$.

- As $n \to \infty$, the error probability goes to 0 extremely rapidly!

Proof of the amp

If M_0 uses $R(n)$ many random bits, then how many random bits does the new TM use?

A: $R(n) + n^k$

B: $R(n) \cdot n^k$

C: $R(n)^k$

D: Not enough information

Respond at PollEv.com/whoza or text "whoza" to 22333

- For simplicity, assume that
  - For $w \notin Y$, we merely assur

Given $w \in \{0,1\}^n$:

1) For $i = 1$ to n^k:
   a) Simulate M_0 on w using fresh random bits. If it rejects, reject.
2) Accept.

Time complexity: $O(T(n) \cdot n^k)$

- If $w \in Y$, then $\Pr[M \text{ accepts } w] = 1$

- If $w \notin Y$, then $\Pr[M \text{ accepts } w] \leq (1/3)^{n^k} = 1/3^{n^k}$

BPP as a model of tractability

- Because of the amplification lemma, languages in BPP should be considered "tractable"
- A mistake that occurs with probability $1/3^{100}$ can be safely ignored
- (Even if you use a deterministic algorithm, can you really be 100% certain that the computation was carried out correctly?)

Extended Church-Turing Thesis

Extended Church-Turing Thesis:

For every $Y \subseteq \{0, 1\}^*$, it is physically possible to build a device that decides Y in polynomial time if and only if $Y \in \mathrm{P}$.

- Is PIT a counterexample?