

Foundations of Probabilistic Proofs

A course by **Alessandro Chiesa**

Lecture 04

Doubly-Efficient IPs

Inefficiency of Honest Provers

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

Today: what about the honest prover?

Let ϕ be an n -variable boolean formula.

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

Are these times useful for computations of interest?

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

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).

claim: $\text{deIP} \subseteq \text{BPP}$

proof: The probabilistic algorithm $A(x)$ simulates the interaction between the honest prover $P(x)$ and the honest verifier $V(x)$. ■

To make a deIP **non-trivial** (rule out simply running the BPP decider) we require the verifier to **work less than deciding the language alone**.

e.g. less time, less space, ...

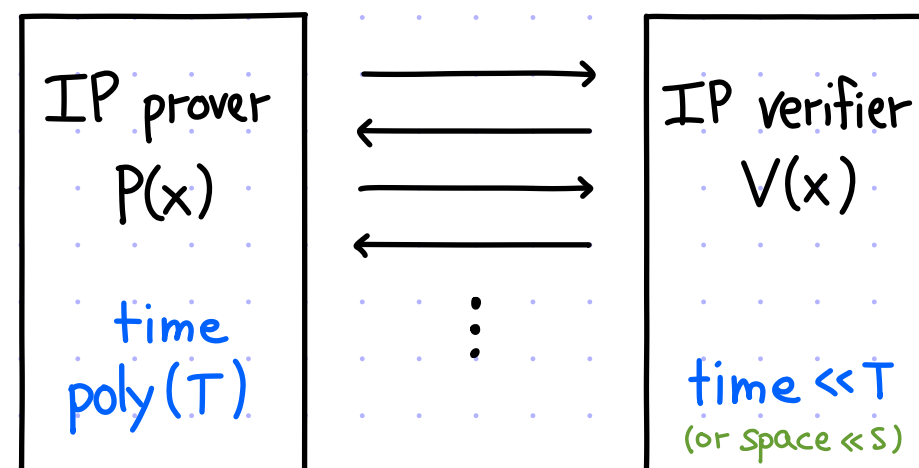
Best time: the verifier $V(x)$ runs in **quasilinear time** $\tilde{O}(|x|)$ (or even linear time $O(|x|)$).

This setting can be viewed as

DELEGATION OF COMPUTATION

from a **weak client** to a **powerful server**.

If $x \in L$ is decidable in time T (& space S)

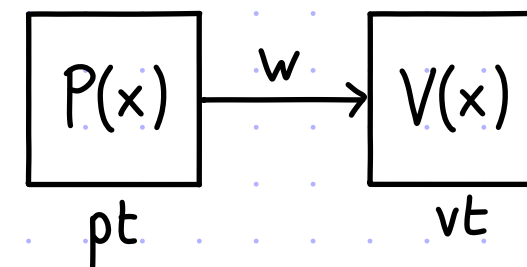


A Lower Bound and An Upper Bound

Q: Which languages have (non-trivial) deIPs?

- A lower bound: doubly-efficient NP proofs

$$\text{NTIME}[pt, vt] := \left\{ L \mid \begin{array}{l} \exists (P, V) \text{ s.t. } \forall x \in L \ V(x, P(x)) = 1 \\ \forall x \notin L \ \forall w \ V(x, w) = 0 \\ P(x), V(x) \text{ run in time } pt(|x|), vt(|x|) \end{array} \right\}.$$



EXAMPLE: $\text{CLIQUE}_t := \{ G \text{ is a graph that contains a clique of size } t \}$

- CLIQUE_t is believed to require (even probabilistic) time $n^{\Omega(t)}$.
- $\text{CLIQUE}_t \in \text{NTIME}[pt = O(n^t), vt = \text{poly}(t, \log n)]$ because
 $P(G)$ finds a clique of size t (if one exists) in time $O(n^t)$
 and $V(G, w)$ checks in time $\text{poly}(t, \log n)$ that w is a t -clique in G .

More generally, $\text{NTIME}[pt = \text{poly}(n), vt = \tilde{O}(n)] \subseteq \text{deIP}$.

- An upper bound: space-bounded computation

theorem: $\text{IP}[\varepsilon_c = 0, vt] \subseteq \text{DSPACE}[O(vt)]$.

Proof idea: modify the analysis of $\text{IP} \subseteq \text{PSPACE}$,
to detect if the optimal prover has a rejecting path.

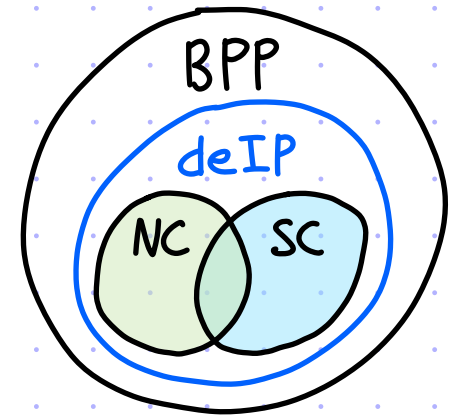
Hence likely $\text{DTIME}[T] \not\subseteq \text{deIP}[\varepsilon_c = 0, vt = o(T)]$ (we do not believe that $\text{DTIME}[T] \subseteq \text{DSPACE}[o(T)]$).

Power Of Interaction: Two Results

Doubly-efficient IPs for bounded depth

$NC := \left\{ L : \begin{array}{l} L \text{ is decidable by } (\log(n)\text{-space uniform}) \text{ boolean circuits} \\ \text{with max fan-in } 2, \text{ size } \text{poly}(n), \text{ and depth } \text{poly}(\log n) \end{array} \right\}$

NC includes arithmetic and linear algebra.



NC = Nick Pippenger's Class
 SC = Steve Cook's Class

theorem: $NC \subseteq \text{deIP}$. Specifically, $NC \subseteq \text{IP}[\varepsilon_c = 0, p_t = \text{poly}(n), v_t = \tilde{O}(n), k = \text{poly}(\log n), cc = \text{poly}(\log n)]$.

[Goldwasser, Kalai, Rothblum 2008]



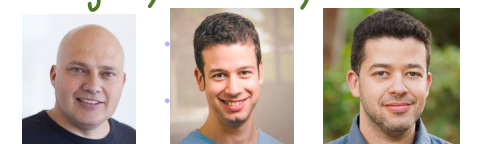
Doubly-efficient IPs for bounded space

$SC := \left\{ L : \begin{array}{l} L \text{ is decidable in deterministic} \\ \text{time } \text{poly}(n) \text{ and space } \text{poly}(\log n) \end{array} \right\} = \text{DTISP}[t = \text{poly}(n), s = \text{poly}(\log n)]$

SC contains deterministic context-free languages (DCFLs), randomized logarithmic space (RL), and bounded-error probabilistic logarithmic space (BPL).

theorem: $SC \subseteq \text{deIP}$. Specifically, $SC \subseteq \text{IP}[\varepsilon_c = 0, p_t = \text{poly}(n), v_t = \tilde{O}(n), k = O(1), cc = o(n)]$.

[Reingold, Rothblum, Rothblum 2016]



Both theorems can be stated more generally (generic size/depth and generic time/space).

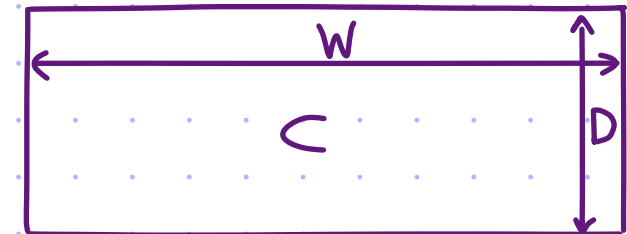
GKR Protocol: Delegation for Bounded-Depth Circuits

def: A circuit family $\{C_n\}_{n \in \mathbb{N}}$ is **s-space uniform** if $\exists$ machine M such that for every $n \in \mathbb{N}$ $M(1^n) = C_n$ and $M(1^n)$ runs in space $O(s(n))$.

theorem: Suppose that L is decidable by a circuit family of width W and depth D that is $O(\log(W \cdot D))$ -space uniform.

Then L has a (public-coin) IP where:

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



size is $W \cdot D$

The case of NC corresponds to $W = \text{poly}(n)$ and $D = \text{poly}(\log n)$.

The proof of the theorem is **technical**.

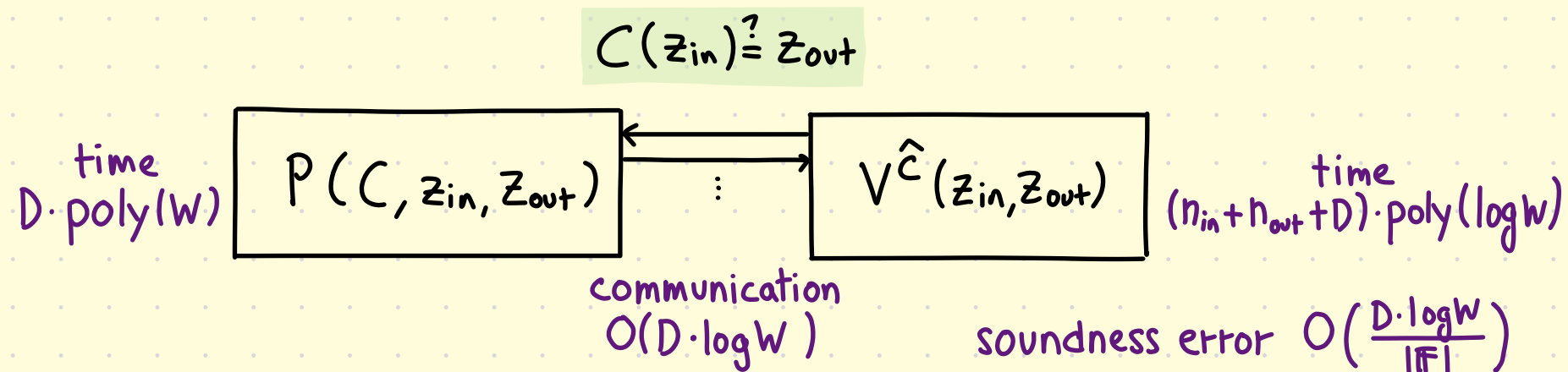
Today we see one piece: **the BARE-BONES GKR protocol**.

Today: The Bare-Bones GKR Protocol

theorem: Let $C: \mathbb{F}^{n_{in}} \rightarrow \mathbb{F}^{n_{out}}$ be a (layered) arithmetic circuit with width W and depth D . Let $\hat{C}$ be a low-degree extension of C . There is a (public-coin) IP for evaluating C where:

to be defined

to be defined



Giving the verifier query access to $\hat{C}$ saves us from discussing uniformity.

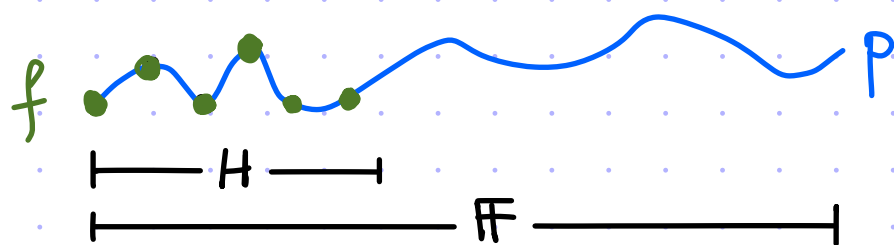
Main ingredients: more arithmetization, more sumcheck, some new ideas.

This protocol is VERY fast in practice!

Low-Degree Extension (Univariate)

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

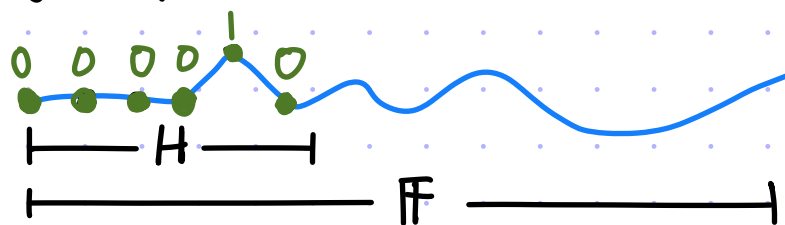
The higher the allowed degree, the more low-degree extensions a function has.

An extension of degree $< |H|$ always exists and is unique.

This extension can be 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 a linear combination according to the given function

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

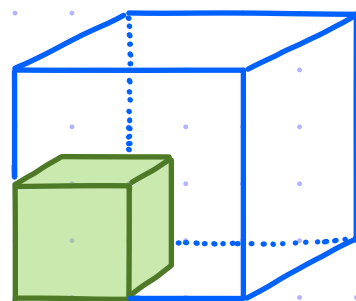
$\forall x \in \mathbb{F}$, can compute $p(x)$ in $\text{poly}(|H|)$ fops

Low-Degree Extension (Multivariate)

The multivariate case builds on the univariate case.

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

We say that $p \in \mathbb{F}[x_1, \dots, x_n]$ extends $f: H^n \rightarrow \mathbb{F}$ if $p|_{H^n} \equiv f$.



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

① consider the multivariate Lagrange polynomials $\{L_{H^n, (\alpha_1, \dots, \alpha_n)}(x_1, \dots, x_n)\}_{\alpha_1, \dots, \alpha_n \in H}$

$$\text{where } L_{H^n, (\alpha_1, \dots, \alpha_n)}(x_1, \dots, x_n) := \prod_{i \in [n]} L_{H, \alpha_i}(x_i) = \prod_{i \in [n]} \prod_{\beta \in H \setminus \{\alpha_i\}} \frac{x_i - \beta}{\alpha_i - \beta}.$$

$\forall x \in \mathbb{F}^n$, can compute $L(x)$ in $\text{poly}(|H|, n)$ fops

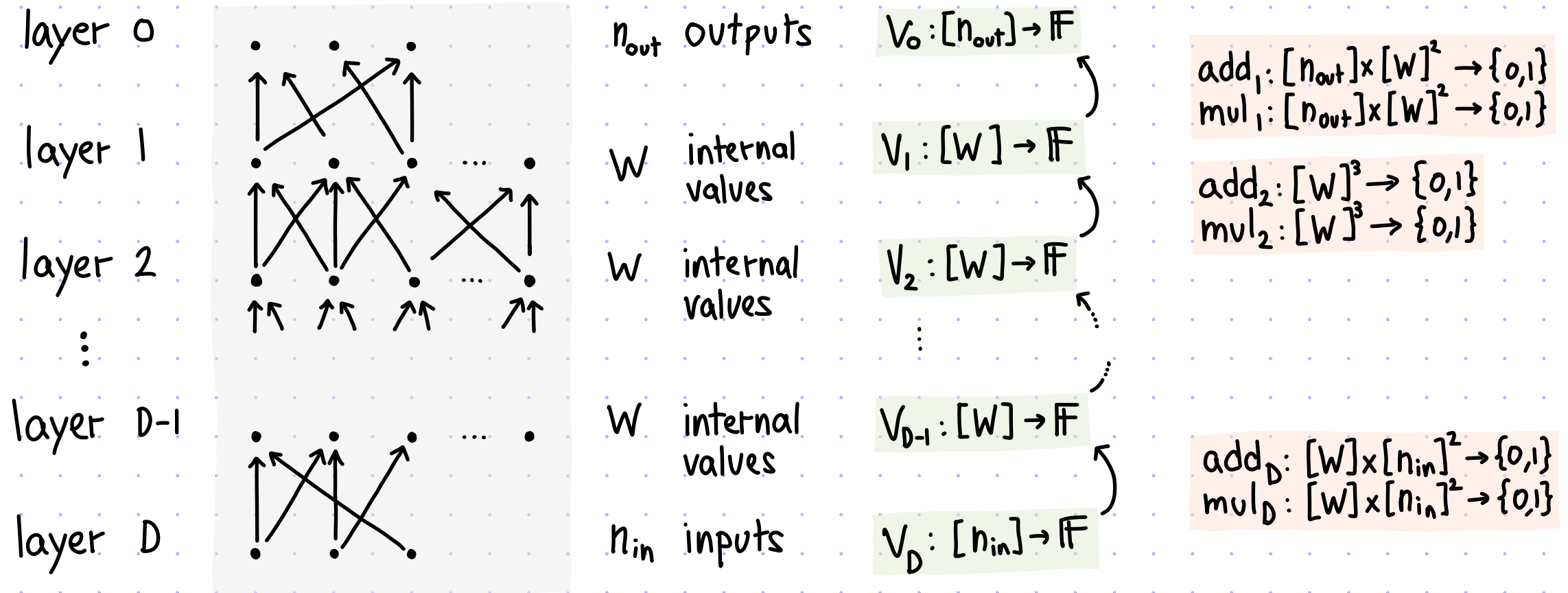
② take the linear combination according to the given function

$$p(x_1, \dots, x_n) := \sum_{\alpha_1, \dots, \alpha_n \in H} f(\alpha_1, \dots, \alpha_n) \cdot L_{H^n, (\alpha_1, \dots, \alpha_n)}(x_1, \dots, x_n).$$

$\forall x \in \mathbb{F}^n$, can compute $p(x)$ in $\text{poly}(|H|^n)$ fops

Layered Arithmetic Circuits

A **layered arithmetic circuit** $C: \mathbb{F}^{n_{in}} \rightarrow \mathbb{F}^{n_{out}}$ of width W and depth D (with $n_{in}, n_{out} \leq W$) is an arithmetic circuit with fan-in 2 arranged in $D+1$ layers:



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 .

NOTATIONAL SIMPLICITY We assume C has only 1 type of gate $g: \mathbb{F}^2 \rightarrow \mathbb{F}$.

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

Arithmetize Each Layer

[1/2]

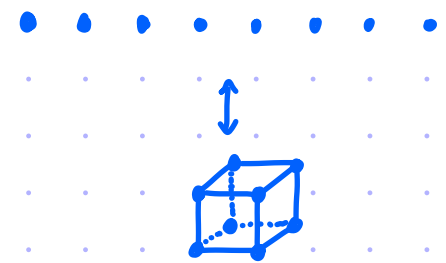
STEP 0: set parameters

Fix a subset $H \subseteq \mathbb{F}$.

Define: $m := \frac{\log W}{\log |H|}$ $m_{in} := \frac{\log n_{in}}{\log |H|}$ $m_{out} := \frac{\log n_{out}}{\log |H|}$

This induces bijections:

$$[W] \leftrightarrow H^m \quad [n_{in}] \leftrightarrow H^{m_{in}} \quad [n_{out}] \leftrightarrow H^{m_{out}}$$



STEP 1: rewrite computation as summations

Let $z_{in} \in \mathbb{F}^{n_{in}}$ be an input to the circuit $C: \mathbb{F}^{n_{in}} \rightarrow \mathbb{F}^{n_{out}}$.

- The input layer $V_D: H^{m_{in}} \rightarrow \mathbb{F}$ is defined as $V_D(a) := z_{in}(a)$.
- For $i = D-1, \dots, 1$: $V_i: H^m \rightarrow \mathbb{F}$ is defined as

$$V_i(a) := \sum_{b, c \in H^m} w_{p_i}(a, b, c) \cdot g(V_{i+1}(b), V_{i+1}(c))$$

- The output layer $V_0: H^{m_{out}} \rightarrow \mathbb{F}$ is defined as

$$V_0(a) := \sum_{b, c \in H^m} w_{p_0}(a, b, c) \cdot g(V_1(b), V_1(c))$$

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

Arithmetize Each Layer

[2/2]

STEP 2: low-degree extend each layer

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

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

- For $i = D-1, \dots, 1$: the extension of the i -th layer is $\hat{V}_i: \mathbb{F}^m \rightarrow \mathbb{F}$ where

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

- The extension of the output layer is $\hat{V}_0: \mathbb{F}^{m_{out}} \rightarrow \mathbb{F}$ where

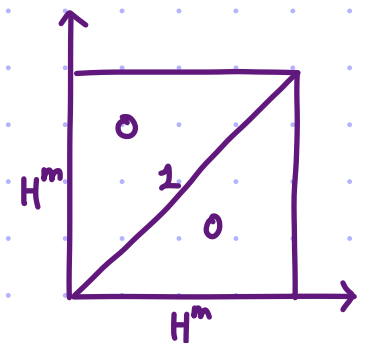
$$\hat{V}_0(x) := \sum_{a \in H^{m_{out}}} V_0(a) \cdot L_{H^{m_{out}}, a}(x) = \sum_{a \in H^{m_{out}}} \left(\sum_{b, c \in H^m} w_1(a, b, c) \cdot g(V_1(b), V_1(c)) \right) \cdot L_{H^{m_{out}}, a}(x)$$

STEP 3: ensure that addend is low-degree

- Replace w_{i+1} and V_{i+1} with their extensions $\hat{w}_{i+1}$ and $\hat{V}_{i+1}$.

This does not change $\hat{V}_i(x)$ because they receive inputs in H .

- Replace $L_{H^m, a}(x)$ with $I_{H^m}(x, a)$ where $I_{H^m}(X, Y) := \prod_{i \in [m]} \sum_{\alpha \in H} L_{H, \alpha}(X_i) \cdot L_{H, \alpha}(Y_i)$.



We obtain: $\hat{V}_i(x) := \sum_{a \in H^m} \left(\sum_{b, c \in H^m} \hat{w}_{i+1}(a, b, c) \cdot g(\hat{V}_{i+1}(b), \hat{V}_{i+1}(c)) \right) \cdot I_{H^m}(x, a)$ (and similarly for $\hat{V}_0$)

Batch Output Claims

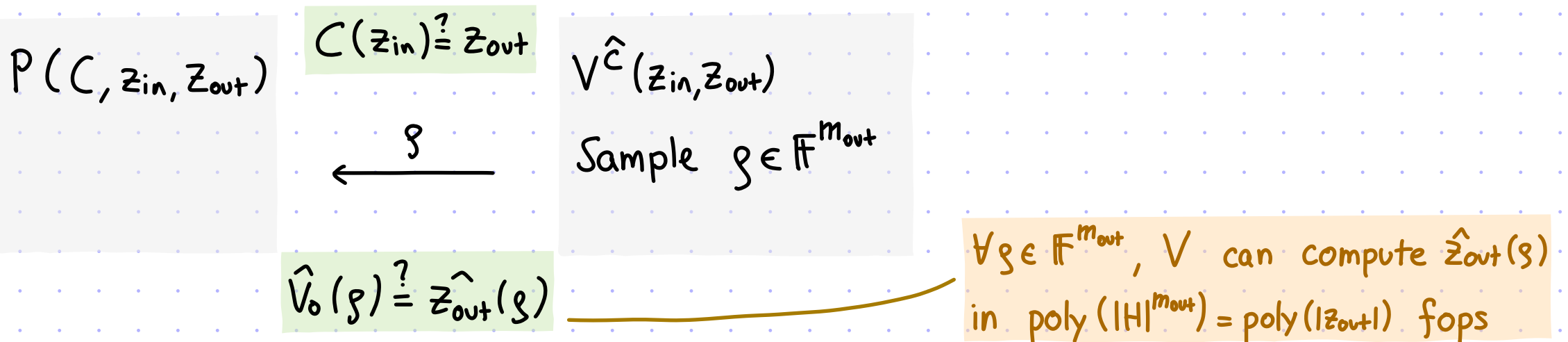
The statement " $C(z_{in}) = z_{out}$ " is equivalent to these $n_{out} = |H|^{m_{out}}$ statements:

$$\forall a \in H^{m_{out}}, V_0(a) = z_{out}(a).$$

In turn, this is equivalent to:

$$\forall a \in H^{m_{out}}, \hat{V}_0(a) = \hat{z}_{out}(a).$$

If $n_{out} > 1$ we batch these into a single statement about $\hat{V}_0$:



Completeness: if $C(z_{in}) = z_{out}$ then $\forall g \in \mathbb{F}^{m_{out}} \hat{V}_0(g) = \hat{z}_{out}(g)$.

Soundness: if $C(z_{in}) \neq z_{out}$ then $\hat{V}_0 \neq \hat{z}_{out}$ so $\Pr_g[\hat{V}_0(g) = \hat{z}_{out}(g)] \leq \frac{n_{out} \cdot (|H| - 1)}{|\mathbb{F}|}$.

Check Computation via Iterated Sumchecks

The statement " $C(z_{in}) = z_{out}$ " is rewritten as " $\hat{V}_0(g) = \hat{z}_{out}(g)$ " for a random $g \in \mathbb{F}^{m_{out}}$.

Equivalently,

$$\sum_{a \in H^{m_{out}}, b, c \in H^m} \hat{w}_p(a, b, c) \cdot g(\hat{V}_1(b), \hat{V}_1(c)) \cdot I_{H^{m_{out}}}(g, a) = \hat{z}_{out}(g)$$

Next do a **sumcheck protocol** on variables for a, b, c .

- round complexity $m_{out} + 2m$ (the sum is over $H^{m_{out} + 2m}$)
- soundness error $O\left(m \cdot \frac{|H| \cdot \deg_{ind}(g)}{|\mathbb{F}|}\right)$ (individual degrees are $< |H| \cdot \deg_{ind}(g) + |H|$)
- prover time is $\text{poly}(|H|^m) = \text{poly}(W)$ (THIS IS EFFICIENT!)
- verifier time is $\text{poly}(|H|, m) = \text{poly}(\log W)$ **given answers to:**
 - 1 query to $\hat{w}_p: \mathbb{F}^{m_{out} + 2m} \rightarrow \mathbb{F}$ at $(r, s, t) \in \mathbb{F}^{m_{out} + 2m} \leftarrow$ we assumed that the verifier has this oracle
 - 2 queries to $\hat{V}_1: \mathbb{F}^m \rightarrow \mathbb{F}$ at $s, t \in \mathbb{F}^m \leftarrow$ the prover sends claimed answers $\gamma, \delta \in \mathbb{F}$

We recurse on two claims: " $\hat{V}_1(s) = \gamma$ " and " $\hat{V}_1(t) = \delta$ "

Each claim is itself a sum:

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

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

PROBLEM: the number of claims doubles at each layer

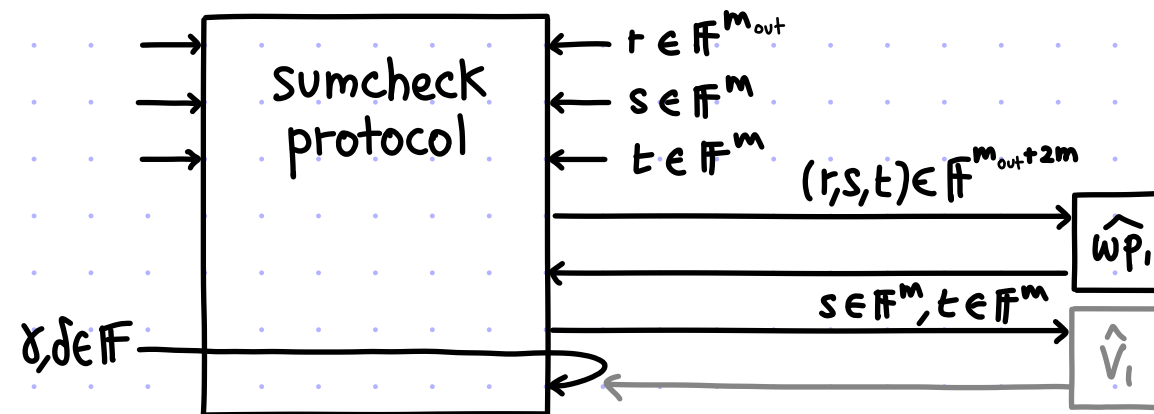
Avoiding Claim Blowup

[1/2]

We batch the two claims via a **random linear combination**.

1 claim
about layer 0

$$\sum_{a \in H^{m_{\text{out}}}, b, c \in H^m} \hat{w}_{p_1}(a, b, c) \cdot g(\hat{V}_1(b), \hat{V}_1(c)) \cdot I_{H^{m_{\text{out}}}}(g, a) = \hat{z}_{\text{out}}(g)$$



2 claims
about layer 1

$$\sum_{a, b, c \in H^m} \hat{w}_{p_2}(a, b, c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot I_{H^m}(s, a) = \gamma$$

$$\sum_{a, b, c \in H^m} \hat{w}_{p_2}(a, b, c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot I_{H^m}(t, a) = \delta$$

$$\xleftarrow{\alpha, \beta \in \mathbb{F}} \quad \alpha, \beta \leftarrow \mathbb{F}$$

1 claim
about layer 1

$$\sum_{a, b, c \in H^m} \hat{w}_{p_2}(a, b, c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot [\alpha \cdot I_{H^m}(s, a) + \beta \cdot I_{H^m}(t, a)] = \alpha \cdot \gamma + \beta \cdot \delta$$

} random linear
combination
of the two claims

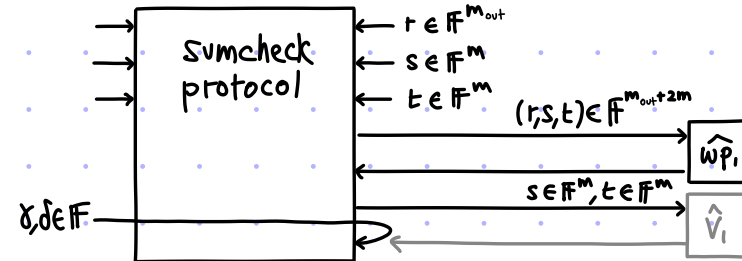
BUT... the new claim has a different form than the claim we started with!

Avoiding Claim Blowup

[2/2]

1 claim
about layer 0

$$\sum_{a \in H^{m_{\text{out}}}, b, c \in H^m} \hat{w}_{p_1}(a, b, c) \cdot g(\hat{V}_1(b), \hat{V}_1(c)) \cdot I_{H^{m_{\text{out}}}}(p, a) = \hat{z}_{\text{out}}(p)$$



2 claims
about layer 1

$$\sum_{a, b, c \in H^m} \hat{w}_{p_2}(a, b, c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot I_{H^m}(s, a) = \gamma$$

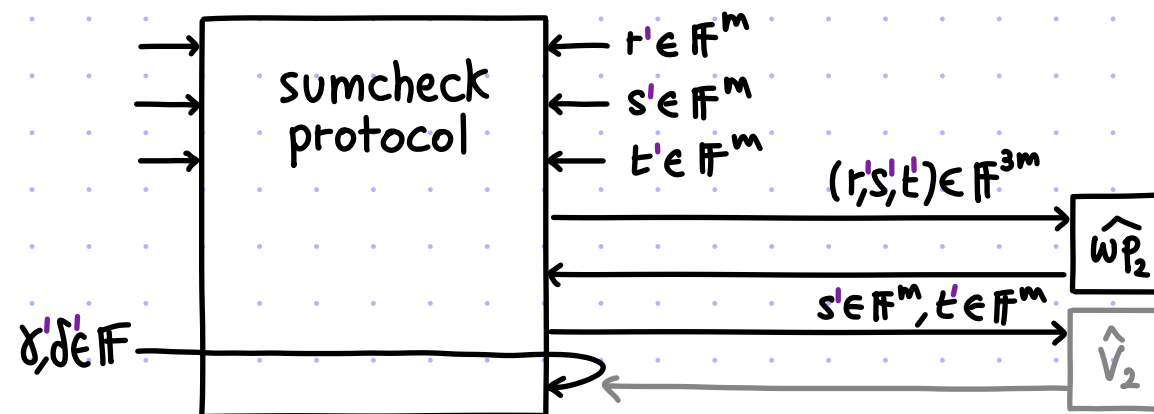
$$\sum_{a, b, c \in H^m} \hat{w}_{p_2}(a, b, c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot I_{H^m}(t, a) = \delta$$

$$\xleftarrow{\alpha, \beta \in \mathbb{F}} \quad \alpha, \beta \leftarrow \mathbb{F}$$

1 claim
about layer 1

$$\sum_{a, b, c \in H^m} \hat{w}_{p_2}(a, b, c) \cdot g(\hat{V}_2(b), \hat{V}_2(c)) \cdot [\alpha \cdot I_{H^m}(s, a) + \beta \cdot I_{H^m}(t, a)] = \alpha \cdot \gamma + \beta \cdot \delta$$

} random linear
combination
of the two claims



$$\sum_{a, b, c \in H^m} \hat{w}_{p_3}(a, b, c) \cdot g(\hat{V}_3(b), \hat{V}_3(c)) \cdot I_{H^m}(s', a) = \gamma'$$

$$\sum_{a, b, c \in H^m} \hat{w}_{p_3}(a, b, c) \cdot g(\hat{V}_3(b), \hat{V}_3(c)) \cdot I_{H^m}(t', a) = \delta'$$

2 claims
about layer 2

We run another
sumcheck protocol ...
and we get two claims
of the **same form** as before.
⇒ We can recurse!

GKR Bare Bones Protocol

- round complexity:

$$1 + D \cdot ([\text{SC on } 3m \text{ vars}] + 1)$$

$$= O(D \cdot m) = O\left(D \cdot \frac{\log W}{\log |H|}\right) = O(D \cdot \log W)$$

- communication complexity (in field elements):

$$1 + D \cdot ([\text{SC on } 3m \text{ vars of degree } O(|H|)] + 4)$$

$$= O(D \cdot m \cdot |H|) = O\left(D \cdot \frac{\log W}{\log |H|} \cdot |H|\right) = O(D \cdot \log W)$$

- soundness error:

$$\frac{m_{\text{out}} \cdot |H|}{|F|} + D \cdot ([\text{SC on } 3m \text{ vars of degree } O(|H|)] + \frac{1}{|F|})$$

$$= O\left(D \cdot \frac{m \cdot |H|}{|F|}\right) = O\left(D \cdot \frac{\log W \cdot |H|}{\log |H| \cdot |F|}\right) = O\left(D \cdot \frac{\log W}{|F|}\right)$$

- prover time (in field operations):

$$D \cdot ([\text{SC on } 3m \text{ vars of degree } O(|H|)] + \text{small})$$

$$= D \cdot \text{poly}(|H|^m) = D \cdot \text{poly}(W)$$

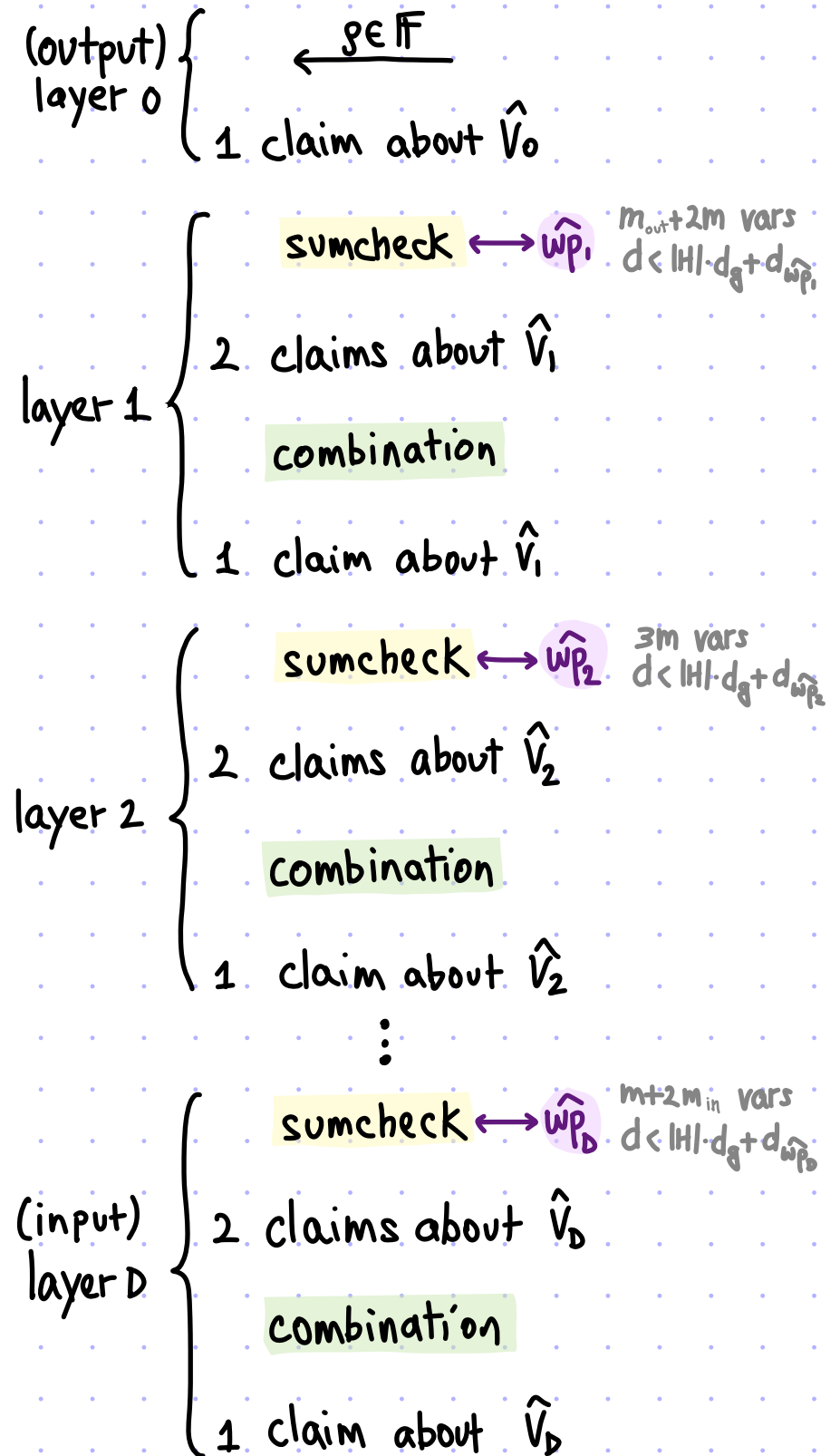
- verifier time (in field operations):

$$n_{\text{in}} \cdot \text{poly}(m_{\text{in}}, |H|) + n_{\text{out}} \cdot \text{poly}(m_{\text{out}}, |H|) + D \cdot ([\text{SC on } 3m \text{ vars of degree } O(|H|)])$$

$$= (n_{\text{in}} + n_{\text{out}} + D) \cdot \text{poly}(m, |H|) = (n_{\text{in}} + n_{\text{out}} + D) \cdot \text{poly}\left(\frac{\log W}{\log |H|}, |H|\right) = (n_{\text{in}} + n_{\text{out}} + D) \cdot \text{poly}(\log W)$$

For simplicity:
 $d_g = 1$ and $d_{\hat{w}_i} < |H|$

$$P(C, z_{\text{in}}, z_{\text{out}}) \quad V^{\hat{C}}(z_{\text{in}}, z_{\text{out}})$$



obtained by
 setting $H = \{0, 1\}$

Bibliography

GKR Protocol

- [GKR 2008]: [Delegating computation: interactive proofs for muggles](#), by Shafi Goldwasser, Yael Kalai, Guy Rothblum. (▶[Video](#)).
- [CMT 2011]: [Practical verified computation with streaming interactive proofs](#), by Graham Cormode, Michael Mitzenmacher, Justin Thaler.
- [TMRP 2012]: [Verifiable computation with massively parallel interactive proofs](#), by Justin Thaler, Mike Roberts, Michael Mitzenmacher, Hanspeter Pfister.
- [Thaler 2013]: [Time-optimal interactive proofs for circuit evaluation](#), by Justin Thaler.
- [WHGSW 2015]: [Verifiable ASICs](#), by Riad Wahby, Max Howald, Siddharth Garg, abhi shelat, Michael Walfish.
- [Thaler 2017]: [Notes on efficient GKR](#), by Justin Thaler.

Other doubly-efficient IPs

- [RRR 2016]: [Constant-round interactive proofs for delegating computation](#), by Omer Reingold, Guy Rothblum, Ron Rothblum. (▶[Video1](#), [Video2](#)), (▶[Video3](#)).
- [GR 2018]: [Simple doubly-efficient interactive proof systems for locally-characterizable sets](#), by Oded Goldreich, Guy Rothblum.
- [GR 2020]: [Constant-round interactive proof systems for \$AC^0\[2\]\$ and \$NC^1\$](#) , by Oded Goldreich, Guy Rothblum.
- [GR 2018]: [Counting t-cliques: worst-case to average-case reductions and direct interactive proof systems](#), by Oded Goldreich, Guy Rothblum. (▶[Video](#))

Surveys

- [Goldreich 2018]: [On doubly-efficient interactive proof systems](#), by Oded Goldreich.
- [Rothblum 2019]: [Doubly efficient interactive proofs](#), by Guy Rothblum.