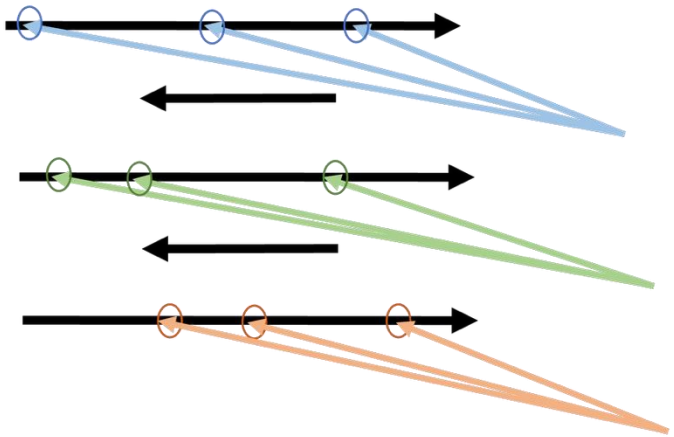
Linear Prover IOPs in Log Star Rounds

Noor Athamnah
Harvard University

Prover



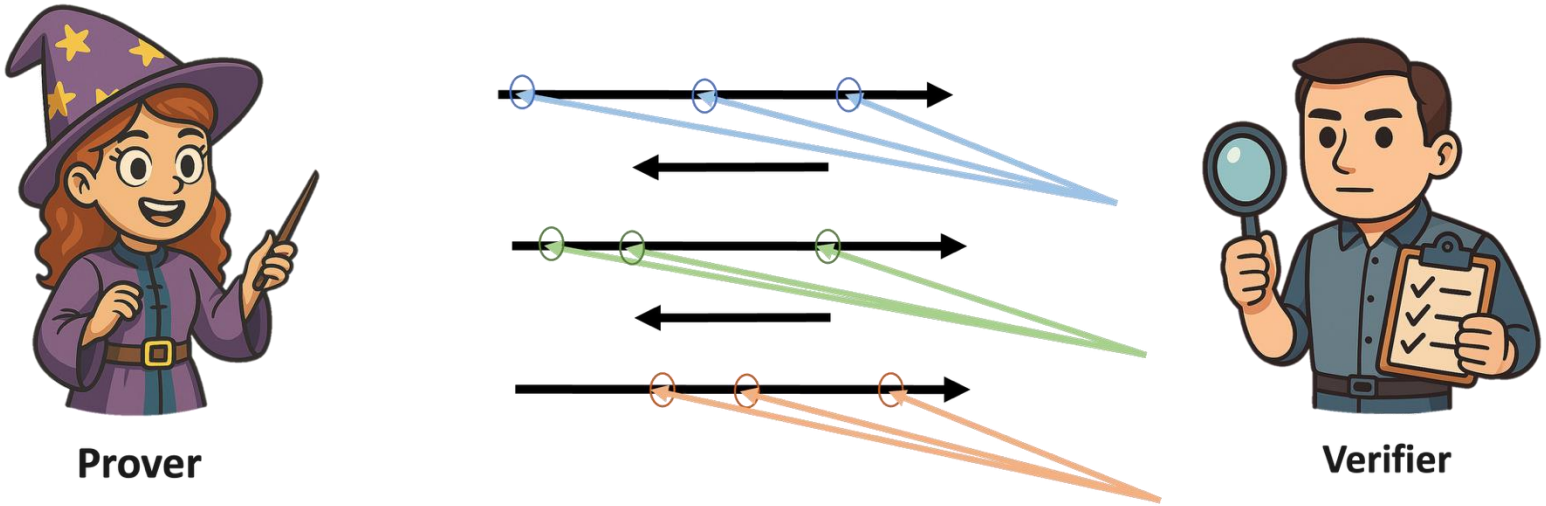
Verifier

Joint work with

Noga Ron-Zewi
Haifa University

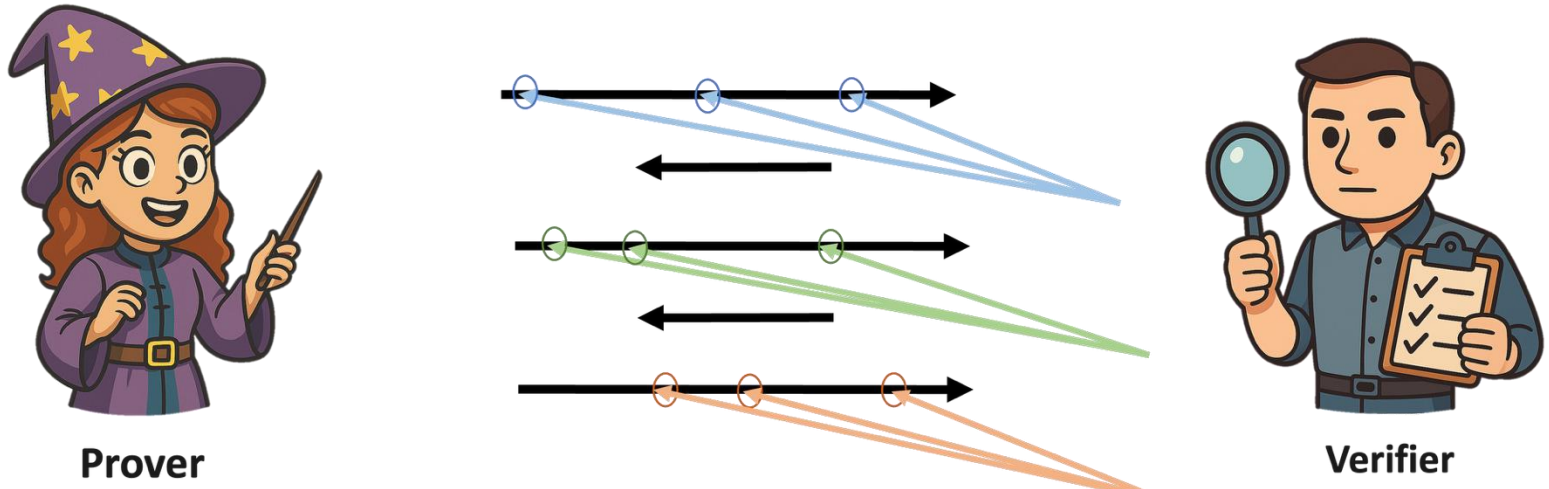
Ron Rothblum
Succinct

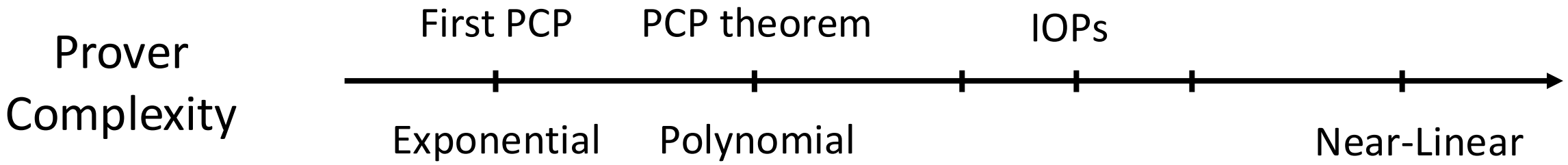
Interactive Oracle Proofs (IOP) [BCS16,RRR16]



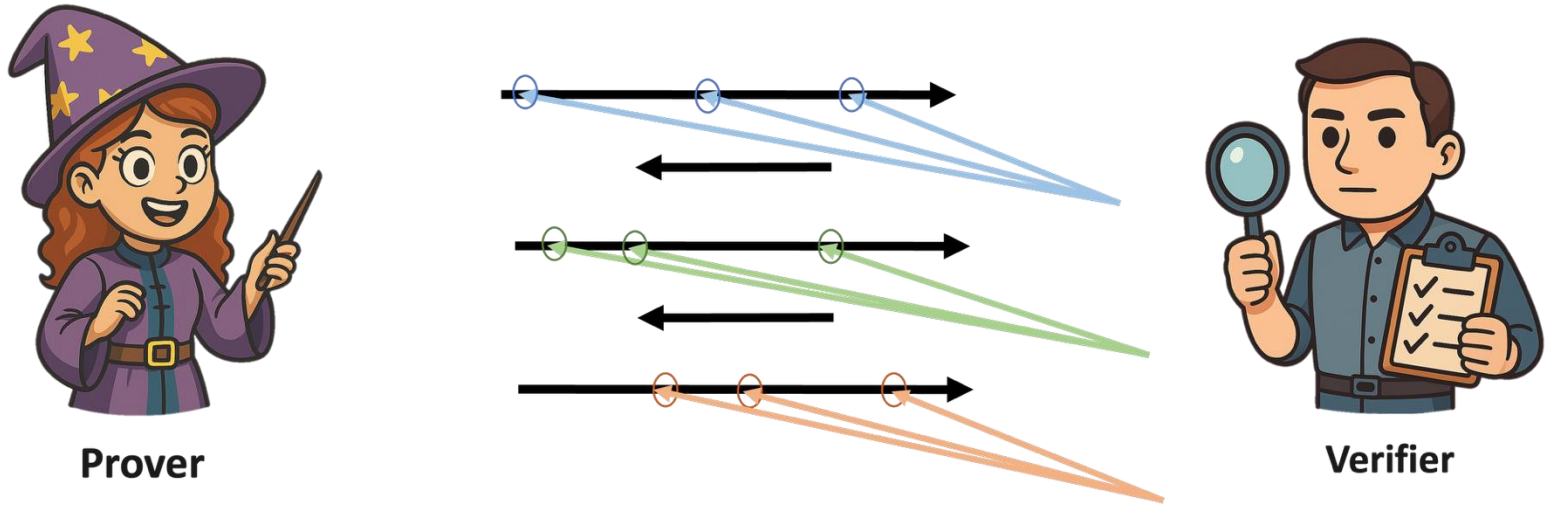
Prover
Efficiency

Interactive Oracle Proofs (IOP) [BCS16,RRR16]





Interactive Oracle Proofs (IOP) [BCS16,RRR16]



<u>Prover efficiency:</u>

Prover Overhead

Minimizing number of rounds

Talk Overview

Computational Model & Efficiency Measures

A Sumcheck Based Protocol

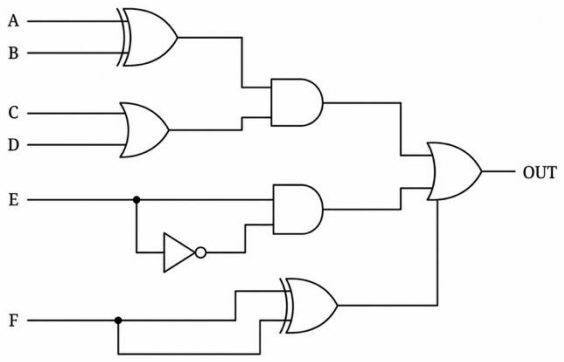
Quasi-Linear Prover

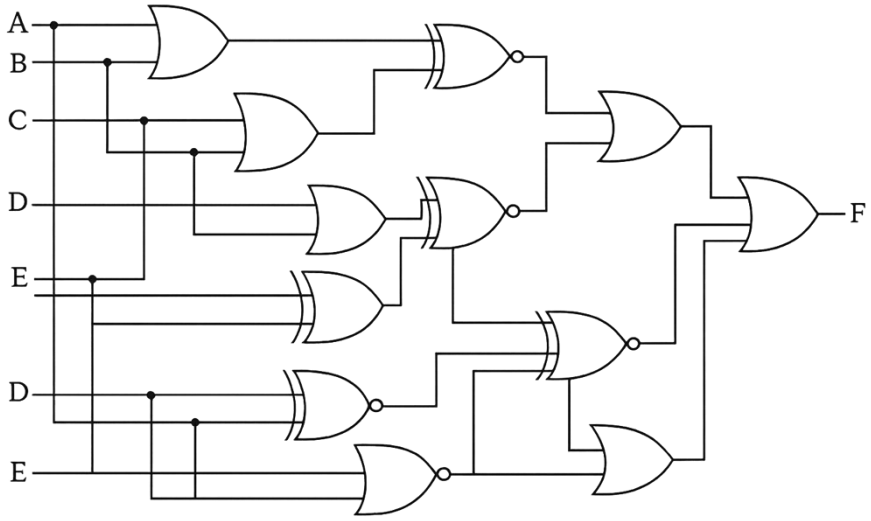
Logarithmic Number of Rounds

Reducing to a Linear Size Prover

Reducing the Number of Rounds

Computation Model





Boolean circuit

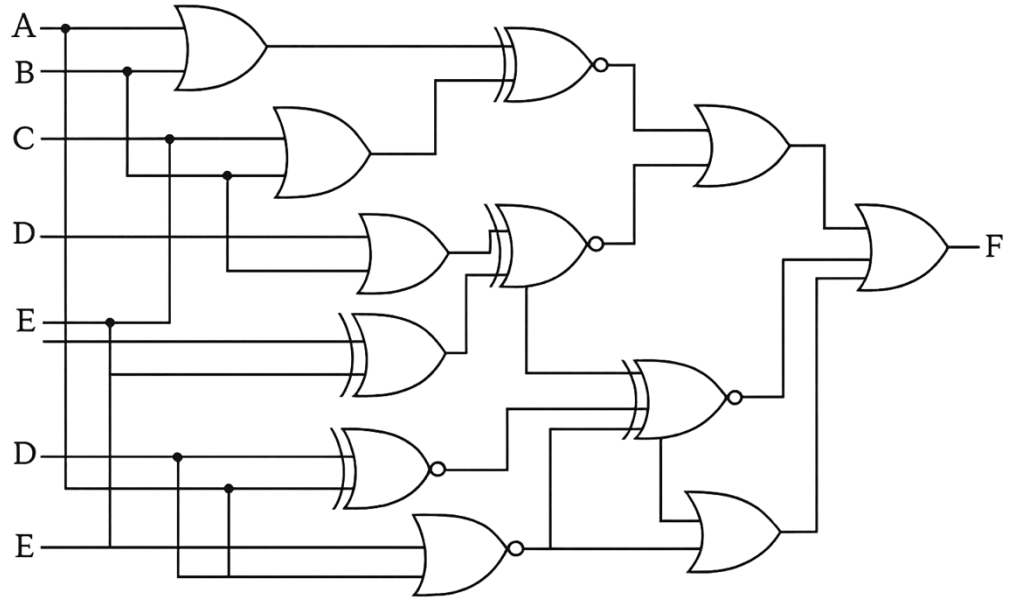
Arithmetic circuit
(Big field)

Original Computation
a Boolean circuit of size S

Prover
a Boolean circuit of size f(S)?

Computation Model

- A Boolean circuit of size S.
- Prover - a Boolean circuit of size f(S).

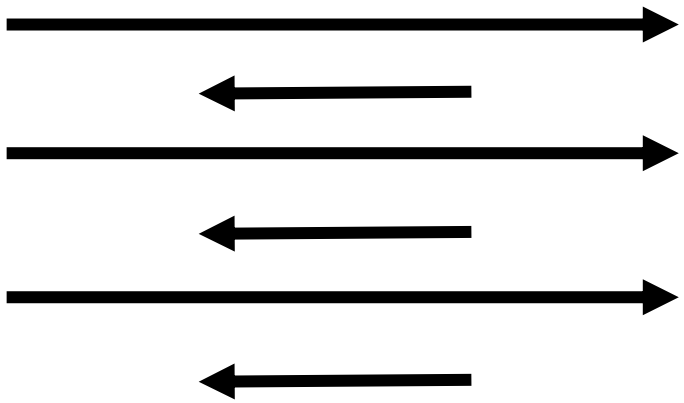


Today: The prover is linear in the size of the computation. [RR22]

Number of Rounds

Prover

$$x \overset{?}{\in} L$$



Verifier

☆ Parallel Computation

Interactive Oracle Proofs: Number of rounds vs. Proving overhead

S – size of Boolean circuit for computation

Number of Rounds	**Prover Size**	
Single Round (PCP)	Super-Linear	[BFLS91,AS92, ALMSS92,…]
Constant Number of Rounds	Super-Linear	[AHIV17,BCG+17, GLS+23,BFK+24]
$O(\log^*(S))$	Linear	New
$O(\log(S))$	Linear	[RR22, HR22]

Main Result

Theorem: For every "nice" Boolean circuit of size S, there exists an IOP:

- Prover is an $O(S)$ size Boolean circuit.
- Verifier runs in $\mathrm{polylog}(S)$ time.
- $O(\log^*(S))$ rounds.
- $O(\log^*(S))$ query complexity.
- Constant soundness error:
  - Can obtain $2^{-\lambda}$ soundness error with only $\mathrm{polylog}(\lambda)$ overhead via [HR22].

☆ Arbitrary circuits: linear verifier preprocessing

☆ $2\log^*(S) + O(1)$

$\log^*(n)$

$$\log(\log(\log(\ldots.\log(n)\ \ldots))) \leq 1$$

$$\log^*(4) =$$
$$\log^*(8) = \log^*(16) =$$
$$\log^*(2^{65535}) =$$

Number of atoms in the universe $\leq 10^{82} \leq 2^{273}$

Key Ingredient

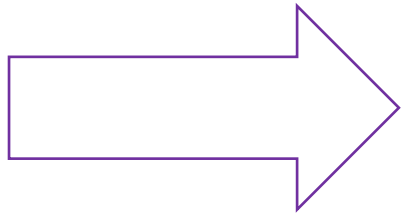
IOP for Inner product

Given $x, y \in \{0,1\}^n$,

Prove $\langle x, y \rangle := \sum_i x_i \cdot y_i = v$.

V runs in sublinear time.

V has oracle access to encodings of x, y.

IOP for R1CS  IOP for “nice” NP

Proof Outline

The Sumcheck Protocol [LFKN92]

Sumcheck:

$Q(z_1, \ldots, z_m)$ – a low-degree polynomial over $\mathbb{F}$.

Prove: $\sum_{z_i \in \{0,1\}} Q(z_1, \ldots, z_m) = v$.

V has oracle access to Q.

At the end, V queries one point of Q.

Inner Product Check:

$x, y \in \{0,1\}^n$

$\downarrow$

$f, g: \{0,1\}^m \to \{0,1\}$ $\longrightarrow$ $\hat{f}, \hat{g}$ multilinear extensions over $\mathbb{F}$ $\longrightarrow$ Sumcheck on $\hat{f} \cdot \hat{g}$

The Sumcheck Protocol [LFKN92]

Sumcheck:

$Q(z_1, \dots, z_m)$ – a low-degree polynomial over $\mathbb{F}$

Prove: $\sum_{z_i \in \{0,1\}}$

V has oracle acc

At the end, V qu

Inner Product Ch

Problem Solved?

$x, y \in \{0,1\}^n$

$f, g: \{0,1\}^m \to \{0,1\}$ → $\hat{f}, \hat{g}$ multilinear extensions over $\mathbb{F}$ → Sumcheck on $\hat{f} \cdot \hat{g}$

The Sumcheck Protocol [LFKN92]

Sumcheck:

$Q(z_1, \ldots, z_m)$ – a low-degree polynomial over $\mathbb{F}$.

Prove: $\sum_{z_i \in \{0,1\}} Q(z_1, \ldots, z_m) = v$.

V has oracle access to Q.

At the end, V queries one point of Q.

Inner Product Check:

$x, y \in \{0,1\}^n$

$f, g: \{0,1\}^m \to \{0,1\}$ → $\hat{f}, \hat{g}$ multilinear extensions over $\mathbb{F}$ → Sumcheck on $\hat{f} \cdot \hat{g}$

1. Prover complexity of sumcheck.

2. Cannot provide oracle access to a polynomial.

3. Logarithmic number of rounds.

1. Prover complexity of sumcheck.
2. Prover cannot provide oracle access to a polynomial.
3. Logarithmic number of rounds.

Sumcheck Protocol

Prover Complexity

Soundness

$Q(z_1, \dots, z_m) = \hat{f}(z_1, \dots, z_m) \cdot \hat{g}(z_1, \dots, z_m)$ over $\mathbb{F}$.

Want to prove: $\sum_{z_i \in \{0,1\}} Q(z_1, \dots, z_m) = v$.

Compute $Q_1 : \mathbb{F} \to \mathbb{F}$:

$$Q_1(x) = \sum_{z_i \in \{0,1\}} \hat{f}(x, z_2, \dots, z_m) \cdot \hat{g}(x, z_2, \dots, z_m)$$

Prover

$Q_1(x)$ →

← r_1

Verifier

Sumcheck Protocol

$Q(z_1, \ldots, z_m) = \hat{f}(z_1, \ldots, z_m) \cdot \hat{g}(z_1, \ldots, z_m)$ over $\mathbb{F}$.

Want to prove: $\sum_{z_i \in \{0,1\}} Q(z_1, \ldots, z_m) = v$.

Prover

$Q_1(x)$

r_1

$Q_2(x)$

r_2

$Q_m(x)$

r_m

Verifier

Query

$Q_m(r_m) = Q(r_1, r_2, \ldots, r_m) = \hat{f}(r_1, \ldots, r_m) \cdot \hat{g}(r_1, \ldots, r_m)$

Sumcheck Analysis

Prover complexity:

At round t: compute Q_t: $\mathbb{F} \to \mathbb{F}$:

$Q_t(x) = \sum_{z_i \in \{0,1\}} \hat{f}(r_1, \dots, r_{t-1}, x, z_{t+1}, \dots, z_m) \cdot \hat{g}(r_1, \dots, r_{t-1}, x, z_{t+1}, \dots, z_m).$

At round t: 2^{m-t} field operations

$\sum_{i=1}^{m} 2^{m-t}$ field operations

 Linear ☺

Sumcheck Analysis

Prover complexity:

At round t: compute $Q_t: \mathbb{F} \to \mathbb{F}$:

$Q_t(x) = \sum_{z_i \in \{0,1\}} \hat{f}(r_1, \dots, r_{t-1}, x, z_{t+1}, \dots, z_m) \cdot \hat{g}(r_1, \dots, r_{t-1}, x, z_{t+1}, \dots, z_m).$

At round t: 2^{m-t} field operations

$= 2^{m-t} \cdot \text{polylog}(|\mathbb{F}|)$

$2^m \cdot \sum_{i=1}^{m} 2^{-t} \text{ polylog}(|\mathbb{F}|)$

Sumcheck Analysis

Soundness:

$\frac{d}{|\mathbb{F}|}$ every round.

$\sum_{i=1}^{m} \frac{d}{|\mathbb{F}|} = m \cdot \frac{d}{|\mathbb{F}|}$

Sumcheck Analysis

Prover complexity vs. soundness:

Prover Complexity

$$2^m \sum_{i=1}^{m} 2^{-t} \cdot \text{polylog}(|\mathbb{F}_t|)$$

Soundness

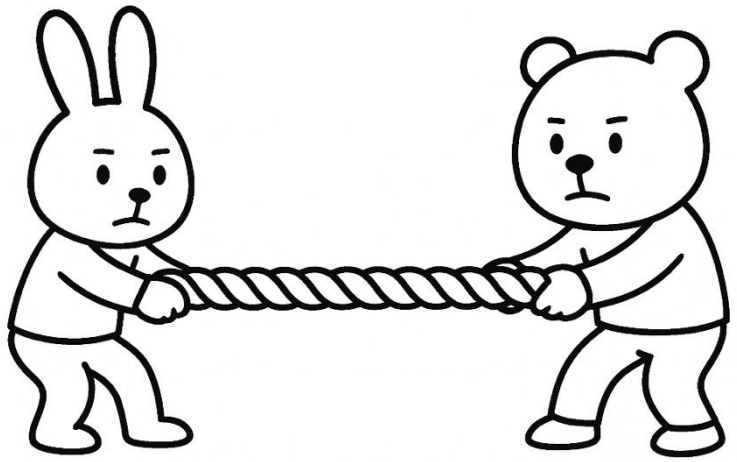
$$\sum_{i=1}^{m} d/|\mathbb{F}_t|$$

$|\mathbb{F}_t| \approx t^2$

$|\mathbb{F}_t| \approx 2^t$

$|\mathbb{F}_t| \approx 2^{2^t}$

Soundness Linear Prover



$|\mathbb{F}_i|$ super-constant

$|\mathbb{F}_i|$ constant

$\mathbb{F}_1 \subset \mathbb{F}_2 \subset \mathbb{F}_3 \subset \mathbb{F}_4 \subset \mathbb{F}_5$

1. Prover complexity of sumcheck.
2. Prover cannot provide oracle access to a polynomial.
3. Logarithmic number of rounds.

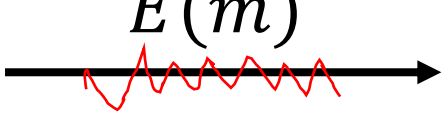
Error-Correcting Codes 101

message

m

Error-Correcting Codes 101

$E(m)$



A code is a function $E: \Sigma^k \to \Sigma^n$

If $m \neq m'$, then $E(m)$ *is far from* $E(m')$

$$\text{Dist}(E) = \min_{x,y \in C \; x \neq y} \frac{\{i | x_i \neq y_i\}}{n}$$

$$\text{Rate}(E) = \frac{k}{n}$$

Encoding complexity

Example:

Multilinear Extensions

Error-Correcting Codes 101

Proof Setting

m = correct computation

m' = wrong computation

A code is a function $E: \Sigma^k \to \Sigma^n$

If $m \neq m'$, then $E(m)$ *is far from* $E(m')$

$$\text{Dist}(E) = \min_{x,y \in C \; x \neq y} \frac{\{i | x_i \neq y_i\}}{n}$$

soundness

$$\text{Rate}(E) = \frac{k}{n}$$

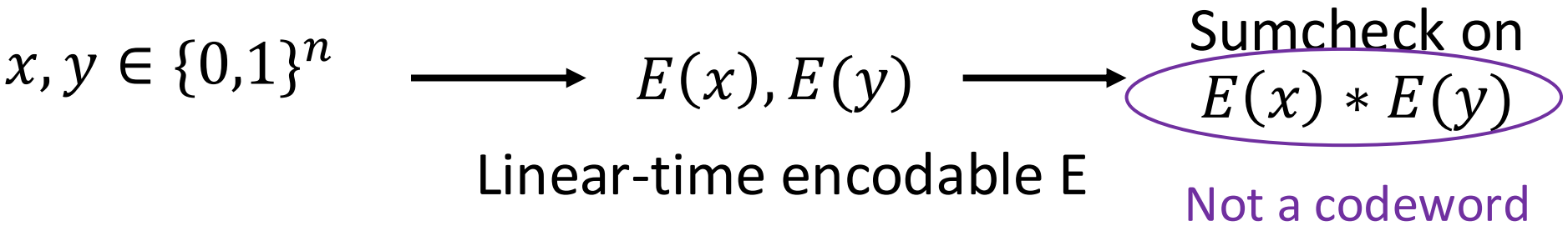
proof length

Encoding complexity

prover complexity

Linear-Time Encodable Codes

- We can do sumcheck on any (tensored) linear code:
  - P sends $E(x)$.
  - P and V run a sumcheck like protocol to prove the sum of x.
  - At the end V queries one point in $E(x)$.



- At the end, V queries $E(x), E(y)$ in a single point

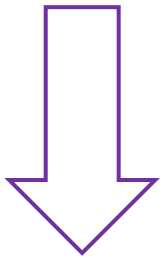
- **<u>Problem:</u>** No linear-time encodable code is multiplicative.

Proof Outline

Claim on the inner product of $\langle x, y \rangle$

$O(\log(n))$

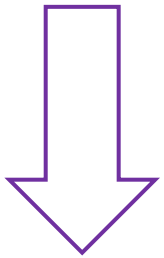
$O(\log^*(n))$ rounds



Sumcheck

Claim on a single point in the MLE of x and y separately

$O(1)$ rounds



Code-switching (Generalized sumcheck)

Claim on the encodings (under linear-time code) of x and y

Polynomial commitment scheme

Inner Product IOP

Theorem: Let $x, y \in \{0,1\}^n$, such that P has access to (x, y), and V has oracle access to $E(x), E(y)$ for a linear-time encodable code E.

Then there is an IOP for proving $\langle x, y \rangle = v$, such that:

- Prover is $O(n)$ size Boolean circuit.
- $O(\log^*(n))$ rounds.
- $O(\log^*(n))$ query complexity.

Warmup 1: Inner Product IOP with $O(\log(n))$ rounds

Theorem: Let $x, y \in \{0,1\}^n$, such that P has access to (x, y), and V has oracle access to $E(x), E(y)$ for a linear-time encodable code E.

Then there is an IOP for proving $\langle x, y \rangle = v$, such that:

- Prover is $O(n)$ size Boolean circuit.
- $O(\log(n))$ rounds.
- $O(\log(n))$ query complexity.

☆ Result of [RR22]

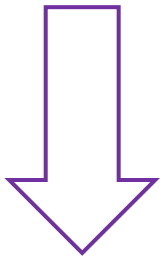
☆ Simpler protocol

Proof Outline

Claim on the inner product of $\langle x, y \rangle$

$O(\log(n))$

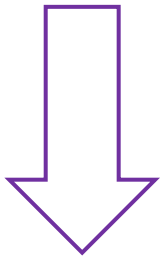
$O(\log^*(n))$ rounds



Sumcheck

Claim on a <u>single point</u> in the MLE of x and y <u>separately</u>

$O(1)$ rounds



Code-switching (Generalized sumcheck)

Claim on the encodings (under linear-time code) of x and y

Code-switching

Lemma: Let $f:\{0,1\}^m \to \{0,1\}$, such that P has access to f, and V has oracle access to $E(f)$.

Then there is an IOP for proving $\hat{f}(r) = v$, for a "nice" point r, such that:

- Prover is $O(2^m)$ size Boolean circuit.
- $O(1)$ rounds.
- $O(1)$ query complexity.

Computing a single point in the MLE

Extend the polynomial one variable at a time.

Reuse work done in previous rounds.

Make use of “nice” points.

Proving - similar to computing.

1. Prover complexity of sumcheck.
2. Prover cannot provide oracle access to a polynomial.
3. Logarithmic number of rounds.

Reducing to $\log^*$ rounds

$x \in \{0,1\}^n$ ⟶ $f:\{0,1\}^m \to \{0,1\}$ $\quad m = \log(n)$

Reducing to $\log^*$ rounds

$x \in \{0,1\}^n$ ⟶ $f: H_1 \times H_2 \times \cdots \times H_\ell \to \{0,1\}$

ℓ constant ⟶ $H_i = n^{\left(\frac{1}{\ell}\right)}$

Can no longer have multilinear extension ⟶ Can have low-degree extension

FFT

~~Prover of size $n \cdot n^{\left(\frac{1}{\ell}\right)}$~~ ⟶ Prover of size $n \cdot \text{polylog}\,(n)$

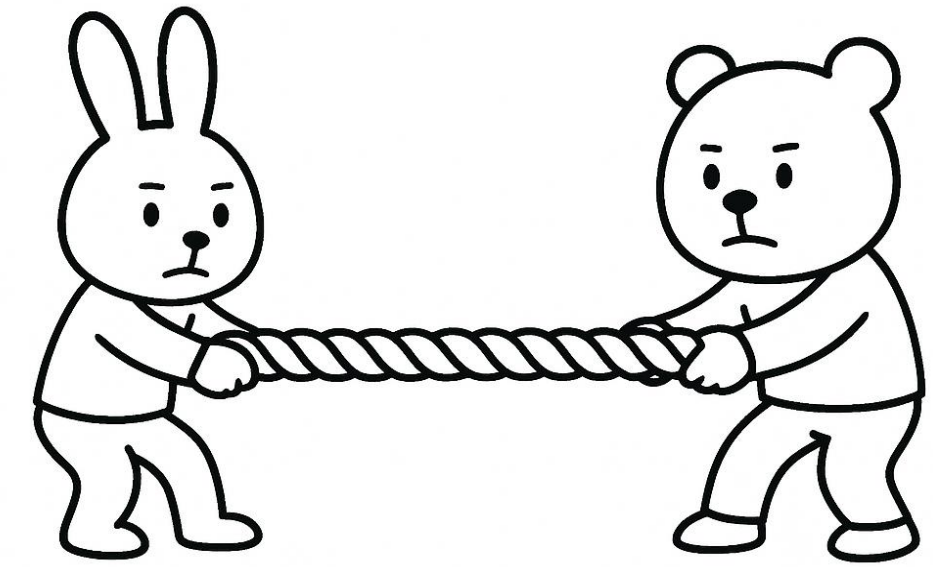
Reducing to $\log^*$ rounds

$$f: H_1 \times H_2 \times \cdots \times H_\ell \to \{0,1\}$$

$$|H_i| \approx 2^{|H_{i-1}|}$$

Fewer Rounds

Linear Prover



$|H_i|$

super-constant

$|H_i|$

constant

Summary & Open Questions

Main Result:

IOP for “regular” Boolean circuits of size S with:

- Prover of size $O(S)$.
- $O(\log^*(S))$ rounds.

Open Questions:

Can we get the same result for **all** circuits?

Constant-round IOPs with a linear prover.

Even for arithmetic circuits over large fields?