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 V.

TODAY: what about the honest prover P?

Let φ be an n-variable boolean formula.

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

Are these times useful for computations of interest?

For example, 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 $|\varphi| = \text{poly}(\log T, S)$ and $n \geq (\log T) \cdot S$ variables.

Even if $T, S = \text{poly}(|x|)$, the honest prover P runs in time

$$\text{time}(P) = \Omega(2^n) = \Omega(T^S) = |x|^{\omega(1)}.$$

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: $\mathsf{deIP} \subseteq \mathsf{BPP}$

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, ...

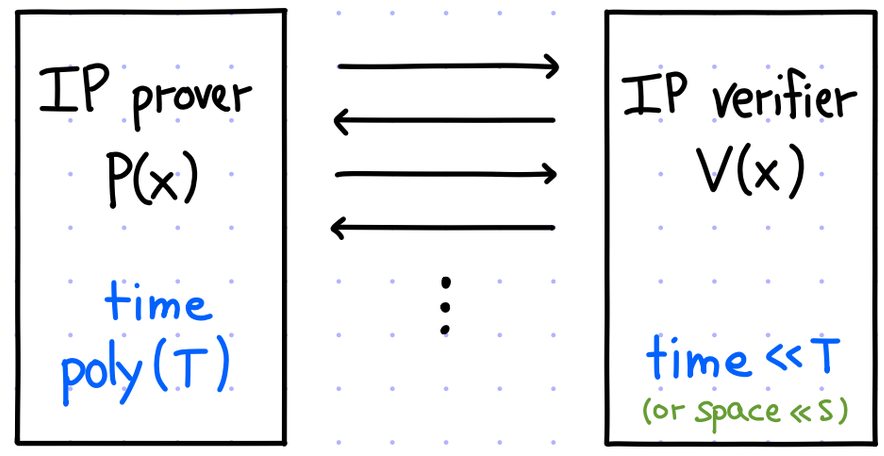
Best time: the verifier $V(x)$ runs in quasilinear time $\tilde{O}(|x|)$ (or even linear time $O(|x|)$).

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)

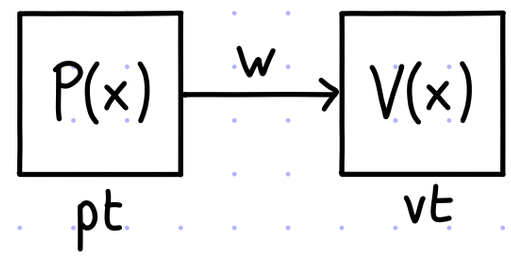


A Lower Bound and An Upper Bound

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

- A lower bound: doubly-efficient NP proofs

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



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

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

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

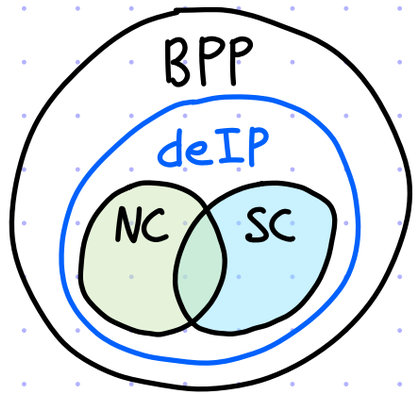
- An upper bound: space-bounded computation

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

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

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

Power Of Interaction: Two Results



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

Doubly-efficient IPs for bounded depth

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

NC includes arithmetic and linear algebra.

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

[Goldwasser, Kalai, Rothblum 2008]



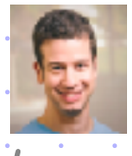
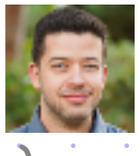
Doubly-efficient IPs for bounded space

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

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

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

[Reingold, Rothblum, Rothblum 2016]

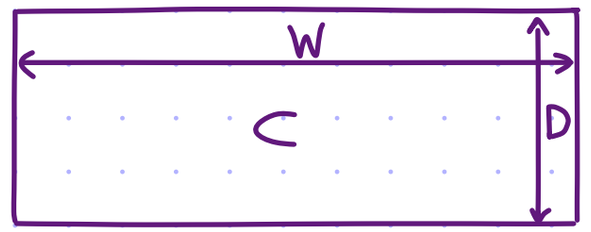
Both theorems can be stated more generally (generic size/depth and generic time/space).

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.

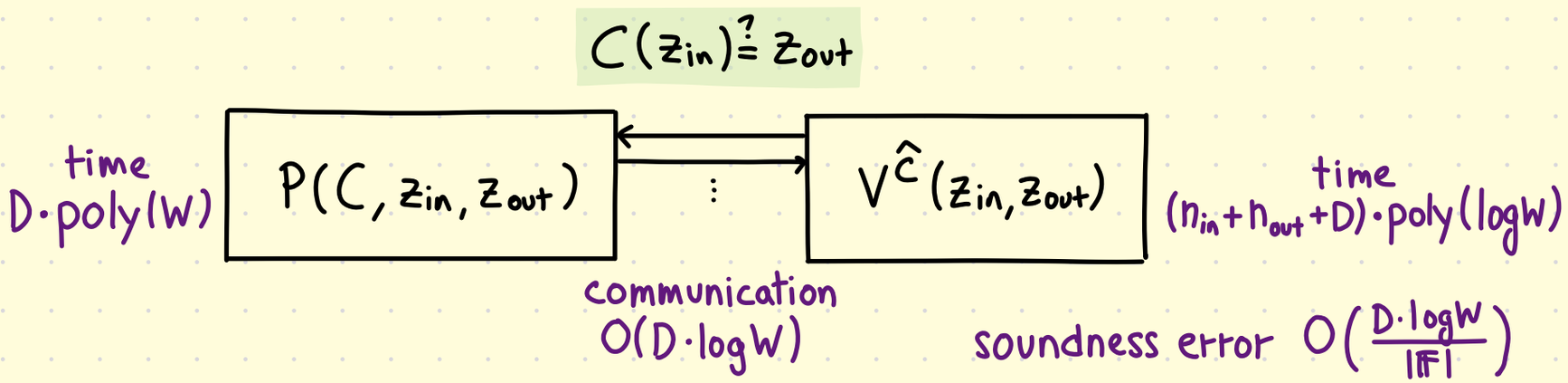
Today: The Bare-Bones GKR Protocol

to be defined

to be defined

theorem: Let $C: \mathbb{F}^{n_{in}} \to \mathbb{F}^{n_{out}}$ 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:



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

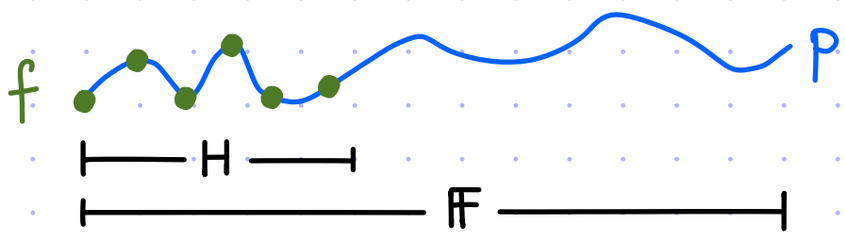
Main ingredients: more arithmetization, more sumcheck, some new ideas.

This protocol is VERY fast in practice!

Low-Degree Extension (Univariate)

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

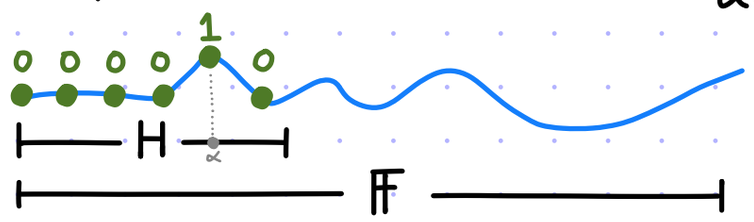
The higher the allowed degree, the more low-degree extensions a function has.

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

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

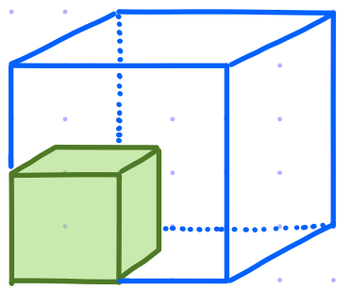
$\forall \gamma \in \mathbb{F}$, can compute $p(\gamma)$ in $\text{poly}(|H|)$ fops

Low-Degree Extension (Multivariate)

The multivariate case builds on the univariate case.

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

We say that $p \in \mathbb{F}[X_1,\dots,X_n]$ 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 $\{L_{H^n,(\alpha_1,\dots,\alpha_n)}(X_1,\dots,X_n)\}_{\alpha_1,\dots,\alpha_n \in H}$

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

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

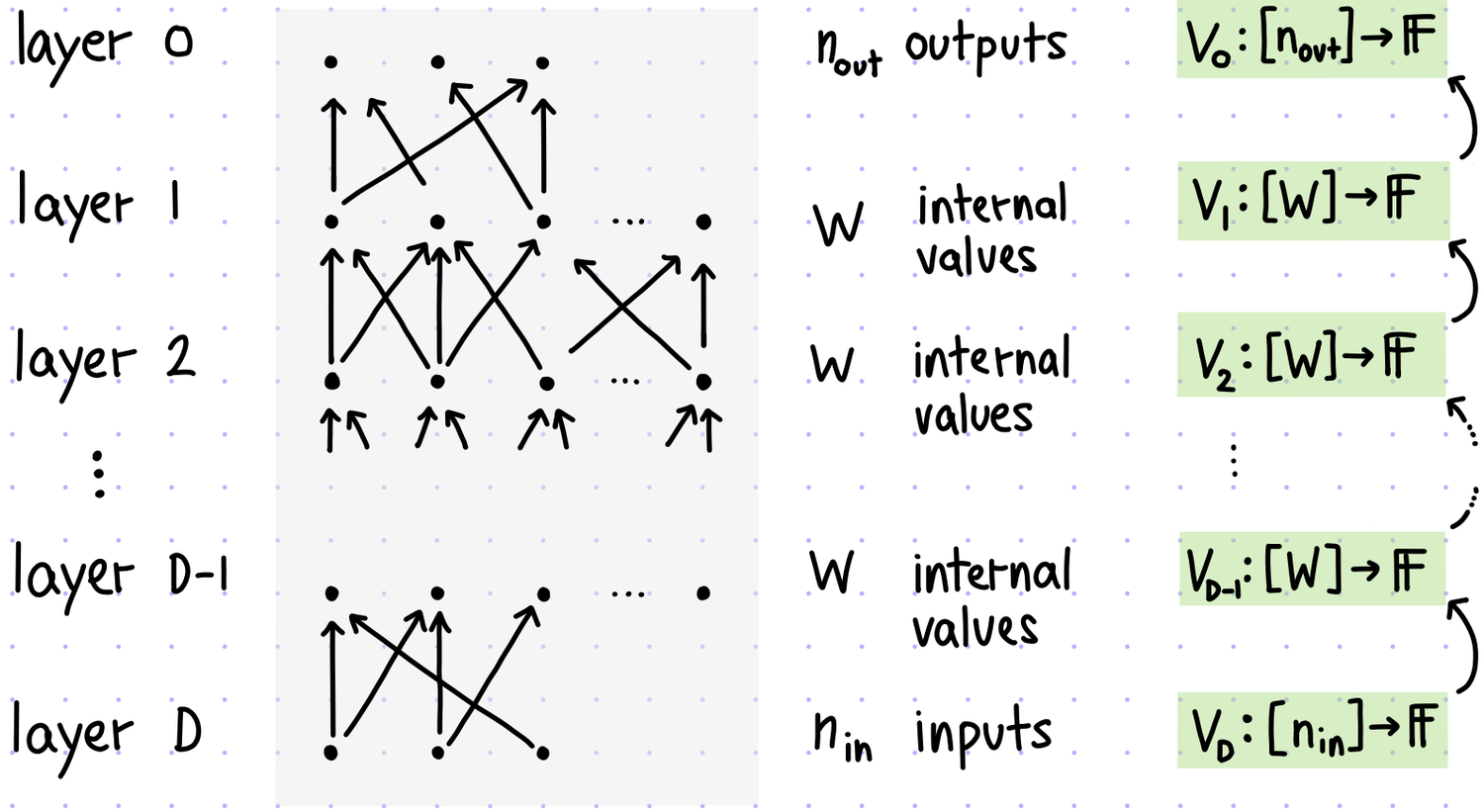
② take the linear combination according to the given function

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

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

Layered Arithmetic Circuits

A layered arithmetic circuit $C: \mathbb{F}^{n_{in}} \to \mathbb{F}^{n_{out}}$ 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 1 type of gate $g: \mathbb{F}^2 \to \mathbb{F}$. (Extending to multiple gate types is straightforward.)

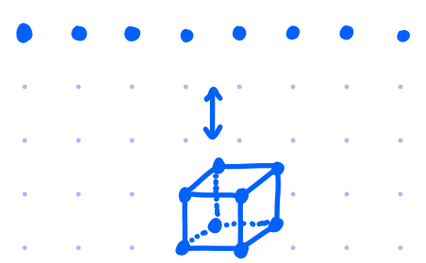
We let $(wp_1, \dots, wp_D)$ be the wiring predicates for g.

Arithmetize Each Layer [1/2]

STEP 0: set parameters

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

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



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 $C : \mathbb{F}^{n_{in}} \to \mathbb{F}^{n_{out}}$.

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

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

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

$$V_0(a) := \sum_{b,c \in H^m} wp_1(a,b,c) \cdot g\left(V_1(b), V_1(c)\right).$$

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 $\hat{V}_D : \mathbb{F}^{m_{in}} \to \mathbb{F}$ 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 $i = D-1, \ldots, 1$: the extension of the i-th layer is $\hat{V}_i : \mathbb{F}^m \to \mathbb{F}$ where

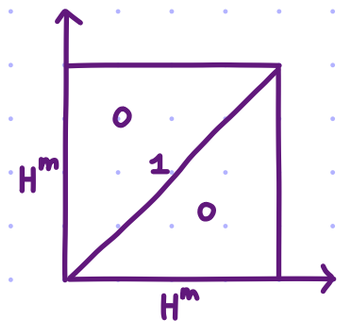
$$\hat{V}_i(x) := \sum_{a \in H^m} V_i(a) \cdot L_{H^m,a}(x) = \sum_{a \in H^m} \left(\sum_{b,c \in H^m} wp_{i+1}(a,b,c) \cdot g\left(V_{i+1}(b), V_{i+1}(c)\right) \right) \cdot L_{H^m,a}(x)$$

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

$$\hat{V}_o(x) := \sum_{a \in H^{m_{out}}} V_o(a) \cdot L_{H^{m_{out}},a}(x) = \sum_{a \in H^{m_{out}}} \left(\sum_{b,c \in H^m} wp_1(a,b,c) \cdot g\left(V_1(b), V_1(c)\right) \right) \cdot L_{H^{m_{out}},a}(x)$$

STEP 3: ensure that addend is low-degree

- Replace wp_{i+1} and V_{i+1} with their extensions $\hat{wp}_{i+1}$ 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: $$\hat{V}_i(x) := \sum_{a \in H^m} \left(\sum_{b,c \in H^m} \hat{wp}_{i+1}(a,b,c) \cdot g\left(\hat{V}_{i+1}(b), \hat{V}_{i+1}(c)\right) \right) \cdot I_{H^m}(x,a)$$ (and similarly for $\hat{V}_o$)

Batch Output Claims

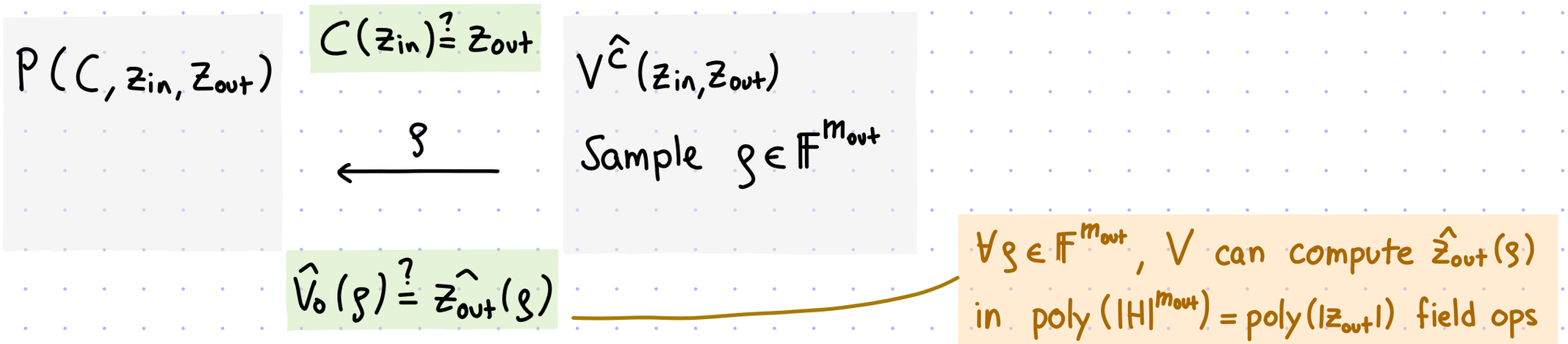
The statement "$C(z_{in}) = z_{out}$" is equivalent to these $n_{out} = |H|^{m_{out}}$ statements:

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

In turn, this is equivalent to:

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

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



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

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

Check Computation via Iterated Sumchecks

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

Equivalently, $$\sum_{a \in H^{m_{out}},\, b,c \in H^m} \widehat{wp}_1(a,b,c) \cdot g\left(\hat{V}_1(b), \hat{V}_1(c)\right) \cdot I_{H^{m_{out}}}(\rho, a) = \hat{z}_{out}(\rho)$$

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 $\widehat{wp}_1 : \mathbb{F}^{m_{out}+2m} \to \mathbb{F}$ at $(r,s,t) \in \mathbb{F}^{m_{out}+2m}$ ← the verifier has this oracle by assumption
  - 2 queries to $\hat{V}_1 : \mathbb{F}^m \to \mathbb{F}$ at $s,t \in \mathbb{F}^m$ ← 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:

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

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

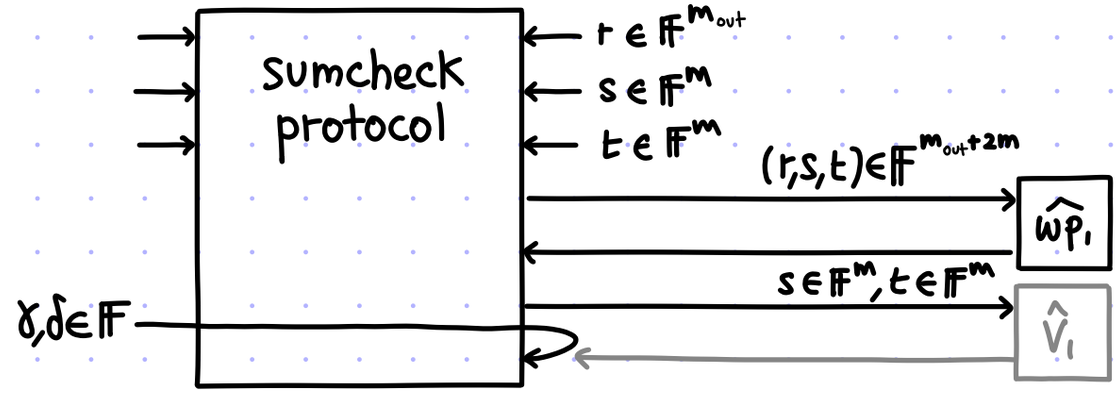
PROBLEM: the number of claims doubles at each layer

Avoiding Claim Blowup

We batch the two claims via a random linear combination.

1 claim about layer 0

$$\sum_{a \in H^{m_{out}},\, b,c \in H^m} \widehat{wp}_1(a,b,c) \cdot g\left(\hat{V}_1(b), \hat{V}_1(c)\right) \cdot I_{H^{m_{out}}}(\rho, a) = \hat{z}_{out}(\rho)$$



2 claims about layer 1

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

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

$\alpha, \beta \in F$ ← $\alpha, \beta \leftarrow F$

1 claim about layer 1

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

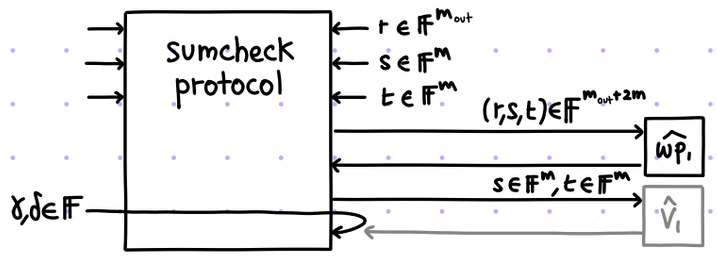
} random linear combination of the two claims

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

Avoiding Claim Blowup

1 claim about layer 0

$$\sum_{a \in H^{m_{out}},\, b,c \in H^m} \widehat{wp}_1(a,b,c) \cdot g\left(\hat{V}_1(b), \hat{V}_1(c)\right) \cdot I_{H^{m_{out}}}(\rho, a) = \hat{z}_{out}(\rho)$$



2 claims about layer 1

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

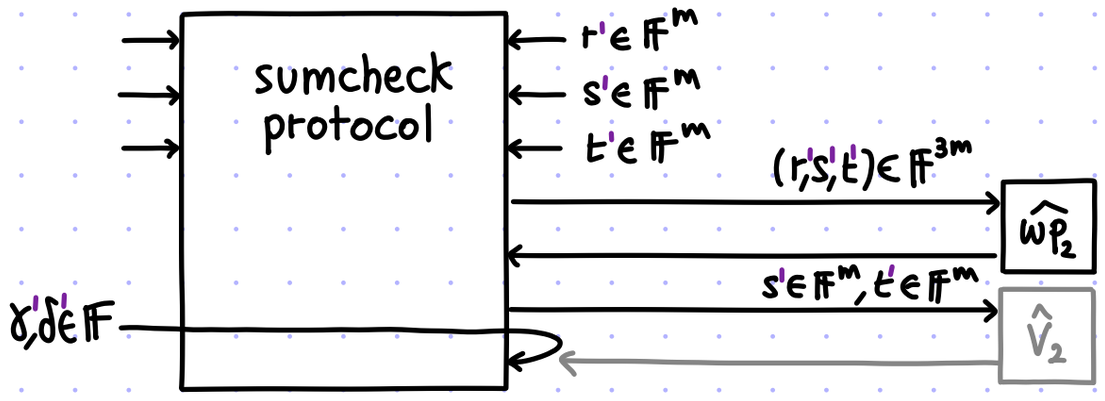
$$\sum_{a,b,c \in H^m} \widehat{wp}_2(a,b,c) \cdot g\left(\hat{V}_2(b), \hat{V}_2(c)\right) \cdot I_{H^m}(t,a) = \delta$$

$\alpha, \beta \in F$ ← $\alpha, \beta \leftarrow F$

1 claim about layer 1

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

} random linear combination of the two claims



2 claims about layer 2

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

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

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

GKR Bare Bones Protocol

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

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

- round complexity:

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

  $= O(D \cdot m) = O(D \cdot \frac{\log W}{\log |H|}) = O(D \cdot \log W)$ obtained by setting $H = \{0,1\}$

- 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:

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

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

- prover time (in field 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):

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

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

(output) layer 0
- ← $g \in \mathbb{F}$
- 1 claim about $\hat{V}_0$

layer 1 ($m_{out} + 2m$ vars, $d < |H| \cdot \deg_{ind}(g) + \deg_{ind}(\hat{wp}_1)$)
- sumcheck ↔ $\hat{wp}_1$
- 2 claims about $\hat{V}_1$
- combination
- 1 claim about $\hat{V}_1$

layer 2 ($3m$ vars, $d < |H| \cdot \deg_{ind}(g) + \deg_{ind}(\hat{wp}_2)$)
- sumcheck ↔ $\hat{wp}_2$
- 2 claims about $\hat{V}_2$
- combination
- 1 claim about $\hat{V}_2$

⋮

(input) layer D ($m + 2m_{in}$ vars, $d < |H| \cdot \deg_{ind}(g) + \deg_{ind}(\hat{wp}_D)$)
- sumcheck ↔ $\hat{wp}_D$
- 2 claims about $\hat{V}_D$
- combination
- 1 claim about $\hat{V}_D$

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 AC0[2] and NC1, 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.