Lecture A.4

Doubly-Efficient IPs

Summer Graduate School on
Foundations and Frontiers of Probabilistic Proofs
2021.07.29

Inefficiency of Honest Provers

Our focus so far: achieve a polynomial-time verifier.

What about the honest prover?

Say we are given an n-variable boolean formula ϕ.

- in the sumcheck protocol (for #SAT): $\text{time}(P) = \Omega(2^n |\phi|)$.
- in Shamir's protocol (for TQBF): $\text{time}(P) = \Omega(2^n |\phi|)$.

[in fact Shamir's original protocol, without Shen's simplification, reduced the QBF to a "simple" QBF, squaring #vars so $\text{time}(P) = O(2^{n^2} |\phi|)$]

Are these times useful for computations of interest?

Let M be a machine running in time T and space S, and define

$$L_M := \{ x \mid M(x) = 1 \}.$$

The reduction from L_M to TQBF maps x to a boolean formula ϕ with size $|\phi| = \text{poly}(\log T, S)$ and $n \geq (\log T) \cdot S$ variables.

Even if $T, S = \text{poly}(|x|)$, the honest prover runs in time $\Omega(2^n) = \Omega(T^S) = |x|^{\omega(1)}$.

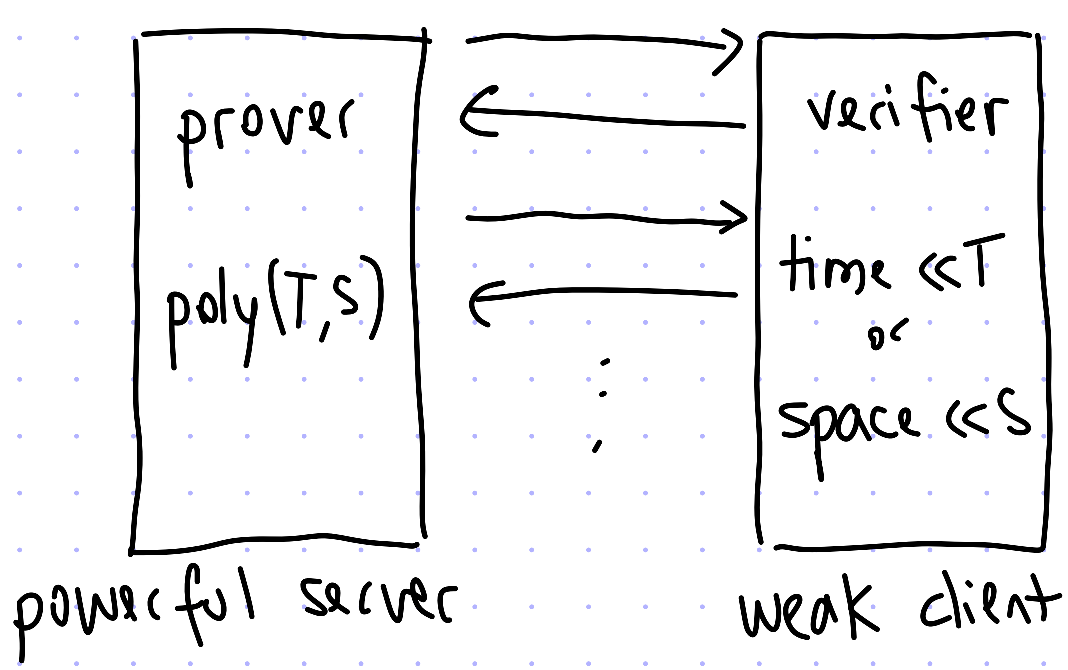
Doubly-Efficient Interactive Proofs

New goal: additionally restrict honest prover to run in polynomial time.
We call this a doubly-efficient interactive proof (deIP).

<u>claim:</u> deIP $\subseteq$ BPP

<u>proof:</u> The probabilistic algorithm simulates the interaction between the honest prover and the honest verifier. $\square$

To make deIP non-trivial, we require the verifier to work less than deciding the language alone (e.g. less space, less time, ...).
This setting can be viewed as delegation of computation:



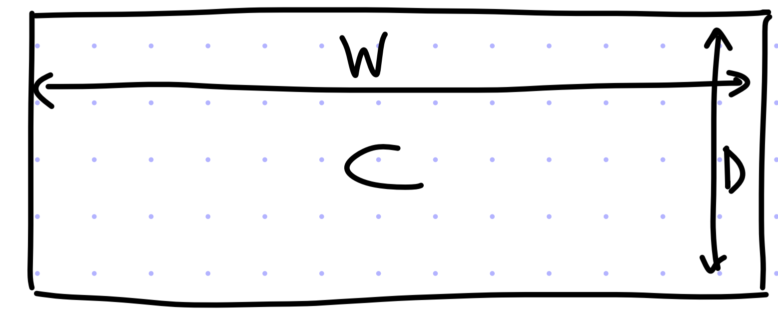
Q: what languages have doubly-efficient interactive proofs?

Delegation for Bounded-Depth Circuits

<u>theorem</u>: Suppose that L is decidable by a circuit family of width W and depth D that is $O(\log(W\cdot D))$-uniform.
Then L has a public-coin IP such that:

- prover time is $\mathrm{poly}(W,D)$
- verifier time is $(n+D)\cdot \mathrm{poly}(\log W)$ [& space is $O(\log(W\cdot D))$]
- communication (and # rounds) is $D\cdot \mathrm{poly}(\log W)$

a circuit family $\{C_n\}_{n\in\mathbb{N}}$ is a-space uniform if $\exists$ machine M s.t. (i) $M(1^n) = C_n$ (ii) $M(1^n)$ runs in space $O(a(n))$



size is $W\cdot D$

Ex: Can delegate (log-space uniform) NC (where $W = \mathrm{poly}(n)$, $D = \mathrm{polylog}(n)$). It contains arithmetic, and linear algebra.

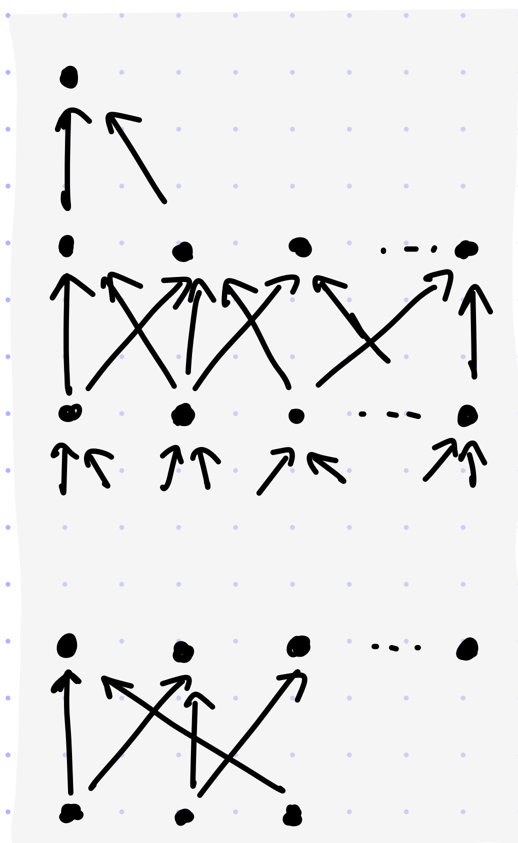
The proof of the theorem is quite technical.
We will see one piece: the "bare bones" protocol, which is same as above except that the verifier has oracle access to information about the circuit's topology (it will make $O(D)$ calls to this oracle) which saves us from discussing uniformity.

Main tools for bare-bones protocol: ← this has been implemented and it is <u>very</u> efficient!

more arithmetization, more sumcheck, some new ideas.

Layered Arithmetic Circuits

A layered arithmetic circuit $C: \mathbb{F}^n \to \mathbb{F}$ of width W and depth D (with $n \le W$) is an arithmetic circuit with fan-in 2 arranged in $D+1$ layers:



layer 0	1 output	$V_0 \in \mathbb{F}$	
layer 1	W internal values	$V_1 : [W] \to \mathbb{F}$	$add_1 : [W]^3 \to \{0,1\}$, $mul_1 : [W]^3 \to \{0,1\}$
layer 2	W internal values	$V_2 : [W] \to \mathbb{F}$	$add_2 : [W]^3 \to \{0,1\}$, $mul_2 : [W]^3 \to \{0,1\}$
⋮		⋮	
layer D-1	W internal values	$V_{D-1} : [W] \to \mathbb{F}$	$add_D : [W] \times [n]^2 \to \{0,1\}$, $mul_D : [W] \times [n]^2 \to \{0,1\}$
layer D	n inputs	$V_D : [n] \to \mathbb{F}$	

The wiring predicates $[(add_i, mul_i)]_{i=1,\dots,D}$ describe the circuit C: add_i/mul_i at (a,b,c) is 1 if a-th value in layer $i-1$ is the addition/multiplication of b-th & c-th values in layer i.

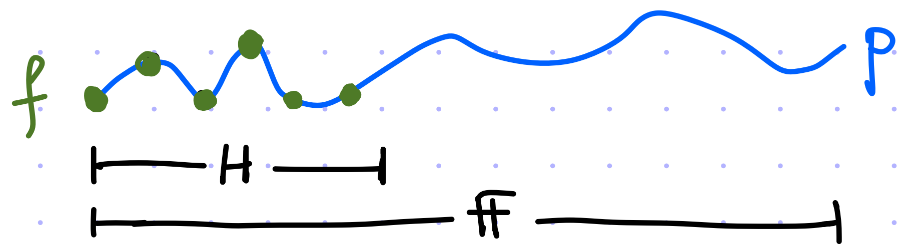
For notational simplicity, we assume C has 1 type of gate: $g: \mathbb{F}^2 \to \mathbb{F}$.

We can take $(wp_1, \dots, wp_D)$ to be the wiring predicates for g. [Extending to multiple gate types is straightforward.]

Low-Degree Extension (Univariate)

Let $H \subseteq \mathbb{F}$ be a domain, and $f: H \to \mathbb{F}$ a function.

A polynomial $p \in \mathbb{F}[X]$ is an extension of f if $p|_H \equiv f$.

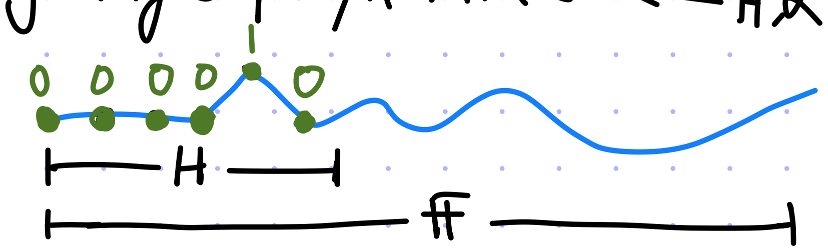


It is a low-degree extension if p has "low degree" [the specific condition varies].
The higher the allowed degree, the more low-degree extensions a function has.

The extension of degree $<|H|$ is unique and is constructed via INTERPOLATION:

① consider the univariate Lagrange polynomials $\{L_{H,\alpha}(X)\}_{\alpha \in H}$ where

$$L_{H,\alpha}(X) := \prod_{\beta \in H \setminus \{\alpha\}} \frac{X-\beta}{\alpha-\beta}$$



② take the linear combination according to the function

$$p(X) := \sum_{\alpha \in H} f(\alpha) \cdot L_{H,\alpha}(X) = \sum_{\alpha \in H} f(\alpha) \cdot \left(\prod_{\beta \in H \setminus \{\alpha\}} \frac{X-\beta}{\alpha-\beta} \right).$$

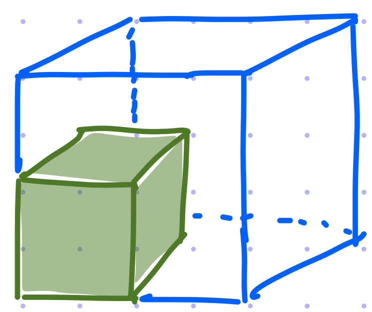
can evaluate p at any point in poly($|H|$) ops

Low-Degree Extension (Multivariate)

The multivariate case builds on the univariate case.

We consider functions of the form $f: H^n \to \mathbb{F}$.

We say that $p \in \mathbb{F}[X_1, \ldots, X_n]$ extends $f: H^n \to \mathbb{F}$ if $p|_{H^n} \equiv f$.



The extension of individual degree $< |H|$ is unique and is by INTERPOLATION:

① consider the multivariate Lagrange polynomials $\{L_{H^n, \alpha_1, \ldots, \alpha_n}(X_1, \ldots, X_n)\}_{\alpha_1, \ldots, \alpha_n \in H}$

where $L_{H^n, \alpha_1, \ldots, \alpha_n}(X_1, \ldots, X_n) := \prod_{i \in [n]} L_{H, \alpha_i}(X_i) = \prod_{i \in [n]} \prod_{\beta \in H \setminus \{\alpha_i\}} \frac{X_i - \beta}{\alpha_i - \beta}$.

← can evaluate at any point in $\mathrm{poly}(|H|, n)$ ops

② take the linear combination according to the function

$$p(X_1, \ldots, X_n) := \sum_{\alpha_1, \ldots, \alpha_n \in H} f(\alpha_1, \ldots, \alpha_n) \cdot L_{H^n, \alpha_1, \ldots, \alpha_n}(X_1, \ldots, X_n).$$

← can evaluate p at any point in $\mathrm{poly}(|H|^n)$ ops

Arithmetize Each Layer

Fix a subset $H \subseteq \mathbb{F}$ of size $\leq O(\log W)$ and set $m := \frac{\log W}{\log|H|}$ and $m_{in} := \frac{\log n}{\log|H|}$.
This induces bijections $[W] \leftrightarrow H^m$ and $[n] \leftrightarrow H^{m_{in}}$.

Step 1: rewrite computation as summations

Let $z \in \mathbb{F}^n$ be an input to the circuit $C: \mathbb{F}^n \to \mathbb{F}$.

- the input layer $V_D: H^{m_{in}} \to \mathbb{F}$ is defined as $V_D(a) := z_a$
- for $i = D-1, \ldots, 1, 0$: $V_i: H^m \to \mathbb{F}$ is defined as $V_i(a) := \sum_{b,c \in H^m} wp_{i+1}(a,b,c) \cdot g(V_{i+1}(b), V_{i+1}(c))$

we assumed for simplicity that C has 1 type of gate rather than add & mul gates

Step 2: low-degree extend each layer

- the extension of the input layer is $\hat{V}_D: \mathbb{F}^{m_{in}} \to \mathbb{F}$ where

$$\hat{V}_D(x) := \sum_{a \in H^{m_{in}}} z_a \cdot L_{H^{m_{in}},a}(x).$$

- the extension of the i-th layer is $\hat{V}_i: \mathbb{F}^m \to \mathbb{F}$ where

$$\hat{V}_i(x) := \sum_{a \in H^m} \left(\sum_{b,c \in H^m} \hat{wp}_{i+1}(a,b,c) \cdot g(\hat{V}_{i+1}(b), \hat{V}_{i+1}(c)) \right) \cdot L_{H^m,a}(x).$$

can consider extension instead of function as summation is over H

analogous to Shen's relinearization

Step 3: replace $L_{H^m,a}(X)$ with $I_{H^m}(X,a)$ where $I_{H^m}(X,Y) := \prod_{i=1}^{m} \sum_{\alpha \in H} L_{H,\alpha}(X_i) L_{H,\alpha}(Y_i)$
to ensure that a has low degree in addend

Rewrite Computation as Iterated Sumchecks

The statement "$C(z) = y$" is rewritten as "$\hat{V}_0(0) = y$".

Equivalently, "$\sum_{a,b,c \in H^m} \hat{wp}_1(a,b,c) \cdot g(\hat{V}_1(b), \hat{V}_1(c)) \cdot I_{H^m}(0,a) = y$".

So we can do a sumcheck on variables for a,b,c. This involves:

- $3m$ rounds [we are summing over the hypercube H^{3m}]
- soundness error $O(m \cdot \frac{|H|}{|\mathbb{F}|})$ [individual degrees are $O(|H|)$ so we pay $O(\frac{|H|}{|\mathbb{F}|})$ per round]
- $\mathrm{poly}(|H|^m) = \mathrm{poly}(W)$ operation for the honest prover — THIS IS EFFICIENT
- $\mathrm{poly}(m, |H|) = \mathrm{poly}(\log W)$ operations for the verifier, given
  - 1 query to $\hat{wp}_1 : \mathbb{F}^{3m} \to \mathbb{F}$ ← assume that verifier can evaluate on its own
  - 2 queries to $\hat{V}_1 : \mathbb{F}^m \to \mathbb{F}$

The prover sends the answers and we recurse on two claims: "$\hat{V}_1(s) = \gamma$" and "$\hat{V}_1(t) = \delta$".

Indeed, each claim is itself a sum:

"$\sum_{a,b,c \in H^m} \hat{wp}_2(a,b,c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot I_{H^m}(s,a) = \gamma$"

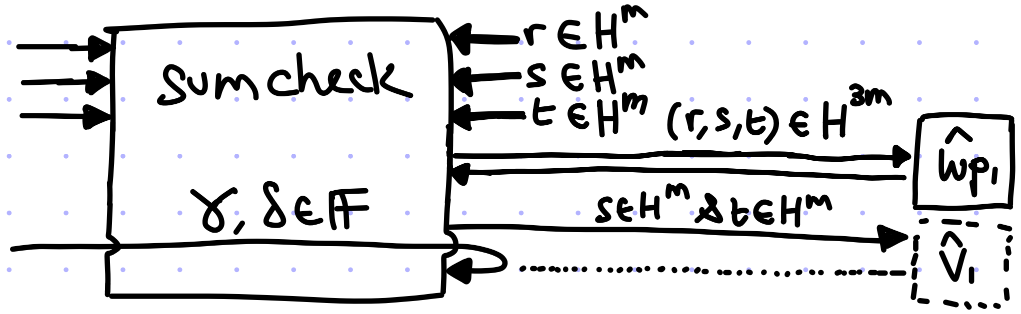
"$\sum_{a,b,c \in H^m} \hat{wp}_2(a,b,c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot I_{H^m}(t,a) = \delta$"

Problem: the number of claims doubles at each layer

Avoiding Claim Blowup

1 claim about layer 0

$$\text{"}\sum_{a,b,c\in H^m} \hat{wp}_1(a,b,c)\cdot g(\hat{V}_1(b),\hat{V}_1(c))\cdot I_{H^m}(0,a) = y\text{"}.$$



2 claims about layer 1

$$\text{"}\sum_{a,b,c\in H^m} \hat{wp}_2(a,b,c)\cdot g(\hat{V}_2(b),\hat{V}_2(c))\cdot I_{H^m}(s,a) = \gamma\text{"}$$

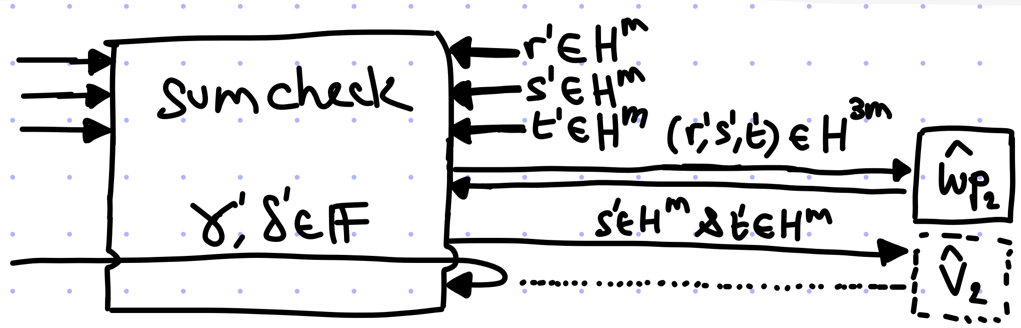
$$\text{"}\sum_{a,b,c\in H^m} \hat{wp}_2(a,b,c)\cdot g(\hat{V}_2(b),\hat{V}_2(c))\cdot I_{H^m}(t,a) = \delta\text{"}$$

$$\xleftarrow{\alpha,\beta\in_R \mathbb{F}}$$

1 claim about layer 1

$$\text{"}\sum_{a,b,c\in H^m} \hat{wp}_2(a,b,c)\cdot g(\hat{V}_2(b),\hat{V}_2(c))\cdot\left[\alpha\cdot I_{H^m}(s,a)+\beta\cdot I_{H^m}(t,a)\right] = \alpha\cdot\gamma+\beta\cdot\delta\text{"}$$

random linear combination of the 2 claims



2 claims about layer 2

$$\text{"}\sum_{a,b,c\in H^m} \hat{wp}_3(a,b,c)\cdot g(\hat{V}_3(b),\hat{V}_3(c))\cdot I_{H^m}(s',a) = \gamma'\text{"}$$

$$\text{"}\sum_{a,b,c\in H^m} \hat{wp}_3(a,b,c)\cdot g(\hat{V}_3(b),\hat{V}_3(c))\cdot I_{H^m}(t',a) = \delta'\text{"}$$

... and so on.

Protocol Summary

public coin

set $H=\{0,1\}$

- number of rounds is

$$D\cdot\left([\text{SC on } 3m \text{ vars}]+1\right)$$
$$=O(Dm)=O\left(D\cdot\frac{\log W}{\log|H|}\right)=O(D\cdot\log W)$$

- communication complexity (in elts) is

$$D\cdot\left([\text{SC on } 3m \text{ vars of degree } O(|H|)]+2\right)$$
$$=O(D\cdot m\cdot|H|)=O\left(D\cdot\frac{\log W}{\log|H|}\cdot|H|\right)=O(D\cdot\log W)$$

- soundness error is

$$D\cdot\left([\text{SC on } 3m \text{ vars of degree } O(|H|)]+\frac{1}{|\mathbb{F}|}\right)$$
$$=O\left(D\cdot\frac{m\cdot|H|}{|\mathbb{F}|}\right)=O\left(D\cdot\frac{\log W\cdot|H|}{\log|H|\cdot|\mathbb{F}|}\right)=O\left(D\cdot\frac{\log W}{|\mathbb{F}|}\right)$$

- prover time (in field ops) is

$$D\cdot\left([\text{SC on } 3m \text{ vars of degree } O(|H|)]+\text{small}\right)$$
$$=D\cdot\text{poly}(|H|^m)=D\cdot\text{poly}(W)$$

- verifier time (in field ops) is

$$D\cdot\left([\text{SC on } 3m \text{ vars of degree } O(|H|)]\right)$$
$$=D\cdot\text{poly}(m,|H|)=D\cdot\text{poly}\left(\frac{\log W}{\log|H|},|H|\right)=D\cdot\text{poly}(\log W)$$

[+ poly(n) to eval LDE of input
+ eval each of $\{\hat{wp}_i\}_{i\in[D]}$ at 1 point]

1 claim about $\hat{V}_0$

1 query to $\hat{wp}_1$ — sumcheck

2 claims about $\hat{V}_1$

combination

1 claim about $\hat{V}_1$

1 query to $\hat{wp}_2$ — sumcheck

2 claims about $\hat{V}_2$

combination

1 claim about $\hat{V}_2$

⋮

1 claim about $\hat{V}_D$

[can compute on its own]

Did not discuss:

in last sumcheck one variable group is in $H^{m_{in}}$ [not H^m]