Lecture 8: GKR protocol analysis and IOPs

Zero-knowledge proofs

263-4665-00L

Lecturer: Jonathan Bootle

Graded homework postponed

- Sheet still needs to be thoroughly debugged.
- Also want to make sure all necessary material is covered in today's lecture.

- Will post graded sheet soon on Moodle and extend deadline accordingly.

Multilinear extensions

Definition: Given $f : \{0,1\}^\ell \to \mathbb{F}$, a polynomial $\tilde{f}(X_1, \ldots, X_\ell) \in \mathbb{F}[X_1, \ldots, X_\ell]$ satisfying

- $\deg_{X_i} \tilde{f} = 1, \forall i \in [\ell]$, and
- $f(\vec{\omega}) = \tilde{f}(\vec{\omega}) \; \forall \vec{\omega} \in \{0,1\}^\ell$

is called the *multilinear extension* (MLE) of f.

Fact: $\tilde{f}$ exists for any such f and is *unique*.

For vectors $\vec{a} \in \mathbb{F}^N$ $\leftrightarrow$ $a : \{0,1\}^{\log N} \to \mathbb{F}$, $a(\vec{j}) \coloneqq (\vec{a})_j \; \forall j \in [N]$, $\vec{j} \coloneqq \text{Binary}(j)$ $\leftrightarrow$ $\tilde{a} \in \mathbb{F}[X_1, \ldots, X_{\log N}]$

Polynomial definitions and facts

- $\mathrm{EQ} : \{0,1\}^2 \to \{0,1\}$. $\mathrm{EQ}(i;j) \coloneqq (i == j)$.
- $\widetilde{\mathrm{EQ}}(X;Y) = XY + (1-X)(1-Y)$.

- Extend to ℓ inputs.
- $\mathrm{EQ} : \{0,1\}^{2\ell} \to \{0,1\}$. $\mathrm{EQ}(\vec{\imath};\vec{\jmath}) \coloneqq (\vec{\imath} == \vec{\jmath})$. Symmetric in $\vec{X}, \vec{Y}$
- $\widetilde{\mathrm{EQ}}\left(\vec{X};\vec{Y}\right) = \prod_{j=1}^{\ell} \widetilde{\mathrm{EQ}}(X_j;Y_j)$, where $\vec{X}$ is shorthand for variables $X_1, \ldots, X_\ell$ and similarly for $\vec{Y}$.
- Explicit MLEs for vectors and matrices:

Multivariate Lagrange basis polynomials

- If $\vec{a} \in \mathbb{F}^N$, $\tilde{a}(X_1, \ldots, X_{\log N}) \coloneqq \sum_{\vec{\imath} \in \{0,1\}^{\log N}} a_i \cdot \widetilde{\mathrm{EQ}}\left(\vec{X};\vec{\imath}\right)$.
- If $A \in \mathbb{F}^{N \times N}$, $\tilde{A}(X_1, \ldots, X_{\log N}, Y_1, \ldots, Y_{\log N}) \coloneqq \sum_{\vec{\imath},\vec{\jmath} \in \{0,1\}^{\log N}} A_{ij} \cdot \widetilde{\mathrm{EQ}}\left(\vec{X},\vec{Y};\vec{\imath},\vec{\jmath}\right)$.

$\vec{\imath} = \mathrm{Binary}(i)$

Evaluating multilinear polynomials

Fact:

$\widetilde{\mathrm{EQ}}\left(\vec{X};\vec{Y}\right) = \prod_{j=1}^{\ell} \widetilde{\mathrm{EQ}}(X_j;Y_j)$ costs $O(\ell)$ $\mathbb{F}$-ops to evaluate over $\mathbb{F}$.

Lemma:

Given $\vec{r} \in \mathbb{F}^{\ell}$, it costs $O(2^{\ell})$ $\mathbb{F}$-ops to compute $\left\{\widetilde{\mathrm{EQ}}(\vec{i};\vec{r})\right\}_{\vec{i}\in\{0,1\}^{\ell}}$.

Proof:

- It costs $O(\ell)$ $\mathbb{F}$-ops to compute $\widetilde{\mathrm{EQ}}(0,r_j), \widetilde{\mathrm{EQ}}(1,r_j)$ for $j \in [\ell]$.
- Initialise a table with $\widetilde{\mathrm{EQ}}(0,r_1), \widetilde{\mathrm{EQ}}(1,r_1)$.
- Given a table of length 2^t containing $\prod_{j=1}^{t} \widetilde{\mathrm{EQ}}(i_j;r_j)$ for each $i_1,\dots,i_t \in \{0,1\}$, multiply by $\widetilde{\mathrm{EQ}}(0,r_{t+1}), \widetilde{\mathrm{EQ}}(1,r_{t+1})$.
- Total cost is $O(2^{\ell})$ $\mathbb{F}$-ops.

Plan for the next few lectures

Sumcheck protocol ✓

coNP ⊆ IP ✓
Extends to #SAT
(count SAT inputs)
Shows **#P ⊆ IP**

Interactive proof for layered arithmetic circuits

Full completeness and soundness
Proof size and V complexity

Interactive oracle proofs for general arithmetic circuits

$\mathrm{DescriptionSize}(C) = N$
Indirect access to instance

V makes queries to instance

Commitments and polynomial commitments

ZK arguments for layered arithmetic circuits

ZK arguments for general arithmetic circuits

GKR protocol overview

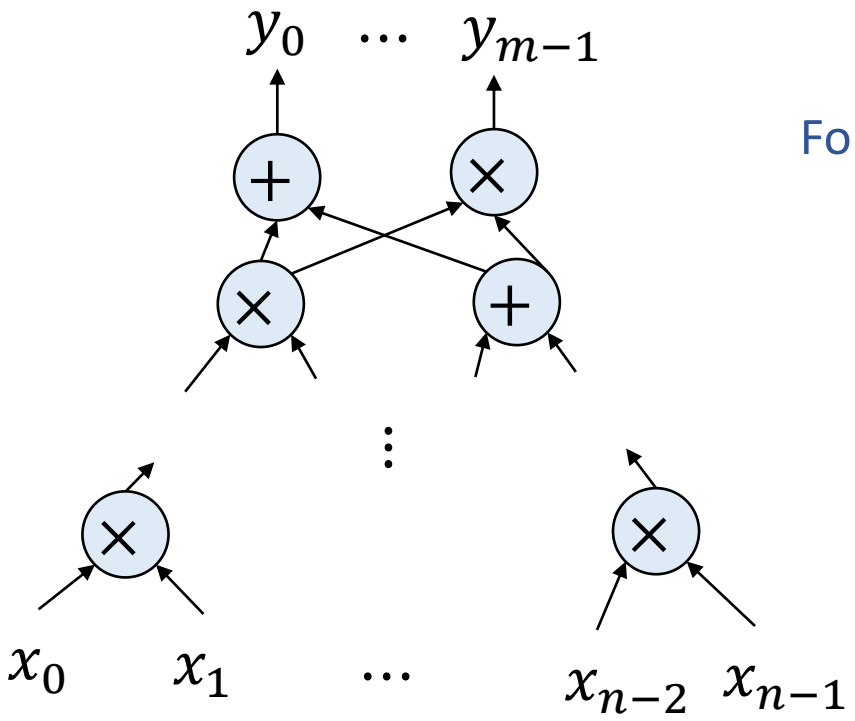
Idea: successively reduce claims about level i to level $i+1$

Instance:

- $\left(\{\text{add}_i, \text{mult}_i\}_{i=0}^{D-1}, \vec{x}, \vec{y}\right)$

Language:

- satisfies $C(\vec{x}) = \vec{y}$



Initial claim: $\widetilde{w}_0 \equiv \tilde{y}$

P — Initial reduction — V

Claim: $\widetilde{w}_0(\vec{r}_0) = v_0$

For $i = 1, \ldots, D$:

Claim: $\widetilde{w}_i(\vec{r}_i) = v_i$

P — sumcheck — V

Claims: $\widetilde{w}_{i+1}(\vec{\beta}_{i+1}) = B_{i+1}$, $\widetilde{w}_{i+1}(\vec{\gamma}_{i+1}) = C_{i+1}$

P — 2-to-1 reduction — V

Claim: $\widetilde{w}_{i+1}(\vec{r}_{i+1}) = v_{i+1}$

Check final claim using $\vec{x}$

Final claim: $\widetilde{w}_D(\vec{r}_D) = v_D$

Initial reduction: many output checks to one

Claimed circuit outputs: $\{y(\vec{z})\}_{\vec{z}\in\{0,1\}^{\ell_0}}$.

Want to check $w_0(\vec{z}) = y(\vec{z}), \forall \vec{z} \in \{0,1\}^{\ell_0}$

$\Leftrightarrow \tilde{w}_0(\vec{z}) = \tilde{y}(\vec{z}), \forall \vec{z} \in \{0,1\}^{\ell_0}$.

Completeness:

(reduce true claim to true claim and checks pass)

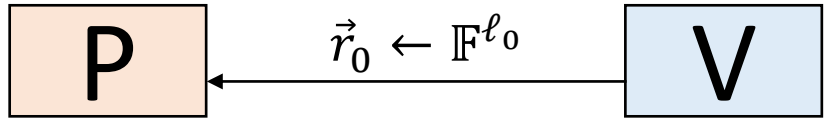
- $\tilde{w}_0, \tilde{y}$ are both MLEs of w_0.
- Hence $\tilde{w}_0 \equiv \tilde{y}$ as polynomials by uniqueness of MLEs.
- They are equal at $\vec{r}_0 \in \mathbb{F}^{\ell_0}$ *outside* $\{0,1\}$ $\Rightarrow \tilde{w}_0(\vec{r}_0) = \tilde{y}(\vec{r}_0) = v_0$.

Soundness: (reduce false claim to false claim, or checks fail, w.h.p.)

- $\deg \tilde{w}_0 = \deg \tilde{y} = \ell_0$.
- If $\tilde{w}_0 \neq \tilde{y}$ then SZ lemma $\Rightarrow \tilde{w}_0(\vec{r}_0) \neq \tilde{y}(\vec{r}_0) = v_0$ except w.p. $\leq \ell_0/|\mathbb{F}|$

w_0 computed from $\vec{x}$

Initial claim: $\tilde{w}_0 \equiv \tilde{y}$



Both compute $v_0 := \tilde{y}(\vec{r}_0)$

Claim: $\tilde{w}_0(\vec{r}_0) = v_0$

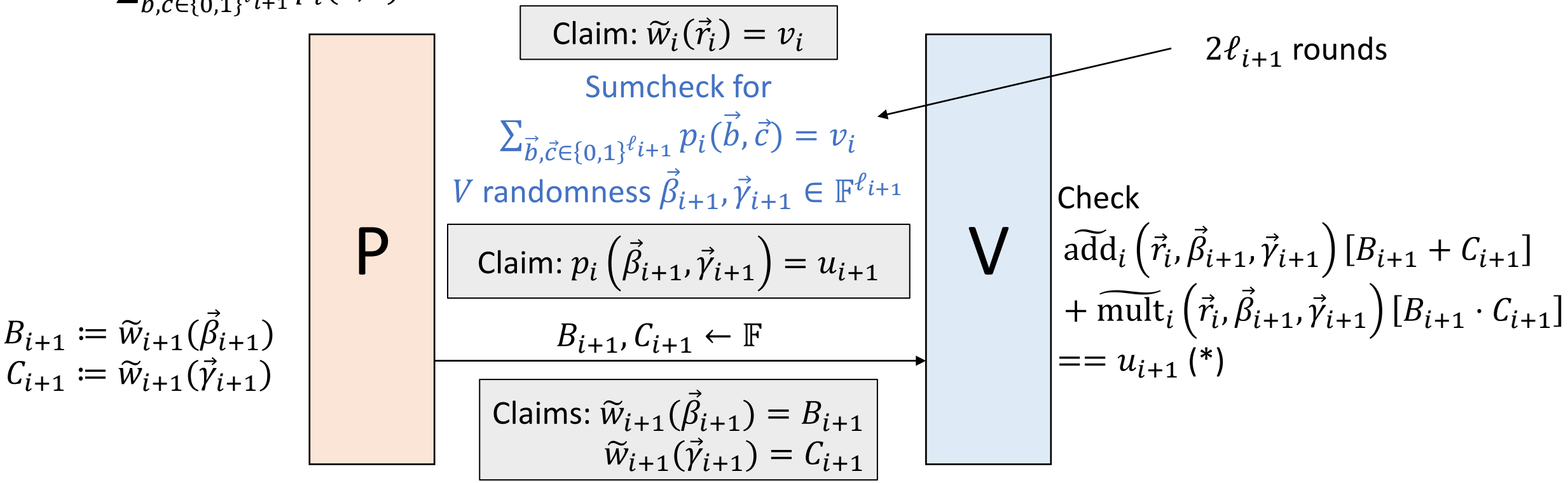
Sumcheck, level i to level i+1

$$p_i\left(\vec{b},\vec{c}\right) := \widetilde{\mathrm{add}}_i\left(\vec{r}_i,\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)+\widetilde{w}_{i+1}(\vec{c})\right] + \widetilde{\mathrm{mult}}_i\left(\vec{r}_i,\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)\cdot\widetilde{w}_{i+1}(\vec{c})\right]$$

Reduction strategy: Evaluate layer sum equation at $\vec{r}_i$

$$\widetilde{w}_i(\vec{r}_i) = \sum_{\vec{b},\vec{c}\in\{0,1\}^{\ell_{i+1}}}\left(\widetilde{\mathrm{add}}_i\left(\vec{r}_i,\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)+\widetilde{w}_{i+1}(\vec{c})\right] + \widetilde{\mathrm{mult}}_i\left(\vec{r}_i,\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)\cdot\widetilde{w}_{i+1}(\vec{c})\right]\right)$$

$$= \sum_{\vec{b},\vec{c}\in\{0,1\}^{\ell_{i+1}}} p_i(\vec{b},\vec{c}).$$

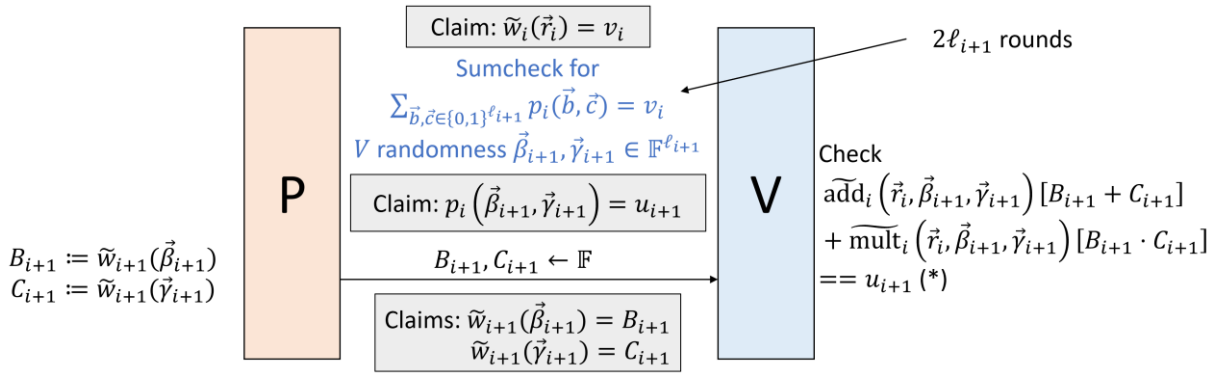


Sumcheck: level i to level i+1, completeness

(reduce true claim to true claims and checks pass): $\forall \vec{z} \in \{0,1\}^{\ell_i}$,

$$\widetilde{w}_i(\vec{z}) = \sum_{\vec{b},\vec{c}\in\{0,1\}^{\ell_{i+1}}} \left(\widetilde{\mathrm{add}}_i\left(\vec{z},\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right) + \widetilde{w}_{i+1}(\vec{c})\right] + \widetilde{\mathrm{mult}}_i\left(\vec{z},\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)\cdot \widetilde{w}_{i+1}(\vec{c})\right]\right)$$

- The L.H.S and R.H.S are *both* MLEs of w_i.
- By uniqueness of MLEs, they are equal as polynomials in $\vec{z}$.
- Therefore we still have equality at $\vec{r}_i \in \mathbb{F}^{\ell_i}$ *outside* $\{0,1\}$.



- $\sum_{\vec{b},\vec{c}\in\{0,1\}^{\ell_{i+1}}} p_i(\vec{b},\vec{c}) = \widetilde{w}_i(\vec{r}_i) = v_i$.
- $p_i\left(\vec{\beta}_{i+1}, \vec{\gamma}_{i+1}\right) = u_{i+1}$ by sumcheck completeness.
- V's check (*) passes by the previous line, and definitions of p_i, B_{i+1} and C_{i+1}.
- $\widetilde{w}_{i+1}(\vec{\beta}_{i+1}) = B_{i+1}$ and $\widetilde{w}_{i+1}(\vec{\gamma}_{i+1}) = C_{i+1}$ by definitions of B_{i+1} and C_{i+1}.

Remember $\widetilde{w}_{i+1}\left(\vec{\beta}_{i+1}\right), \widetilde{w}_{i+1}(\vec{\gamma}_{i+1})$ are defined by the instance and V, not by P

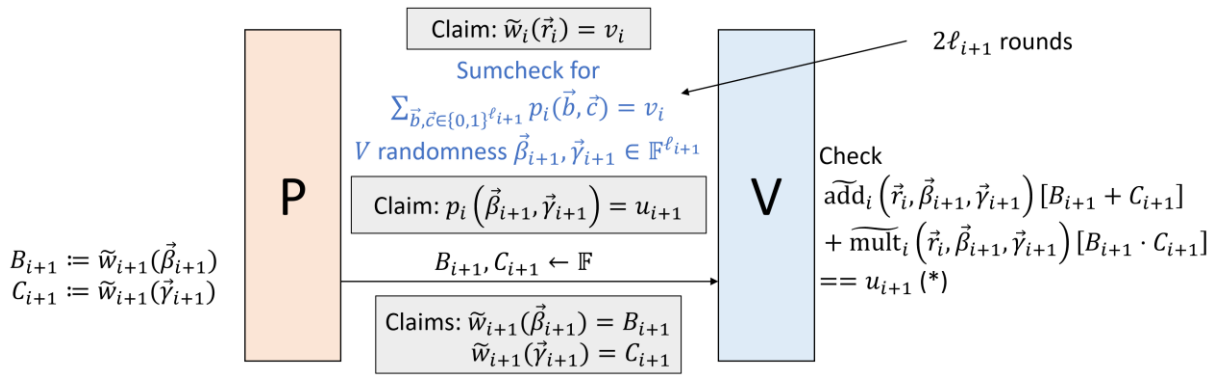
Sumcheck: level i to level i+1, soundness

(reduce false claim to false claims, or checks fail, w.h.p):

- Let $\widetilde{w}_i(\vec{r}_i) \neq v_i$. By the same reasoning as the completeness proof, $\sum_{\vec{b},\vec{c}\in\{0,1\}^{\ell_{i+1}}} p_i(\vec{b},\vec{c}) = \widetilde{w}_i(\vec{r}_i)$ so $\sum_{\vec{b},\vec{c}\in\{0,1\}^{\ell_{i+1}}} p_i(\vec{b},\vec{c}) \neq v_i$.
- By sumcheck soundness, $p_i\left(\vec{\beta}_{i+1},\vec{\gamma}_{i+1}\right) \neq u_{i+1}$ (or V rejects) except w.p. $\leq 4\ell_{i+1}/|\mathbb{F}|$.

If malicious P^* sends the correct $B_{i+1} = \widetilde{w}_{i+1}\left(\vec{\beta}_{i+1}\right), C_{i+1} = \widetilde{w}_{i+1}(\vec{\gamma}_{i+1})$:

- Then $p_i\left(\vec{\beta}_{i+1},\vec{\gamma}_{i+1}\right) = \mathrm{add}_i\left(\vec{r}_i,\vec{\beta}_{i+1},\vec{\gamma}_{i+1}\right)[B_{i+1}+C_{i+1}] + \mathrm{mult}_i\left(\vec{r}_i,\vec{\beta}_{i+1},\vec{\gamma}_{i+1}\right)[B_{i+1}\cdot C_{i+1}] \neq u_{i+1}$.
- Therefore (*) does not hold and V will reject.
- **If P^* sends incorrect B_{i+1}, C_{i+1}:**
- Then at least one final claim is false.



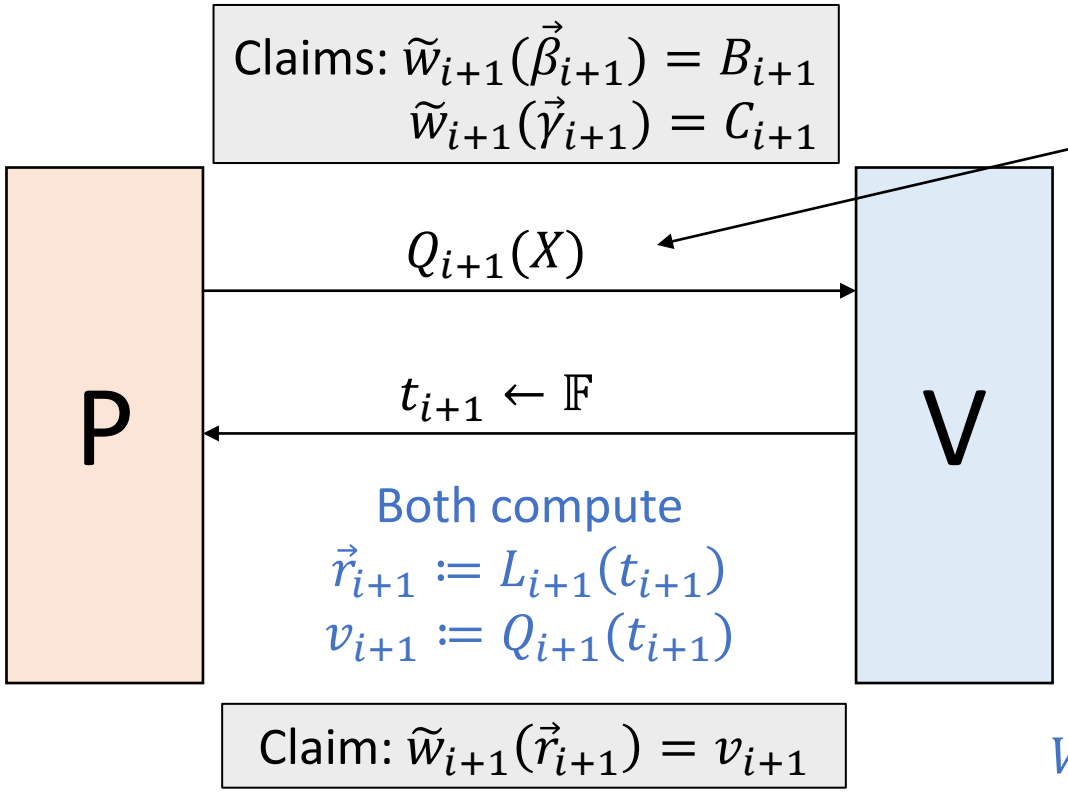
Reducing two claims to one

$L_{i+1} : \mathbb{F} \to \mathbb{F}^{\ell_{i+1}}$

$L_{i+1}(X) \coloneqq (1 - X) \cdot \vec{\beta}_{i+1} + X \cdot \vec{\gamma}_{i+1}$

$Q_{i+1} : \mathbb{F} \to \mathbb{F}$

$Q_{i+1}(X) \coloneqq \widetilde{w}_{i+1}(L_{i+1}(X))$



degree ℓ_{i+1}

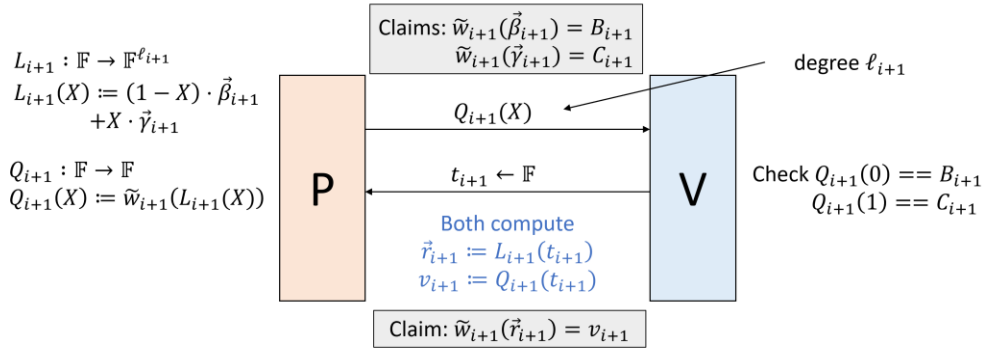
Check $Q_{i+1}(0) == B_{i+1}$
$Q_{i+1}(1) == C_{i+1}$

V computes v_{i+1} from Q_{i+1}
V computes L_{i+1} for themself

Reducing two claims to one, completeness

(reduce true claims to true claim

and checks pass)



V's checks will pass:

- Suppose $\widetilde{w}_{i+1}(\vec{\beta}_{i+1}) = B_{i+1}$ and $\widetilde{w}_{i+1}(\vec{\gamma}_{i+1}) = C_{i+1}$.
- $Q_{i+1}(0) = \widetilde{w}_{i+1}\big(L_{i+1}(0)\big) = \widetilde{w}_{i+1}\left((\vec{\beta}_{i+1})\right) = B_{i+1}$.
- $Q_{i+1}(1) = \widetilde{w}_{i+1}\big(L_{i+1}(1)\big) = \widetilde{w}_{i+1}\big((\vec{\gamma}_{i+1})\big) = C_{i+1}$.

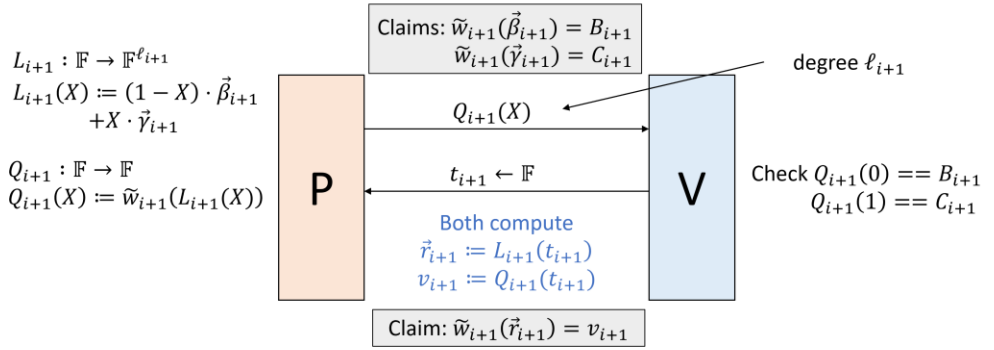
Reduce to a true claim:

- $v_{i+1} := Q_{i+1}(t_{i+1}) = \widetilde{w}_{i+1}\big(L_{i+1}(t_{i+1})\big) = \widetilde{w}_{i+1}(\vec{r}_{i+1})$.

Reducing two claims to one, soundness

(reduce false claims to false claim, or checks fail, w.h.p):

- Suppose $\widetilde{w}_{i+1}(\vec{\beta}_{i+1}) \neq B_{i+1}$ or $\widetilde{w}_{i+1}(\vec{\gamma}_{i+1}) \neq C_{i+1}$.



If malicious P^* sends the correct $Q_{i+1}(X) = \widetilde{w}_{i+1}(L_{i+1}(X))$:

- Then either $Q_{i+1}(0) = \widetilde{w}_{i+1}(L_{i+1}(0)) = \widetilde{w}_{i+1}(\vec{\beta}_{i+1}) \neq B_{i+1}$, or similarly $Q_{i+1}(1) \neq C_{i+1}$.
- V will reject.

Check $Q_{i+1}(0) == B_{i+1}$
$Q_{i+1}(1) == C_{i+1}$

If malicious P^* sends incorrect $Q^*_{i+1}(X) \neq Q_{i+1}(X)$:

Remember $\widetilde{w}_{i+1}(\vec{r}_{i+1})$ is defined by the instance and V

- Then $Q^*_{i+1}(t_{i+1}) \neq Q_{i+1}(t_{i+1})$ except w.p. $\leq \ell_{i+1}/|\mathbb{F}|$ by the Schwartz-Zippel Lemma.
- V will compute $v_{i+1} = Q^*_{i+1}(t_{i+1})$ using Q^*_{i+1} from malicious P^*.
- $v_{i+1} = Q^*_{i+1}(t_{i+1}) \neq Q_{i+1}(t_{i+1}) = \widetilde{w}_{i+1}\big(L_{i+1}(t_{i+1})\big) = \widetilde{w}_{i+1}(\vec{r}_{i+1})$.
- We have reduced to a false claim.

V computes L_{i+1} for themself

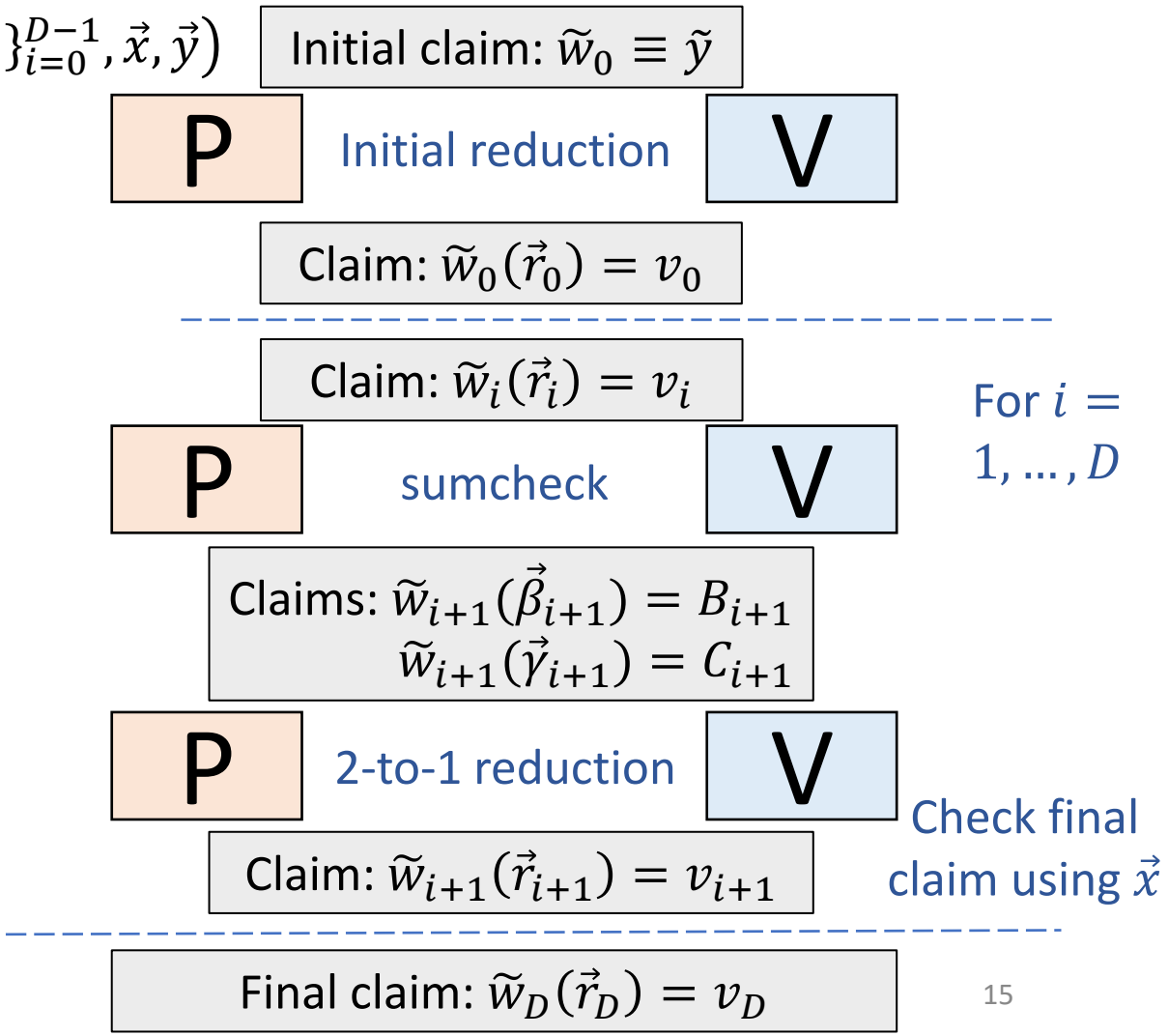
Security of the full GKR protocol

Language:
- satisfies $C(\vec{x}) = \vec{y}$

Instance:
- $\left(\{\text{add}_i, \text{mult}_i\}_{i=0}^{D-1}, \vec{x}, \vec{y}\right)$

Completeness:
- Each reduction maps true claims to true claims and V accepts.
- Final claim is true and V accepts overall.

Soundness:
- If $C(\vec{x}) \neq \vec{y}$ then $\widetilde{w}_0 \neq \tilde{y}$.
- W.h.p. each reduction maps false claims to false, or makes V reject



Parameters of the full GKR protocol

$2^{\ell_i} = S_i \leq S$ $\quad\quad$ $\sum \ell_i \leq O(D \log S)$

Reduction	d for sent polynomials	#variables	#rounds	soundness error	communication	V complexity
Initial	–	–	$O(1)$	$\ell_0/\lvert\mathbb{F}\rvert$	$O(\ell_0)$	eval $\tilde{y}$ ($\vec{y} \in \mathbb{F}^m$)
Sumcheck	$O(1)$	$O(\ell_{i+1})$	$O(\ell_{i+1})$	$4\ell_{i+1}/\lvert\mathbb{F}\rvert$	$O(\ell_{i+1})$	$O(\ell_{i+1})$ (sumcheck) eval $\widetilde{\text{add}}_i, \widetilde{\text{mult}}_i$
2-to-1	$O(\ell_{i+1})$	$O(1)$	$O(1)$	$\ell_{i+1}/\lvert\mathbb{F}\rvert$	$O(\ell_{i+1})$	$O(\ell_{i+1})$ (eval Q_{i+1}) $O(\ell_{i+1})$ (find, eval L_{i+1})
Final check	–	–	–	–	–	eval $\tilde{x}$ ($\vec{x} \in \mathbb{F}^n$)

For $i = 1, \ldots, D$

Totals: $O(D \log S)$ $\quad$ $O(D \log S)$ $\mathbb{F}$-elements $\quad$ $O(m + n + D \log S)$ $\mathbb{F}$-ops + evaluation of $\{\text{add}_i, \text{mult}_i\}_{i=0}^{D-1}$

$O(D \log S / \lvert\mathbb{F}\rvert)$

$O(m + n + D \log S)$ $\mathbb{F}$-ops for suitable circuits

GKR prover complexity

- Dominated by the costs of sumchecks on $H = \{0,1\}$ for

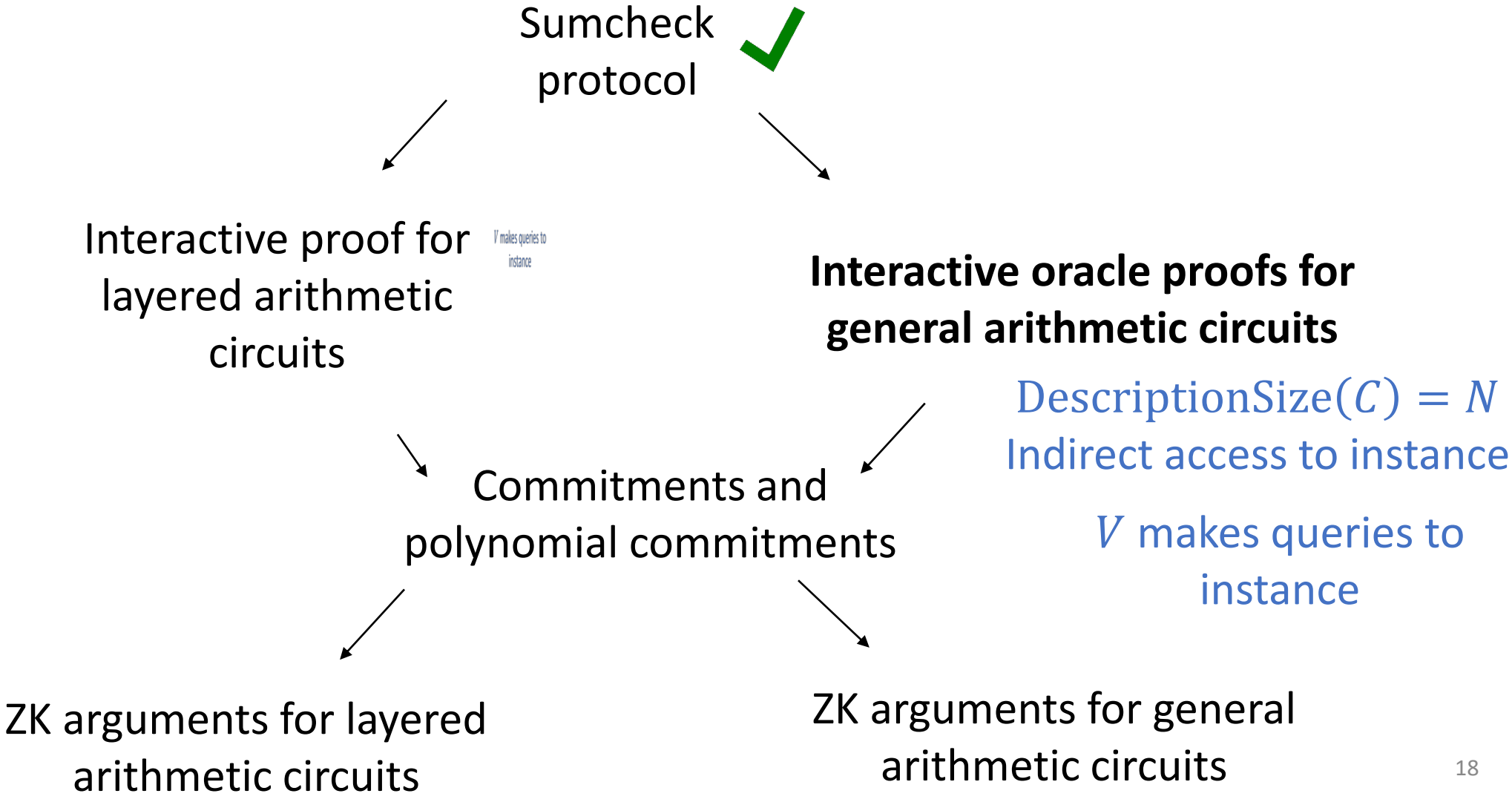
$$p_i\left(\vec{b},\vec{c}\right) \coloneqq \widetilde{\mathrm{add}}_i\left(\vec{r}_i,\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)+\widetilde{w}_{i+1}(\vec{c})\right]+\widetilde{\mathrm{mult}}_i\left(\vec{r}_i,\vec{b},\vec{c}\right)\left[\widetilde{w}_{i+1}\left(\vec{b}\right)\cdot\widetilde{w}_{i+1}(\vec{c})\right]$$

- This time $d+1=4>2=|H|$ so previous algorithm does not work.
- Naïve cost estimate:

$$\widetilde{\mathrm{add}}_i \cdot [\widetilde{w}_{i+1}+\widetilde{w}_{i+1}] + \widetilde{\mathrm{mult}}_i \cdot [\widetilde{w}_{i+1}\cdot\widetilde{w}_{i+1}]$$

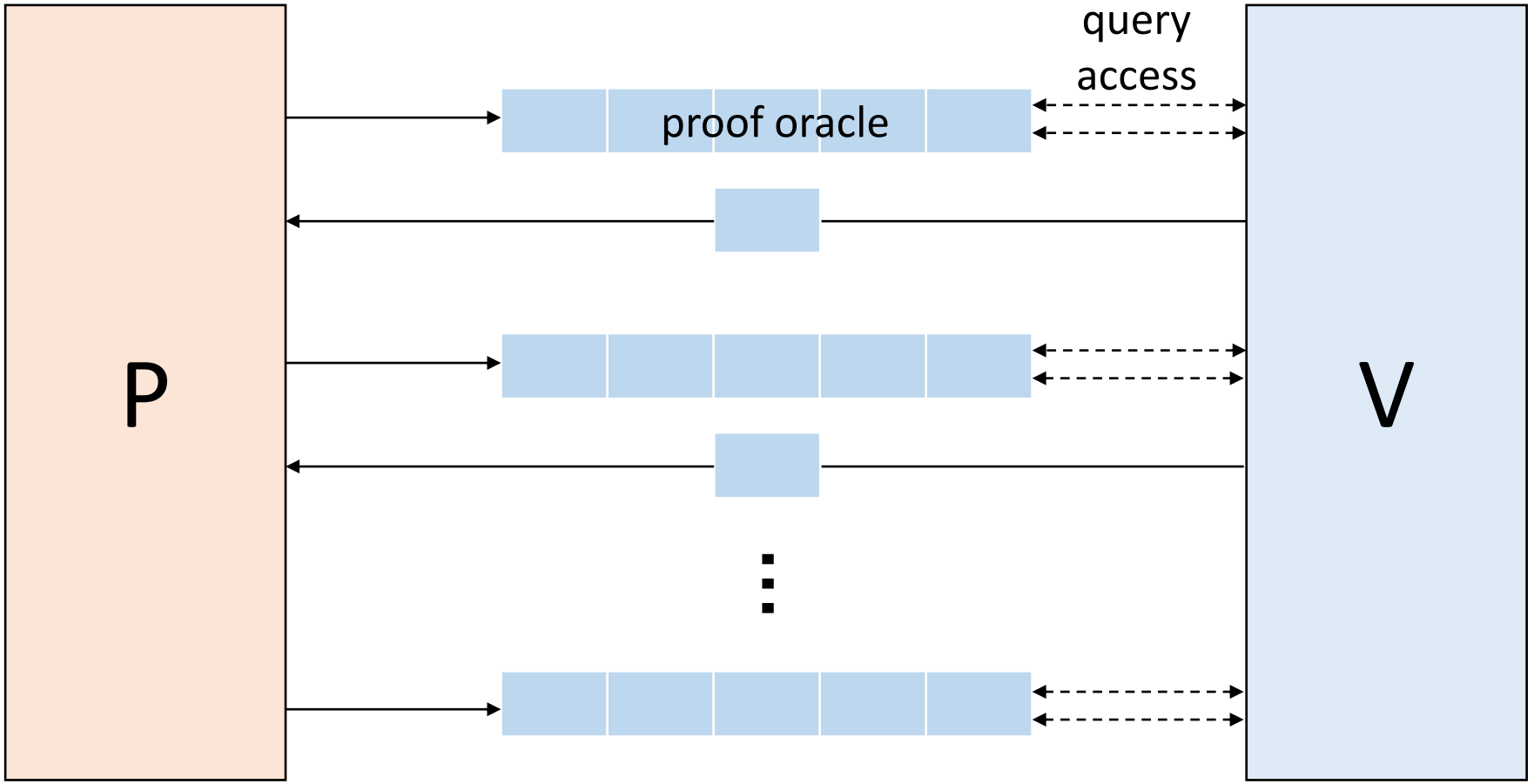
$\ell_i + 2\ell_{i+1}$ variables	ℓ_{i+1} variables	$\ell_i + 2\ell_{i+1}$ variables	$D \cdot poly(S)$ prover time
$2^{\ell_i+2\ell_{i+1}} = poly(S)$ coefficients	$poly(S)$ coefficients	$poly(S)$ coefficients	Can improve this a lot using ‘table method’-later

Plan for the next few lectures



Interactive oracle proofs

Note: query answers are always computed correctly based on the oracle!



Point queries:
$query(\pi, i) = \pi(i)$

Linear queries:
$query(\vec{\pi}, \vec{q}) = \langle \vec{\pi}, \vec{q} \rangle$

Univariate polynomial evaluation queries:
$query(\pi, q) = \pi(q)$

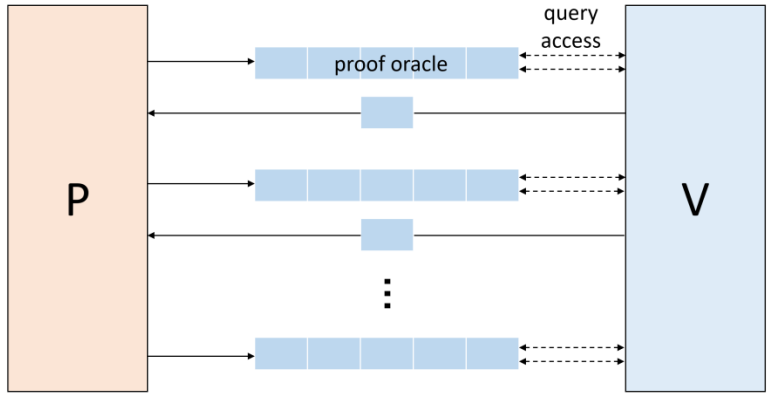
Multivariate polynomial evaluation queries:
$query(\pi, \vec{q}) = \pi(\vec{q})$

proof oracle = committed data

answering query = opening commitment

IOPs, IPs and PCPs

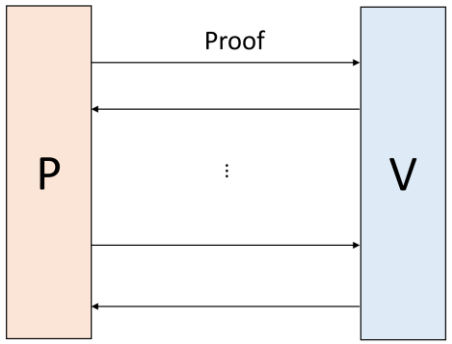
IOPs



IOP,PCP=NEXP

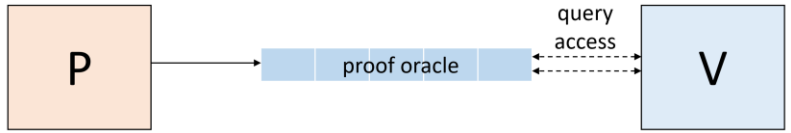
PCP theorem: for any **NP** language, there are PCPs with polynomial proof length, $O(1)$ queries, $\Theta(1)$ soundness error

IPs



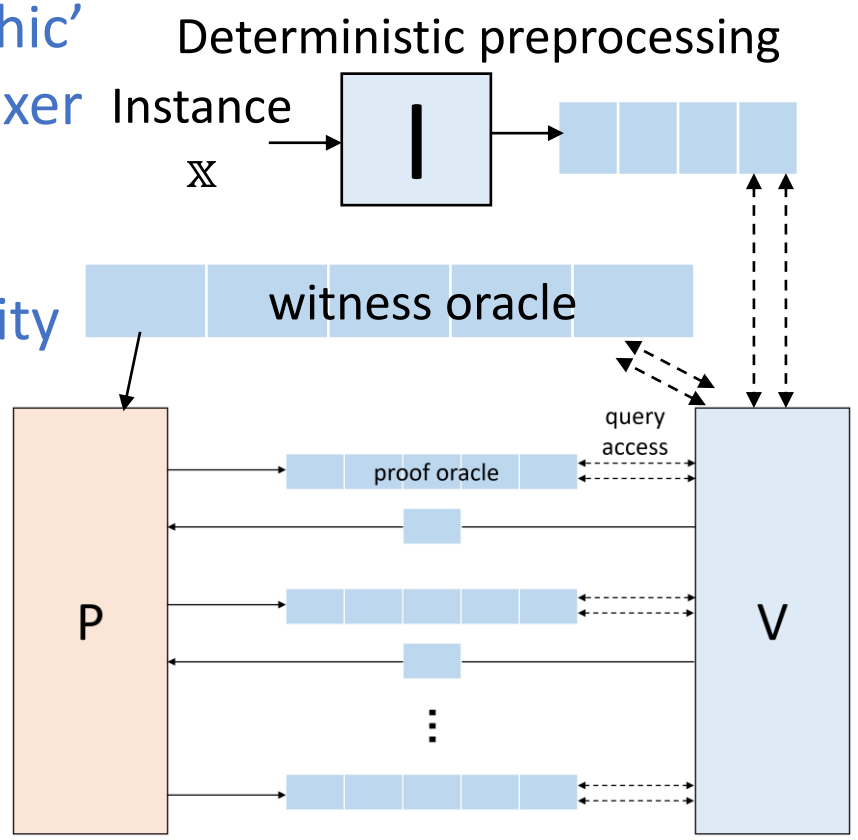
IP=PSPACE

PCPs
Probabilistically checkable proofs



IOP parameters and variants

‘holographic’
With indexer



IOPs of proximity (IOPP)

- $a_1, st_{P,1} = P(x, y, r)$
- $a_2, st_{V,2} = V^{a_1}(x, z, s)$
- $a_3, st_{P,3} = P\left(x, y, st_{P,1}, a_1, a_2, r\right)$

  $\vdots$
- $a_{k(x)}, st_{V,k(x)} = V^{a_1,\ldots,a_{k(x)-1}}\left(x, z, st_{V,k(x)-2}, s\right)$

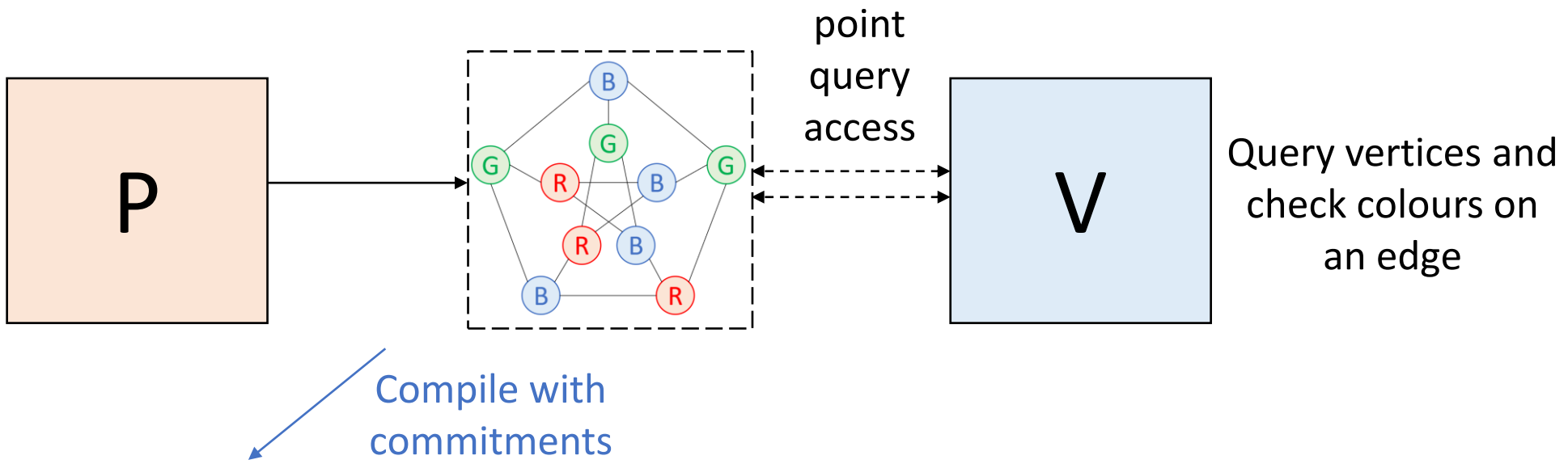
Allow some ‘normal messages’ too

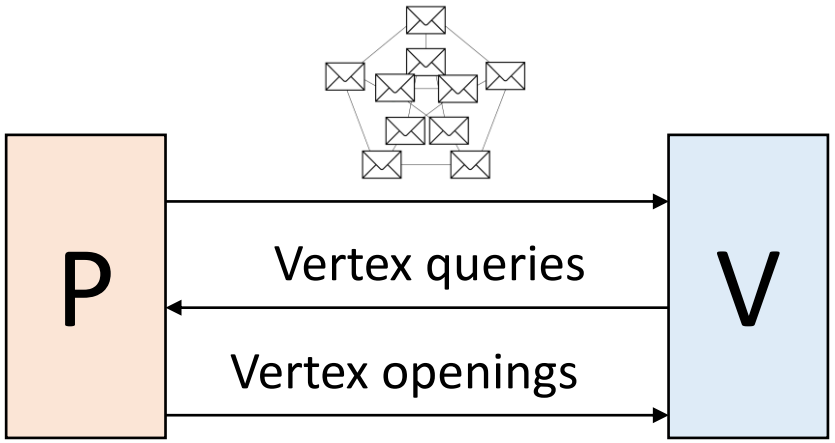
Security:

- Completeness and soundness definitions similar to IPs.
- Zero-knowledge is more complicated to define (not used in the course).

Efficiency: prover and verifier complexity, proof length, query complexity, communication complexity

G3C protocol as an IOP (PCP)





Notes:

- Not a proof of the PCP theorem.
- MPC-in-the-Head is also a PCP.
- Compile with any commitments allowing point openings
- Individual commitments to vertex colours.
- Merkle roots and paths.
- In general, compilers for other query types use commitments with matching functionality.

Rank 1 Constraint Systems (R1CS)

$$\mathcal{R}_{R1CS} = \left\{ \left((\mathbb{F}, A, B, C, \vec{x}) , \vec{w} \right) : \begin{array}{c} A, B, C \in \mathbb{F}^{N_r \times N_c}, \vec{x} \in \mathbb{F}^k \\ \vec{w} \in \mathbb{F}^{N_c - k}, \vec{z} \coloneqq \vec{x} || \vec{w} \\ A\vec{z} \circ B\vec{z} = C\vec{z} \end{array} \right\}.$$

$\vec{x}$ makes the problem non-trivial

entry-wise product

- **NP**-complete. Exercise: reduce SAT to R1CS.
- N gates $\Rightarrow O(N) \times O(N)$ matrices with $O(N)$ non-zero entries.

Multivariate, $\widetilde{EQ}$-basis. Based on Spartan paper

- We will construct a *holographic* IOP with *polynomial* queries for $\mathcal{R}_{R1CS}$.
- Verifier complexity $O(\ell + N_{in}) = O(\log N + N_{in})$ $\mathbb{F}$-ops.
- Query complexity $O(1)$ (to $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{z}$, other message vectors).
- Prover complexity $O(N + |A| + |B| + |C|)$ $\mathbb{F}$-ops.

ETH Zurich, Department of Computer Science — Dr. Jonathan Bootle
HS 2023 — Karen Klein, Varun Maram, Antonio Merino Gallardo

Zero-Knowledge Proofs
Exercise 8

8.1 IP $\subseteq$ PSPACE

The complexity class PSPACE is defined[1] as a set of languages L for which there exists an algorithm M and a polynomial q such that $\forall x \in \{0,1\}^*$,

- $M(x) = 1 \iff x \in L$, and
- M uses $q(|x|)$ memory on input x (but may have *unbounded* run-time).

Note that the class P requires M to run in polynomial time, whereas PSPACE only requires M to use a polynomial amount of memory.

Show that IP $\subseteq$ PSPACE, where the class IP considers provers and verifiers with *no* auxiliary inputs (as defined earlier on in the lectures).

Hint: Writing out a single execution of an IP protocol requires a polynomial amount of space.

8.2 Efficiency of the IP for UNSAT

What is the prover efficiency of the interactive proof for UNSAT given in the lectures?

8.3 Sumcheck with Error-Correcting Codes ($\star$)

A linear error-correcting code $\mathcal{C}$ over a field $\mathbb{F}$ (with dimension k) is a linear subspace of $\mathbb{F}^N$ generated by the rows of a so-called "generator" matrix $M \in \mathbb{F}^{k \times N}$.[2] Elements of $\mathcal{C}$ are called codewords. The minimum distance d of the code $\mathcal{C}$ is defined as follows:

$$d = \min_{c,c' \in \mathcal{C} \setminus \{0^N\}, c \neq c'} \left| \{ j \in [\![N]\!] \mid c_j \neq c'_j \} \right|,$$

where $c = (c_1, \ldots, c_N) \in \mathbb{F}^N$ and $c' = (c'_1, \ldots, c'_N) \in \mathbb{F}^N$.

Let p be a function $p\colon [\![N]\!]^l \to \mathbb{F}$, such that whenever $l-1$ inputs to p are fixed, p evaluates on $[\![N]\!]$ to a codeword in $\mathcal{C}$. More formally, for any $j \in [l]$, and any points $\omega_1, \ldots, \omega_{j-1}, \omega_{j+1}, \ldots, \omega_l \in [\![N]\!]$, we have that $(p(\omega_1, \ldots, \omega_{j-1}, i_j, \omega_{j+1}, \ldots, \omega_l))_{i_j \in [\![N]\!]} \in \mathcal{C}$.
Let $H \subseteq [\![N]\!]$. Give an interactive protocol allowing the verifier to check that

$$\sum_{\omega_1, \ldots, \omega_l \in H} p(\omega_1, \ldots, \omega_l) = v$$

using only a single evaluation of p and prove that it is complete. Give an upper bound on the soundness error of the protocol.

---

[1] See https://en.wikipedia.org/wiki/PSPACE, https://complexityzoo.net/Complexity_Zoo:P#pspace.

[2] More formally, if we denote $M_i \in \mathbb{F}^N$ to be the i-th row of M, then we have for *any* $c \in \mathcal{C}$, there exist field elements $(a_1, \ldots, a_k) \in \mathbb{F}^k$ such that $c = \sum_{i=1}^k a_i \cdot M_i$ where "$a_i \cdot M_i$" is a component-wise multiplication of a field element with a vector of elements.

ETH Zurich, Department of Computer Science — Dr. Jonathan Bootle
HS 2023 — Karen Klein, Varun Maram, Antonio Merino Gallardo

Zero-Knowledge Proofs
Exercise 8

8.1 IP $\subseteq$ PSPACE

The complexity class PSPACE is defined[1] as a set of languages L for which there exists an algorithm M and a polynomial q such that $\forall x \in \{0,1\}^*$,

- $M(x) = 1 \iff x \in L$, and
- M uses $q(|x|)$ memory on input x (but may have *unbounded* run-time).

Note that the class P requires M to run in polynomial time, whereas PSPACE only requires M to use a polynomial amount of memory.

Show that IP $\subseteq$ PSPACE, where the class IP considers provers and verifiers with *no* auxiliary inputs (as defined earlier on in the lectures).

HINT: Writing out a single execution of an IP protocol requires a polynomial amount of space.

Solution: Let L be a language in IP and consider a protocol (P, V) for L. Without loss of generality we assume (P, V) to be public-coin. To prove that L is in PSPACE, it suffices to show that on any input x, the maximum probability with which a prover can make V accept is computable in polynomial space. That is because a decider can then compute that probability and accept if it is at least $3/4$ and reject if it is at most $1/2$ (the completeness and soundness of (P, V) ensure that only those two cases are possible).

Consider a rooted tree in which each node at depth i corresponds to a sequence of i messages exchanged between the prover and the verifier. This tree has polynomial depth and each node has $2^{\text{poly}(|x|)}$ children since V runs in polynomial time and the total length of the messages in the protocol must therefore be polynomial. Each node is now assigned a value as follows.

- A leaf node is assigned the value 1 if the verifier accepts and 0 if the verifier rejects.
- A node at a depth where the prover sends the next message is assigned the *maximum* value of its children nodes.
- A node at a depth where the verifier sends the next message is assigned the *average* of the values of its children nodes.

The value at the root determines the maximum probability with which a prover can make the verifier accept on input x, and this value can be computed in polynomial space (via a depth-first search) since the tree has polynomial depth and $2^{\text{poly}(|x|)}$ arity.

[1] See https://en.wikipedia.org/wiki/PSPACE, https://complexityzoo.net/Complexity_Zoo:P#pspace.

8.2 Efficiency of the IP for UNSAT

What is the prover efficiency of the interactive proof for UNSAT given in the lectures?

Solution: Arithmetising a boolean formula in ℓ variables with m clauses results in an ℓ-variate polynomial p with individual degree at most $3m$.

The bound from the lectures on the complexity of the prover of the sum-check protocol does not apply as it assumed the prover to be given the coefficients of the polynomial as input, whereas the prover is in the present case only given the circuit. Furthermore, the analysis in the lectures only considered the case $d + 1 \leq |H|$, whereas here we have $d = 3m > |H| = |\{0, 1\}| = 2$.

In the i-th round of the sum-check protocol for UNSAT, the prover is supposed to send a univariate polynomial $\sum_{a_{i+1},\ldots,a_\ell \in \{0,1\}} p(r_1, \ldots, r_{i-1}, X_i, a_{i+1}, \ldots, a_\ell) \in \mathbb{F}[X_i]$. The prover can instead send its evaluation at $3m + 1$ interpolation points fixed at the outset of the protocol (and the verifier can reconstruct the polynomial via Lagrange's interpolation). Evaluating each clause of the 3-CNF requires at-most 4 additions and 2 multiplications over the field. Each evaluation of $p(r_1, \ldots, r_{i-1}, X, a_{i+1}, \ldots, a_\ell)$ at a given point thus requires $6m + (m - 1)$ field operations, and evaluating the sum $O\left(2^{\ell-i}m\right)$ operations. Therefore, the overall prover complexity is of order $O(2^\ell m^2)$ since the prover must evaluate each polynomial at $3m + 1$ pairwise distinct points and the number of rounds is $\ell < 2^\ell$.

8.3 Sumcheck with Error-Correcting Codes ($\star$)

A linear error-correcting code $\mathcal{C}$ over a field $\mathbb{F}$ (with dimension k) is a linear subspace of $\mathbb{F}^N$ generated by the rows of a so-called "generator" matrix $M \in \mathbb{F}^{k \times N}$.[2] Elements of $\mathcal{C}$ are called codewords. The minimum distance d of the code $\mathcal{C}$ is defined as follows:

$$d = \min_{c,c' \in \mathcal{C} \setminus \{0^N\}, c \neq c'} \left| \{j \in [\![N]\!] \mid c_j \neq c'_j\} \right|,$$

where $c = (c_1, \ldots, c_N) \in \mathbb{F}^N$ and $c' = (c'_1, \ldots, c'_N) \in \mathbb{F}^N$.

Let p be a function $p\colon [\![N]\!]^l \to \mathbb{F}$, such that whenever $l - 1$ inputs to p are fixed, p evaluates on $[\![N]\!]$ to a codeword in $\mathcal{C}$. More formally, for any $j \in [l]$, and any points $\omega_1, \ldots, \omega_{j-1}, \omega_{j+1}, \ldots, \omega_l \in [\![N]\!]$, we have that $(p(\omega_1, \ldots, \omega_{j-1}, i_j, \omega_{j+1}, \ldots, \omega_l))_{i_j \in [\![N]\!]} \in \mathcal{C}$.

Let $H \subseteq [\![N]\!]$. Give an interactive protocol allowing the verifier to check that

$$\sum_{\omega_1,\ldots,\omega_l \in H} p(\omega_1, \ldots, \omega_l) = v$$

using only a single evaluation of p and prove that it is complete. Give an upper bound on the soundness error of the protocol.

Solution: The interactive protocol proceeds in l rounds as follows: in the i-th round, the prover sends a codeword of evaluations $(q_i(1), q_i(2), \ldots, q_i(N)) \in \mathcal{C}$ where $q_i(X_i) = \sum_{\omega_{i+1},\ldots,\omega_l \in H} p(r_1, \ldots, r_{i-1}, X_i, \omega_{i+1}, \ldots, \omega_l)$ with $r_1, \ldots, r_{i-1} \leftarrow_\$ [\![N]\!]$ being the random values chosen by the verifier (and sent to the prover) in each of the previous $i - 1$ rounds. The verifier first checks if the received vector of evaluations $(q_i(1), q_i(2), \ldots q_i(N)) \in \mathbb{F}^N$ is indeed a codeword in $\mathcal{C}$ (this can be done efficiently using the generator matrix M), and then checks if $\sum_{w_i \in H} q_i(\omega_i) = q_{i-1}(r_{i-1})$ (we have "$q_{i-1}(r_{i-1})$" to be v when $i = 1$); note that the verifier "looks up" the values $q_i(\omega_i)$ each time from the received codeword of evaluations. If both checks pass, the verifier sends a random value $r_i \leftarrow_\$ [\![N]\!]$; otherwise,

[2] More formally, if we denote $M_i \in \mathbb{F}^N$ to be the i-th row of M, then we have for *any* $c \in \mathcal{C}$, there exist field elements $(a_1, \ldots, a_k) \in \mathbb{F}^k$ such that $c = \sum_{i=1}^k a_i \cdot M_i$ where "$a_i \cdot M_i$" is a component-wise multiplication of a field element with a vector of elements.

it terminates with a "rejection" output 0. After the l rounds, the verifier does a final check: namely, whether $q_l(r_l) = p(r_1, r_2, \ldots, r_l)$; this is the only time when the verifier does an evaluation of p in the entire protocol. If the check passes, the verifier "accepts" the proof with an output 1, and otherwise "rejects" with an output 0.

To prove the completeness of this protocol, we only need to show that the "honestly" generated vector of evaluations $(q_i(1), q_i(2), \ldots q_i(N)) \in \mathbb{F}^N$ in any i-th round is a codeword in $\mathcal{C}$; the rest of the completeness analysis follows quite closely to that seen in the lectures with respect to sumcheck protocol with polynomials. Let's fix the random values $(r_1, \ldots, r_{i-1})$ and the values in H $(\omega_{i+1}, \ldots, \omega_l)$. Note that we have the vector $(p(r_1, \ldots, r_{i-1}, X_i, \omega_{i+1}, \ldots, \omega_l))_{X_i \in [\![N]\!]}$ to be a codeword in $\mathcal{C}$ by definition of the function p above. Since $\mathcal{C}$ is also a linear subspace of $\mathbb{F}^N$, it follows that a summation of these codewords results in a valid codeword in $\mathcal{C}$. Specifically, we have

$$\begin{aligned}
&\sum_{\omega_{i+1}, \ldots, \omega_l \in H} (p(r_1, \ldots, r_{i-1}, X_i, \omega_{i+1}, \ldots, \omega_l))_{X_i \in [\![N]\!]} \\
&= \Big(\sum_{\omega_{i+1}, \ldots, \omega_l \in H} p(r_1, \ldots, r_{i-1}, X_i, \omega_{i+1}, \ldots, \omega_l) \Big)_{X_i \in [\![N]\!]} \\
&= (q_i(X_i))_{X_i \in [\![N]\!]} \;=\; (q_i(1), q_i(2), \ldots, q_i(N)) \;\in\; \mathcal{C}.
\end{aligned}$$

To prove soundness of our protocol, we closely follow the soundness analysis of the sumcheck protocol with polynomials seen in the lectures. Namely, suppose $\sum_{\omega_1, \ldots, \omega_l \in H} p(\omega_1, \ldots, \omega_l) \neq v$. Let's consider the message $(\tilde{q}_1(1), \tilde{q}_1(2) \ldots, \tilde{q}_1(N))$ sent by a (potentially) malicious prover in the 1st round, for an arbitrary function $\tilde{q} : [\![N]\!] \to \mathbb{F}$. Let bad be the event when the following two conditions hold:

1. We reduce to a true statement in the 2nd round. That is,

$$\sum_{\omega_2, \ldots, \omega_l \in H} p'(\omega_2, \ldots, \omega_l) = v'$$

   where $p'(\omega_2, \ldots, \omega_l) = p(r_1, \omega_2, \ldots, \omega_l)$ with $r_1 \leftarrow_{\$} [\![N]\!]$ being the random value sent by the verifier in the 1st round, and $v' = \tilde{q}_1(r_1)$.

2. The verifier's checks pass in the 1st round. That is, the vector $(\tilde{q}_1(1), \tilde{q}_1(2) \ldots, \tilde{q}_1(N))$ is a codeword in $\mathcal{C}$, *and* we have $\sum_{w_1 \in H} \tilde{q}_1(\omega_1) = v$.

Now let $(q_1(1), q_1(2) \ldots, q_1(N))$ be the message that should have been sent by an honest prover in the 1st round, where $q_1(X_1) = \sum_{\omega_2, \ldots, \omega_l \in H} p(X_1, \omega_2, \ldots, \omega_l)$. We now consider two cases:

1. We have the malicious prover's 1st round message and the honestly generated message to be the same. That is, $(\tilde{q}_1(1), \ldots, \tilde{q}_1(N)) = (q_1(1), \ldots, q_1(N))$. In this case, we have $\sum_{\omega_1 \in H} \tilde{q}_1(\omega_1) = \sum_{\omega_1 \in H} q_1(\omega_1) = \sum_{\omega_1, \ldots, \omega_l \in H} p(\omega_1, \ldots, \omega_l) \neq v$. Hence, this violates the second condition of the bad event above, and we have $\Pr[\mathsf{bad}] = 0$ in this case.

2. We have $(\tilde{q}_1(1), \ldots, \tilde{q}_1(N)) \neq (q_1(1), \ldots, q_1(N))$. In this case, suppose we also condition on $(\tilde{q}_1(1), \ldots, \tilde{q}_1(N))$ being a valid codeword in $\mathcal{C}$ – as required by the second condition of event bad. Then from the minimum distance d of code $\mathcal{C}$, we have

$$\big|\, \{r_1 \in [\![N]\!] \mid \tilde{q}_1(r_1) = q_1(r_1)\} \,\big| \;\leq\; N - d.$$

   And note that for the random value $r_1 \leftarrow_{\$} [\![N]\!]$ chosen by the verifier, we have $\tilde{q}_1(r_1) = v'$ and $q_1(r_1) = \sum_{\omega_2, \ldots, \omega_l \in H} p(r_1, \omega_2, \ldots, \omega_l) = \sum_{\omega_2, \ldots, \omega_l \in H} p'(\omega_2, \ldots, \omega_l)$.

Hence, with probability at-most $\frac{N-d}{N}$, the verifier choses an $r_1 \in [\![N]\!]$ such that $\tilde{q}_1(r_1) = q_1(r_1)$, or, $\sum_{\omega_2,\ldots,\omega_l \in H} p'(\omega_2, \ldots, \omega_l) = v'$. In other words, we bounded the probability of the first condition of event bad being satisfied – *conditional* on $(\tilde{q}_1(1), \ldots, \tilde{q}_1(N))$ being a codeword in $\mathcal{C}$ (i.e., partial fulfillment of the second condition) – by $\frac{N-d}{N}$ from above. Hence, it's not hard to see that in this case, we have $\Pr[\mathsf{bad}] \leq \frac{N-d}{N}$

The rest of our soundness analysis follows quite closely to that seen in the lectures with respect to the sumcheck protocol with polynomials. Namely, using induction, it's not hard to obtain that the verifier accepts the proof from a malicious prover with probability at-most $\frac{l(N-d)}{N}$. However, note that for linear error-correcting codes where the minimum distance d is a constant fraction of the code length N (e.g., $d = N/4$), our soundness bound may not give a meaningful guarantee, because in such cases we may end up with a trivial bound $1 < \frac{l(N-d)}{N}$. Hence, it's better to use the tighter soundness bound seen in the lectures for sumcheck protocol with polynomials, which in our case leads to a soundness bound $1 - (\frac{d}{N})^l$.