SWIRL: STACKED WHIR WITH INTERACTION REDUCTIONS VIA LOGUP

OPENVM CONTRIBUTORS

Abstract. We present SWIRL, a high-performance STARK designed to bridge the gap between expressive circuit design and efficient proving. As zero-knowledge Virtual Machines (zkVMs) grow in complexity, they increasingly rely on heterogeneous sub-components that defy uniform constraint structures. SWIRL addresses this by using modular applications of sumcheck to interoperate between computations over polynomials on multiple domains. Notably, each phase of the protocol uses the domain most suitable for performance. This involves use of piecewise hypercube stacking to go between uniform phases such as GKR [GKR15] and WHIR [ACFY24] and heterogeneous phases such as ZeroCheck. We further generalize the "univariate skip" technique [Gru24] to offer tunable optimizations for more general sumcheck protocols. Instantiated with the WHIR polynomial commitment scheme, SWIRL yields a SNARK with fast verification, compact proofs, and provable post-quantum security.

Contents

Date: March 13, 2026.
Contact: info@openvm.dev.

1. Introduction

Modern approaches to verifiable computation necessitate complex arithmetic circuits with heterogeneous sub-components. These circuits, exemplified by zero-knowledge virtual machines (zkVMs), feature intricate internal logic within components alongside varied intercommunications between them. Simultaneously, practical deployment demands fast verification times and small proof sizes to support performant recursive proof aggregation and low-latency data transfer.

In this work we introduce SWIRL, a new STARK proof system built to address the friction between expressive circuit design and efficient proving. Our architecture focuses on three key advancements:

- **Supporting heterogeneous circuits without performance penalties:** Many proof systems accomodate heterogeneous circuit arithmetizations only by incurring either significant penalties: the prover often computes on extraneous or "padded" data to meet rigid layout requirements, or the verifier suffers from excessive polynomial openings. We solve this by effectively *stacking* multivariate polynomials over different domains together prior to invoking any polynomial commitment scheme. Crucially, this stacking is efficiently verifiable due to special properties of boolean hypercubes.
- **Fast verification for efficient recursion:** Our system prioritizes low verification latency and compact proof sizes, which also leads to highly performant recursive proof aggregation. We utilize WHIR [ACFY24] as the polynomial commitment scheme, leveraging its verifier efficiency to minimize cryptographic overhead.
- **Sumcheck-based optimizations:** We extensively utilize sumcheck-based methods to reduce prover overhead and the number of cryptographic commitments. We employ a customized LogUp-GKR protocol to prove arbitrary multiset equality constraints without additional witness commitments. Furthermore, we generalize the univariate skip technique of [Gru24] to apply it to sumcheck-based protocols beyond ZeroCheck.

Our main result is:

Theorem 1.0.1 (informal, see Corollary 4.2.6)**.** *SWIRL is a SNARK in the quantum random oracle model, with adaptive soundness and adaptive knowledge soundness error given by* (4.9) *and* (4.10)*.*

The rest of the paper is organized as follows. In §2 we review the circuit frontend arithmetization and develop the algebraic preliminaries used throughout the protocol. In §3 we describe the protocol in full, separated into several components. In §4 we prove round-by-round knowledge soundness of the protocol as an IOP and provide error calculations.

1.1. Notation.

We fix a base field $\mathbb{F}$ of prime order p. We assume that a large power of 2 divides $p-1$. We also fix an extension field $\mathbb{F}_{\text{ext}}$ of $\mathbb{F}$. A set $\mathcal{L} \subseteq \mathbb{F}$ is *smooth* if it is a multiplicative coset of $\mathbb{F}^\times$ whose order is a power of 2. Our assumption on $\mathbb{F}$ ensures there are smooth subsets of large order.

- We fix a multiplicative generator $\mathsf{g} \in \mathbb{F}^\times$.

- We fix a smooth subgroup $D \subseteq \mathbb{F}^\times$ of order 2^ℓ where ℓ is a protocol parameter. This D is the *univariate skip domain*. For explicit computational convenience, we further fix a generator $\omega_D := \mathsf{g}^{\frac{p-1}{2^\ell}}$ of the subgroup D.
- We denote a *boolean hypercube* by $\mathbb{H}_n = \{0,1\}^n$ for $n \geq 0$.
- Denote $\mathbb{D}_n = D \times \mathbb{H}_n$ for $n \geq 0$. These are the primary domains we will use for multivariate polynomial evaluation and we will refer to them as *hyperprisms*.
- We extend the definition of $\mathbb{D}_n$ to $n \geq -\ell$ by letting $\mathbb{D}_n = D \times \mathbb{H}_n$ as above for $n \geq 0$ and letting $\mathbb{D}_{-i} = D^{(2^i)} = \{x^{2^i} \mid x \in D\}$ for $0 \leq i \leq \ell$. We fix the generator $\omega_D^{2^i}$ for $D^{(2^i)}$. For all $n \geq -\ell$, observe that the cardinality of $\mathbb{D}_n$ is $2^{\ell+n}$.
- Let $[N] = \{1, 2, \ldots, N\}$.
- Let $\mathrm{bin}_n : \{0, 1, \ldots, 2^n - 1\} \to \mathbb{H}_n$ send an integer to its little-endian binary expansion.
- We define a bijection $\mathrm{ord}_{\ell,n} : [2^{\ell+n}] \to \mathbb{D}_n$ using the lexicographic ordering of the coordinates and the generator of D. We order $\mathbb{D}_n$ by the D coordinate first, followed by the coordinates of $\mathbb{H}_n$.
- We will use plain math symbols such as f to denote functions on finite sets such as $f : \mathbb{D}_n \to \mathbb{F}$. We will often implicitly use the same symbol to denote the *column vector* in $\mathbb{F}^{2^{\ell+n}}$ whose ith entry is $f(\mathrm{ord}_{\ell,n}(i))$. If we ever need to be explicit, we will denote $\mathrm{ord}^*_{\ell,n}(f) : [2^{\ell+n}] \to \mathbb{F}$ and $\vec{f} \in \mathbb{F}^{2^{\ell+n}}$ the column vector of f.
- The "hat" symbol over a function is typically used to remind the reader that the function is a polynomial. Usually such a function is the polynomial extension of another function on a smaller finite domain.
- In particular for $f : \mathbb{D}_n \to \mathbb{F}$, we denote by $\hat{f} \in \mathbb{F}[Z, X_1, \ldots, X_n]$ the unique polynomial such that $\deg_Z(\hat{f}) < |D|$ and $\deg_{X_i}(\hat{f}) < 2$ for all $i \in [n]$ and the restriction of $\hat{f}|_{\mathbb{D}_n} = f$ to $\mathbb{D}_n$ equals f.
- We use bold symbols to denote vectors of evaluation points (e.g., $\boldsymbol{x} = (x_1, \ldots, x_n)$). We may use $\boldsymbol{z}$ or $(z, \boldsymbol{x})$ to denote $(z, x_1, \ldots, x_n)$. We use $\mathbb{F}[Z, \boldsymbol{X}]$ as a shorthand for $\mathbb{F}[Z, X_1, \ldots, X_n]$.
- Given $\boldsymbol{x} = (x_1, \ldots, x_n), \boldsymbol{y} = (y_1, \ldots, y_m)$ we use $\boldsymbol{x} \,\|\, \boldsymbol{y}$ to denote the concatenation $(x_1, \ldots, x_n, y_1, \ldots, y_m)$.
- We will primarily represent matrices as functions $T : \mathbb{D}_n \times [w] \to \mathbb{F}$ where w is the width of the matrix. In these cases the height of the matrix is $h = 2^{\ell+n}$ (in particular it is a multiple of 2^ℓ). We will use the symbols w, h with subscripts to denote the width and height of a matrix. We will try to reserve the symbol n with subscripts to denote the hypercube dimension. We use capital letters to denote matrices. In some special settings, we use bold capital letters (e.g., $\mathbf{T}$) to emphasize that a matrix is column-wise partitioned. This use of bold capital letters will be distinguished from the notation $\mathbb{F}[Z, \boldsymbol{X}]$ based on the context.
- We use $\delta_{x,y}$ to denote the Kronecker delta function. For consistency with the literature, we use eq to denote the relevant polynomial extension, which is often used as a convolution kernel.

1.2. **Protocol Parameters.** We list the parameters that the protocol depend on.

- The base field $\mathbb{F}$ and extension field $\mathbb{F}_{\mathrm{ext}}$.
- The univariate skip parameter ℓ and the generator of the domain D.
- The stacked dimension $n_{\mathrm{stack}} \geq 0$ which determines the stacked domain $\mathbb{D}_{n_{\mathrm{stack}}}$.
- The hash function $\mathfrak{H}$ used.
- The rate ρ of the constrained Reed-Solomon code.

- A smooth domain $\mathcal{L} \subset \mathbb{F}^\times$ of order $2^{\ell+n_{\text{stack}}}/\rho$. All evaluation domains for Reed–Solomon codes are subsets of $\mathcal{L}$.
- The folding parameter k used in the WHIR protocol. Typically set so $k \approx \log_2(n_{\text{stack}})$.
- The security parameter λ used in the WHIR protocol, which determines the number of WHIR queries.

2. Preliminaries

2.1. Review of Circuit Frontend.

We will use the same definitions of AIRs, interactions, and circuits as in [Ope25b].

Definition 2.1.1 (AIR)**.** An algebraic intermediate representation (AIR) is a set of pairs (C_i, S_i), where $C_i \colon \mathbb{F}[x_1, \dots, x_w, y_1, \dots, y_w]$ are *constraint polynomials* and w is a fixed width associated with the AIR. The S_i are special selectors, to be explained below, which can be one of All, First, Last, $\mathsf{Transition}$.

A *trace matrix* is a matrix with entries in $\mathbb{F}$ where the number of rows is a power of two. We define the height of the matrix to be the number of rows and the width to be the number of columns. We say that a trace matrix $\mathbf{T}$ satisfies the AIR $\{(C_i, \mathsf{S}_i)\}$ if the width of $\mathbf{T}$ equals w and for each i the constraint polynomial $C_i(x_1, \dots, x_w, y_1, \dots, y_w)$ evaluates to zero on the following domain:

- If S_i is All, then this applies to all pairs of cyclically consecutive rows $(x_1, \dots, x_w)$, $(y_1, \dots, y_w)$ of $\mathbf{T}$.
- If S_i is First, then this applies to the first pair of cyclically consecutive rows of $\mathbf{T}$.
- If S_i is Last, then this applies to the pair (last row, first row) of $\mathbf{T}$.
- If S_i is $\mathsf{Transition}$, then this applies to all pairs of non-cyclically consecutive rows – that is, like All but except (last, first).

Note that in our definition, the height of the trace matrix is not specified by the AIR – trace matrices of different heights can satisfy the same AIR.

In our arithmetization, we also allow each AIR to specify a partition of the columns $\{1, \dots, w\}$ of the trace matrix into parts of different types:

- Preprocessed
- Cached
- Common

This partitioning specifies to the ZK backend how data is supplied for different parts of the trace matrix. The *preprocessed trace* is data that shared and agreed upon ahead of time between the prover and verifier. The *cached trace* is data that is only available to the prover, but may be cached and reused across different proofs. Lastly the *common trace* is the remaining data only available to the prover. We note that if an AIR requires a preprocessed trace, then the height of the trace matrix is fixed.

For greater flexibility, we allow multiple AIRs in the arithmetization of a single circuit. In other words, a circuit is proved by providing multiple trace matrices. We extend the AIR arithmetization framework with an intermediate representation for constraining relations between different AIRs. This intermediate representation, known as *interactions*, was first introduced by [Val24], building on previous interfaces for lookup tables and permutation arguments.

Definition 2.1.2 (Interactions and buses)**.** An *interaction* of width w and message length $\mathsf{len}(\sigma)$ on *bus* b is a triple $(\hat{\boldsymbol{\sigma}}, \hat{m}, b)$, where:

- $\hat{\boldsymbol{\sigma}} \in \mathbb{F}[x_1, \ldots, x_w, y_1, \ldots, y_w]^{\mathsf{len}(\hat{\boldsymbol{\sigma}})}$ is a sequence of $\mathsf{len}(\hat{\boldsymbol{\sigma}})$ polynomials defining the message.
- $\hat{m} \in \mathbb{F}[x_1, \ldots, x_w, y_1, \ldots, y_w]$ is a polynomial that determines the multiplicity of the corresponding message.
- $b \in \mathbb{F} \setminus \{0\}$ is the bus index specifying the bus. It must be nonzero.

Given a trace matrix $\mathbf{T}$ of width w with entry $\mathbf{T}_{ij}$ on row i and column j, we say that an interaction $(\hat{\boldsymbol{\sigma}}, \hat{m}, b)$ defined on $\mathbf{T}$ *sends* over bus b, for each row i, the image

$$\hat{\boldsymbol{\sigma}}(\mathbf{T}_{i1}, \ldots, \mathbf{T}_{iw}, \mathbf{T}_{\mathrm{next}(i)1}, \ldots, \mathbf{T}_{\mathrm{next}(i)w}) \in \mathbb{F}^{\ell}$$

with multiplicity

$$\hat{m}(\mathbf{T}_{i1}, \ldots, \mathbf{T}_{iw}, \mathbf{T}_{\mathrm{next}(i)1}, \ldots, \mathbf{T}_{\mathrm{next}(i)w}) \in \mathbb{F}.$$

where $\mathrm{next}(i)$ is the cyclic next row in the trace matrix, i.e., $\mathrm{next}(i) = i + 1$ if i is not the last row and next of last row is first row.

Remark 2.1.3. The width of an interaction should not be confused with its message length. The width refers to the width of the corresponding trace matrix, whereas the message length is the length of the message defined by each row of the trace.

In our arithmetization, an AIR is augmented with a set of interactions, where the AIR width and interaction width coincide. An AIR may have multiple interactions, where each interaction may have a different message length and/or bus index.

Definition 2.1.4 (Circuit)**.** A *circuit* $\mathcal{C}$ is a collection of (A, I) where A is an AIR and I is a collection of interactions associated with A.

2.2. **Representations of matrices.** In the proof system protocol, matrices will most naturally arise as maps in the form $T : \mathbb{D}_n \times [w] \to \mathbb{F}$ for $n \geq -\ell$. Such a map corresponds to a matrix of height $2^{\ell+n}$ and width w. For fixed $j \in [w]$, we consider $t_j := T(\bullet, j) : \mathbb{D}_n \to \mathbb{F}$ as a *column vector*. Using the ordering map $\mathrm{ord}_{\ell,n} : [2^{\ell+n}] \simeq \mathbb{D}_n$, we can identify t_j with an element of $\mathbb{F}^{2^{\ell+n}}$. Separately, for each $t_j : \mathbb{D}_n \to \mathbb{F}$, we can take its prismalinear extension $\hat{t}_j \in \mathbb{F}[Z, \boldsymbol{X}]$.

Observe that $T : \mathbb{D}_n \times [w] \to \mathbb{F}$ can also be canonically identified with a map $T : \mathbb{D}_n \to \mathbb{F}^w$. We will implicitly make this identification and use the same notation for both. Analogously, we will use the notation $\hat{T}$ to denote $(\hat{t}_1, \ldots, \hat{t}_w) \in \mathbb{F}[Z, \boldsymbol{X}]^w$.

2.3. **Prismalinear extensions.** We review some fundamental results on polynomial interpolation in multiple variables.

Lemma 2.3.1. *Fix a finite subset $S \subset \mathbb{F}$ and hypercube dimension n.*

(i) *There exists a unique polynomial*

$$\mathsf{eq} \in \mathbb{F}[Z, X_1, \ldots, X_n, Z', X'_1, \ldots, X'_n]$$

with $\deg_Z(\mathsf{eq}), \deg_{Z'}(\mathsf{eq}) = |S| - 1$, $\deg_{X_i}(\mathsf{eq}), \deg_{X'_i}(\mathsf{eq}) = 1$ *such that*

$$\mathsf{eq}((z, \boldsymbol{x}), (z', \boldsymbol{x}')) = \delta_{(z,\boldsymbol{x}),(z',\boldsymbol{x}')}$$

for all $z, z' \in S$, $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{H}_n$.

(ii) *The polynomial* eq *satisfies the identities*

$$\mathsf{eq}_{S,n}((Z, \boldsymbol{X}), (Z', \boldsymbol{X}')) = \mathsf{eq}_S(Z, Z') \, \mathsf{eq}_n(\boldsymbol{X}, \boldsymbol{X}')$$
$$\mathsf{eq}_n(\boldsymbol{X}_1 \,\|\, \boldsymbol{X}_2, \boldsymbol{X}'_1 \,\|\, \boldsymbol{X}'_2) = \mathsf{eq}_{n_1}(\boldsymbol{X}_1, \boldsymbol{X}'_1) \, \mathsf{eq}_{n_2}(\boldsymbol{X}_2, \boldsymbol{X}'_2),$$

where we use subscripts for clarity to distinguish eq *defined over different domains.*

(iii) *For any function* $f : S \times \mathbb{H}_n \to \mathbb{F}$*, there exists a unique polynomial* $\hat{f} \in \mathbb{F}[Z, X_1, \ldots, X_n]$ *with* $\deg_Z(f) < |S|, \deg_{X_i}(f) < 2$ *such that* $\hat{f}(z, \boldsymbol{x}) = f(z, \boldsymbol{x})$ *for all* $(z, \boldsymbol{x}) \in S \times \mathbb{H}_n$*. Moreover* $\hat{f}$ *is given by the explicit convolution formula*

$$\hat{f}(Z, \boldsymbol{X}) = \sum_{(z,\boldsymbol{x}) \in S \times \mathbb{H}_n} f(z, \boldsymbol{x}) \cdot \mathsf{eq}((z, \boldsymbol{x}), (Z, \boldsymbol{X})) \tag{2.1}$$

The lemma follows from standard interpolation techniques and its proof is left to the reader. We will call $\hat{f}$ above the *prismalinear extension* of f. When $S = \{1\}$, this is referred to as the multilinear extension of $f : \mathbb{H}_n \to \mathbb{F}$ in the literature. When $n = 0$, this is univariate Lagrange interpolation over S.

2.4. **Univariate lifting.** Note that we allow matrices with column vectors defined on $\mathbb{D}_n \to \mathbb{F}$ for $-\ell \leq n < 0$. Assume we are in this setting where $n = -i$ for $0 \leq i \leq \ell$. Then $\mathbb{D}_n = D^{(2^i)}$ is a subgroup of D and the prismalinear extension $\hat{f}$ of $f : \mathbb{D}_n \to \mathbb{F}$ is a univariate polynomial of degree $< 2^{\ell - i}$. We define the *lift* $\tilde{f} \in \mathbb{F}[Z]$ of $\hat{f}$ by $\tilde{f}(Z) := \hat{f}(Z^{2^i})$. The lift is a univariate polynomial of degree $< 2^\ell$. More specifically it is the interpolation of the function $D \to \mathbb{F} : z \mapsto f(z^{2^i})$. We similarly define $\widetilde{T}$ for matrix $T : \mathbb{D}_{-i} \times [w] \to \mathbb{F}$.

We generalize the lift to all $-\ell \leq n$ by defining $\tilde{f} = \hat{f}$ for $n \geq 0$ and $\tilde{f}$ as above for $n < 0$. In other words, the lift is non-trivial only for negative n, in which case $\tilde{f}$ is a univariate polynomial. The lift is a lways trivial when $\hat{f}$ is a polynomial in more than one variable.

2.5. **Rotations.** Define the *rotation map*

$$\mathrm{rot} : \mathbb{D}_n \to \mathbb{D}_n : \boldsymbol{z} \mapsto \mathrm{ord}_{\ell,n}((\mathrm{ord}_{\ell,n}^{-1}(\boldsymbol{z}) + 1) \mod 2^{\ell+n}) \tag{2.2}$$

where we use the ordering map to bootstrap a notion of adjacency.

Suppose that we have a polynomial $\hat{f} \in \mathbb{F}[Z, \boldsymbol{X}]$ and its restriction to a function $f : \mathbb{D}_n \to \mathbb{F}$. We define the *rotation kernel* $\kappa_{\mathrm{rot}} : \mathbb{D}_n \times \mathbb{D}_n \to \mathbb{F}$ such that it is the unique convolution kernel that makes the following identity hold for all $\boldsymbol{z}' \in \mathbb{D}_n$:

$$f(\mathrm{rot}(\boldsymbol{z}')) = \sum_{\boldsymbol{z} \in \mathbb{D}_n} f(\boldsymbol{z}) \kappa_{\mathrm{rot}}(\boldsymbol{z}, \boldsymbol{z}').$$

We define the rotation kernel polynomial $\hat{\kappa}_{\mathrm{rot}} \in \mathbb{F}[Z, \boldsymbol{X}, Z', \boldsymbol{X}']$ such that $\hat{\kappa}_{\mathrm{rot}}|_{\mathbb{D}_n \times \mathbb{D}_n} = \kappa_{\mathrm{rot}}$ and $\deg_Z(\hat{\kappa}_{\mathrm{rot}}), \deg_{Z'}(\hat{\kappa}_{\mathrm{rot}}) < |D|$ and $\deg_{X_i}(\hat{\kappa}_{\mathrm{rot}}), \deg_{X_i'}(\hat{\kappa}_{\mathrm{rot}}) < 2$ for all $i \in [n]$. An explicit formula for $\hat{\kappa}_{\mathrm{rot}}$ is given by

$$\hat{\kappa}_{\mathrm{rot}}(\boldsymbol{Z}, \boldsymbol{Z}') = \sum_{\boldsymbol{z}' \in \mathbb{D}_n} \mathsf{eq}(\mathrm{rot}(\boldsymbol{z}'), \boldsymbol{Z}) \cdot \mathsf{eq}(\boldsymbol{z}', \boldsymbol{Z}') \tag{2.3}$$

Remark 2.5.1. Observe that for a given $\boldsymbol{z} \in \mathbb{D}_n$ on the hyperprism, we have the identity

$$\hat{\kappa}_{\mathrm{rot}}(\boldsymbol{z}, \boldsymbol{Z}') = \mathsf{eq}(\mathrm{rot}^{-1}(\boldsymbol{z}), \boldsymbol{Z}').$$

We also observe that $\hat{\kappa}_{\mathrm{rot}}$ over the hyperprism reduces to $\hat{\kappa}_{\mathrm{rot}}$ over hypercube via the formula

$$\begin{aligned}\hat{\kappa}_{\mathrm{rot},\mathbb{D}_n}((Z, \boldsymbol{X}), (Z', \boldsymbol{X}')) = \mathsf{eq}_D(Z, \omega_D Z')\, \mathsf{eq}_{\mathbb{H}_n}(\boldsymbol{X}, \boldsymbol{X}') \\ + \mathsf{eq}_D(Z, 1)\, \mathsf{eq}_D(\omega_D Z', 1)(\hat{\kappa}_{\mathrm{rot},\mathbb{H}_n}(\boldsymbol{X}, \boldsymbol{X}') - \mathsf{eq}_{\mathbb{H}_n}(\boldsymbol{X}, \boldsymbol{X}'))\end{aligned}$$

with subscripts for emphasis.

We define the *rotation convolution operator* $\hat{f} \mapsto \hat{f} \star \hat{\kappa}_{\mathrm{rot}} : \mathbb{F}[Z, \boldsymbol{X}] \to \mathbb{F}[Z, \boldsymbol{X}]$ by

$$(\hat{f} \star \hat{\kappa}_{\mathrm{rot}})(\boldsymbol{Z}) = \sum_{\boldsymbol{z} \in \mathbb{D}_n} \hat{f}(\boldsymbol{z}) \hat{\kappa}_{\mathrm{rot}}(\boldsymbol{z}, \boldsymbol{Z}) = \sum_{\boldsymbol{z}' \in \mathbb{D}_n} \hat{f}(\mathrm{rot}(\boldsymbol{z}'))\, \mathsf{eq}(\boldsymbol{z}', \boldsymbol{Z}). \tag{2.4}$$

Note that the sum on the right-hand side is finite and $f \star \hat{\kappa}_{\text{rot}}$ is a polynomial. Further observe that if we start with a function $f : \mathbb{D}_n \to \mathbb{F}$ and take its prismalinear extension $\hat{f} \in \mathbb{F}[\boldsymbol{Z}]$, then $\hat{f} \star \hat{\kappa}_{\text{rot}}$ is the prismalinear extension of $f_{\text{rot}} : \mathbb{D}_n \to \mathbb{F}$ where $f_{\text{rot}}(\boldsymbol{z}) = f(\text{rot}(\boldsymbol{z}))$.

2.6. **Non-interactive protocols via BCS transform.** This paper focuses on the proof system as a non-interactive protocol. The protocol is obtained by applying the BCS transform [BSCS16] to an interactive oracle proof (IOP). As our focus is on the non-interactive setting, we state all protocols in the non-interactive version, after BCS transform, and leave it to the reader to infer the corresponding IOP.

In particular, instead of saying "the prover sends data to the verifier", we will say that the "Fiat–Shamir transcript observes the data". Instead of saying "the verifier sends a challenge to the prover", we will say that the "Fiat–Shamir transcript samples a challenge". We say "transcript" to reference the Fiat–Shamir transcript when there is no ambiguity. The transcript operations of observe and sample are performed separately and non-interactively by both the prover and the verifier, where the observe operation has different behavior for the prover and the verifier: The prover serializes all data observed in the Fiat–Shamir transcript into a proof. The verifier deserializes the proof, and the Fiat–Shamir transcript observes the proof data in the course of the verification algorithm.

2.7. **Sumcheck.** We review the sumcheck protocol (cf. [LFKN92, BDT24]) and discuss some variations.

Protocol 2.7.1 (Sumcheck)**.** *Given $\hat{f} \in \mathbb{F}[X_1, \ldots, X_n]$, the sumcheck protocol reduces a claim about the value of $\sum_{\boldsymbol{x} \in \mathbb{H}_n} \hat{f}(\boldsymbol{x})$ to a claim about the value of $\hat{f}(\boldsymbol{r})$ at a randomly sampled vector $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^n$ of extension field elements.*

In an interactive protocol, the random vector $\boldsymbol{r}$ is provided by the verifier. In our non-interactive setting, it is sampled from the Fiat–Shamir transcript. The computational analysis depends on the degree of $\hat{f}$. We give the full statement of the protocol, with greater generality, later in Protocol 2.7.4.

We give a variation of the sumcheck protocol which is a reformulation of Gruen's univariate skip [Gru24]. We state it as a standalone protocol independent from its usage in ZeroCheck as it may be of independent interest.

Protocol 2.7.2 (Sumcheck with univariate skip)**.** *Given $\hat{f} \in \mathbb{F}[Z, X_1, \ldots, X_n]$, the sumcheck protocol may be modified with a* univariate skip *in the Z variable to reduce a claim about the value of $\sum_{\boldsymbol{z} \in \mathbb{D}_n} \hat{f}(\boldsymbol{z})$ to a claim about the value of $\hat{f}(\boldsymbol{r})$ at a randomly sampled vector $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^{n+1}$.*

Note that the domains $\mathbb{D}_n$ and $\mathbb{H}_{\ell+n}$ have the same size. The difference between Protocol 2.7.1 over $\mathbb{H}_{\ell+n}$ and Protocol 2.7.2 over $\mathbb{D}_n$ is that in the latter, there is a small decrease in security (one random $\mathbb{F}_{\text{ext}}$ is sampled instead of ℓ), the prover does less computational work, and the verifier needs to do more work in the form of higher degree polynomial interpolations. Thus the univariate skip parameter ℓ can be tuned based on the desired prover–verifier cost tradeoff.

The proof system protocol often encounters situations where multiple independent sumcheck protocols need to be performed. It is well-known to practitioners that these sumchecks can be more efficiently *batched* with the use of additional randomness. We describe the *front-loaded batched sumcheck* protocol (cf. [Irr25]) below. We always use the front-loaded version of the protocol, so we will simply refer to it as the batched sumcheck protocol in the rest of this paper.

Protocol 2.7.3 (Batched sumcheck, front-loaded)**.** *Let $\{\hat{f}_i \in \mathbb{F}[X_1, \ldots, X_{n_i}]\}_{i=1,\ldots,m}$ be a collection of polynomials, where the number of variables may differ by polynomial. Let $n =$*

$\max_i n_i$. *For* $\boldsymbol{r} = (r_1, \ldots, r_n) \in \mathbb{F}_{\text{ext}}^n$, *let* $\boldsymbol{r}_{n_i} = (r_1, \ldots, r_{n_i})$. *We describe a protocol to reduce the computations of*

$$\sum_{\boldsymbol{x} \in \mathbb{H}_{n_i}} \hat{f}_i(\boldsymbol{x})$$

to the evaluations of $\hat{f}_i(\boldsymbol{r}_{n_i})$ *for a single random vector* $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^n$.

First, the transcript samples a random $\lambda \in \mathbb{F}_{\text{ext}}$. *For each* i, *define*

$$\tilde{f}_i(X_1, \ldots, X_n) = \hat{f}_i(X_1, \ldots, X_{n_i}) \prod_{j=n_i+1}^{n} X_j$$

as a polynomial in n *variables. Observe that* $\sum_{\boldsymbol{y} \in \mathbb{H}_n} \tilde{f}_i(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in \mathbb{H}_{n_i}} \hat{f}_i(\boldsymbol{x})$ *since* $\tilde{f}_i(\boldsymbol{y}) = 0$ *unless* $y_{n_i+1} = \cdots = y_n = 1$.

Apply algebraic batching to define $\tilde{f} = \sum_i \lambda^{i-1} \tilde{f}_i \in \mathbb{F}[X_1, \ldots, X_n]$. *The sumcheck claims for each* $\tilde{f}_i$, *and hence each* $\hat{f}_i$, *hold with high probability if and only if the sumcheck claim for* $\sum_{\boldsymbol{y} \in \mathbb{H}_n} \tilde{f}(\boldsymbol{y})$ *holds. The prover and verifier apply the (not batched) sumcheck protocol to* $\tilde{f}$ *to reduce the sumcheck claim to the evaluation of* $\tilde{f}(\boldsymbol{r})$ *for a randomly sampled* $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^n$.

Evaluation of

$$\tilde{f}(\boldsymbol{r}) = \sum_{i=1}^{m} \lambda^{i-1} \hat{f}_i(\boldsymbol{r}_{n_i}) \prod_{j=n_i+1}^{n} r_j$$

follows from the evaluation of $\hat{f}_i(\boldsymbol{r}_{n_i})$ *for all* i *by requiring the verifier to evaluate the right hand side from the* $\hat{f}_i$ *evaluations.*

Batched sumcheck over hyperprisms works in exactly the same way. The protocol holds verbatim if we replace hypercubes $\mathbb{H}_{n_i}, \mathbb{H}_n$ with hyperprisms $\mathbb{D}_{n_i}, \mathbb{D}_n$. We state this version of the batched sumcheck protocol, with the univariate skip, in full for future reference:

Protocol 2.7.4 (Batched sumcheck, front-loaded, with univariate skip)**.** *Let*

$$\left\{\hat{f}_i \in \mathbb{F}[Z, X_1, \ldots, X_{n_i}]\right\}_{i=1,\ldots,m}$$

be a collection of polynomials, where the number of variables may differ by polynomial and $n_i \geq 0$. *Let* $n = \max_i n_i$. *For* $\boldsymbol{r} = (r_0, r_1, \ldots, r_n) \in \mathbb{F}_{\text{ext}}^{n+1}$, *let* $\boldsymbol{r}_{n_i} = (r_0, r_1, \ldots, r_{n_i})$. *We describe a protocol to reduce the computations of*

$$\sum_{\boldsymbol{z} \in \mathbb{D}_{n_i}} \hat{f}_i(\boldsymbol{z})$$

to the evaluations of $\hat{f}_i(\boldsymbol{r}_{n_i})$ *for a single random vector* $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^{n+1}$.

1. *Assume that the Fiat–Shamir transcript has observed commitments to all* $\hat{f}_i$.
2. *The prover computes the claimed sums* $c_i = \sum_{\boldsymbol{x} \in \mathbb{D}_{n_i}} \hat{f}_i(\boldsymbol{x})$. *The transcript observes* c_i *for* $i = 1, \ldots, m$.
3. *The transcript samples random* $\lambda \in \mathbb{F}_{\text{ext}}$.
4. *The verifier computes* $c = \sum_i \lambda^{i-1} c_i$.
5. *Start with a special round* 0 *for the univariate skip:*
   (a) *The prover computes the univariate polynomial*

$$s_0(X) = \sum_{i=1}^{m} \lambda^{i-1} \sum_{\boldsymbol{y} \in \mathbb{H}_{n_i}} \hat{f}_i(X, \boldsymbol{y})$$

   with the second sum over the hypercube.

   (b) *The transcript observes* s_0 in coefficient form.
   (c) *The verifier checks that* $c = \sum_{z \in D} s_0(z)$.
   (d) *The transcript samples random* $r_0 \in \mathbb{F}_{\text{ext}}$.
(6) *Proceed through rounds* $j = 1, \ldots, n$ *of sumcheck. In round* j*, the transcript will have already sampled a vector* $\boldsymbol{r}_{j-1} = (r_0, \ldots, r_{j-1}) \in \mathbb{F}_{\text{ext}}^j$.
   (a) *In round* j*, the prover computes the univariate polynomial*

$$s_j(X) = \sum_{i=1}^{m} \lambda^{i-1} \sum_{\boldsymbol{y} \in \mathbb{H}_{n-j}} \tilde{f}_i(\boldsymbol{r}_{j-1}, X, \boldsymbol{y}).$$

   where $\tilde{f}_i(Z, X_1, \ldots, X_n) = \hat{f}_i(Z, X_1, \ldots, X_{n_i}) \cdot X_{n_i+1} \cdots X_n$*. If* $j > n_i$*, then* $\sum_{\boldsymbol{y} \in \mathbb{H}_{n-j}} \tilde{f}_i(\boldsymbol{r}_{j-1}, X, \boldsymbol{y}) = \hat{f}_i(\boldsymbol{r}_{n_i}) r_{n_i+1} \cdots r_{j-1} \cdot X$*. This means the prover does not need to compute any additional sums involving* $\hat{f}_i$ *once its dimensionality has been exceeded.*
   (b) *The transcript observes* $s_j(1), \ldots, s_j(d)$*, where* $d = \deg s_j$.
   (c) *Verifier sets the claimed value of* $s_j(0)$ *to equal* $s_{j-1}(r_{j-1}) - s_j(1)$.
   (d) *The transcript samples random* $r_j \in \mathbb{F}_{\text{ext}}$.
   (e) *Verifier interpolates the value of* $s_j(r_j)$ *from the values* $s_j(0), \ldots, s_j(d)$.
(7) *After round* n*, the transcript has sampled* $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^n$.
(8) *The transcript observes evaluation claims* v_i *for* $\hat{f}_i(\boldsymbol{r}_{n_i})$.
(9) *The verifier checks that* $s_n(r_n) = \sum_i \lambda^{i-1} v_i \cdot r_{n_i+1} \cdots r_n$.

We remark that the verifier can save computation if the n_i are assumed to be sorted.

2.8. **Evaluating piecewise multilinears.** We record an observation that we learned from [Irr25] that plays a key role in our treatment of multivariate polynomials over different domains.

Lemma 2.8.1 (Piecewise hypercube stacking)**.** *Let* $i \in \mathcal{I}$ *be a finite indexing set. For each* $i \in \mathcal{I}$*, let* n_i *be a positive integer and* S_i *a finite set. Let* n, w *be positive integers such that* $n \geq \max_i n_i$ *and* $2^n w \geq \sum_i 2^{n_i} |S_i|$*. There exists an injection*

$$\jmath : \bigsqcup_{i \in \mathcal{I}} \mathbb{H}_{n_i} \times S_i \hookrightarrow \mathbb{H}_n \times [w]$$

with the following property: given $i \in \mathcal{I}$ *and* $s \in S_i$*, there exists* $j_{i,s} \in [w]$ *and* $\boldsymbol{b}_{i,s} \in \mathbb{H}_{n-n_i}$ *such that*

$$\jmath_i(\boldsymbol{z}, s) = (\boldsymbol{z} \,\|\, \boldsymbol{b}_{i,s}, j_{i,s})$$

where $\jmath_i$ *denotes the restriction of* $\jmath$ *to the* i*-th component.*

Remark 2.8.2. Another important property of $\jmath$ is that the elements $j_{i,s}$ and $\boldsymbol{b}_{i,s}$ are efficiently computable, which will be important for the verifier protocol.

3. Protocol Description

We give a technical overview of the proof system as a non-interactive protocol, deferring security analysis to §4. The protocol will be described with respect to a fixed circuit $\mathcal{C} = \{(A_1, I_1), \ldots, (A_{|\mathcal{C}|}, I_{|\mathcal{C}|})\}$, where we fix a global ordering of the AIRs for indexing purposes.

The prover is given a collection $\mathcal{T} = \{(\mathbf{T}, A_{\mathbf{T}}, I_{\mathbf{T}})\}$ where each $\mathbf{T}$ is a trace matrix and $(A_{\mathbf{T}}, I_{\mathbf{T}})$ is an AIR with interactions from the circuit $\mathcal{C}$. It is required that $|\mathcal{T}| \leq |\mathcal{C}|$, i.e., the number of used AIRs is at most the global number of AIRs. Moreover, the list of $(A_{\mathbf{T}}, I_{\mathbf{T}})$ for $\mathbf{T} \in \mathcal{T}$ must have no duplicates. Each matrix $\mathbf{T}$ of width w will be partitioned as

$$w = w_{\text{pre}} + w_{\text{common}} + w_{\text{cache},0} + \ldots + w_{\text{cache},m_{\text{cache}}}.$$

3.1. **Summary.** The protocol consists of the following high-level steps:

(i) The prover commits to the trace matrices using at least one commitment, where multiple commitments are used if there are cached traces. The commitment is done using a *stacked polynomial commitment scheme* for multivariate polynomials of heterogeneous degrees. The commitment is ultimately a commitment to certain *stacked polynomials*.
(ii) The LogUp GKR protocol is used to reduce the interactions' bus constraints to a claim about the GKR input layer polynomials.
(iii) ZeroCheck/LogUp Input Layer. Reduce the claim about the GKR input layer and the claim that AIR constraints vanish to a claim about column polynomial openings.
(iv) Reduce the column polynomial claims to opening claims for the stacked polynomials.
(v) Run the WHIR protocol to generate proofs of the opening claims.

3.2. **Stacked polynomial commitment.** In this section, we describe how a collection $\mathcal{T}$ of trace matrices is committed to in multiple commitments using a reduction to a more typical multilinear polynomial commitment scheme. Each $\mathbf{T} \in \mathcal{T}$ has a partitioning of its columns into preprocessed, common main, and a possibly empty list of cached mains. The partitioning is a property of the associated AIR, so it is agreed upon ahead of time by the prover and verifier. We view the matrix as a map

$$\mathbf{T} : \mathbb{D}_n \times [w] \to \mathbb{F}$$

where n is the hypercube dimension and w is the overall width. Given the partition $w = w_{\text{pre}} + w_{\text{common}} + w_{\text{cache},1} + \ldots + w_{\text{cache},m_{\text{cache}}}$, we let $\mathbf{T}_{\text{common}} : \mathbb{D}_n \times [w_{\text{common}}] \to \mathbb{F}$ denote the restriction, which corresponds to the matrix with all rows and only a subset of the columns. We let $\mathbf{T}_{\text{pre}}, \mathbf{T}_{\text{cache},j}$ denote similar sub-matrices.

We will commit to $\mathcal{T}$ by:

- During proving key generation, pre-committing to each nonempty $\mathbf{T}_{\text{pre}}$ in a separate stacked polynomial commitment.
- Committing to all $\{\mathbf{T}_{\text{common}}\}_{\mathbf{T}\in\mathcal{T}}$ together in a stacked polynomial commitment described below.
- Committing to each nonempty $\mathbf{T}_{\text{cache},j}$ for $\mathbf{T} \in \mathcal{T}$ in a separate stacked polynomial commitment.

After generating the commitments, the prover transcript must observe all commitments.

Remark 3.2.1. The protocol treats the preprocessed and cached commitments in the same way, with the only difference being whether the prover and verifier agree upon the commitment during the key generation stage or not.

3.2.2. *Stacked matrix construction.* Each commitment will follow the same protocol, so below we describe the *stacked polynomial commitment scheme* that commits a collection of trace matrices into a single commitment.

We start with a collection $\mathcal{T} = \{T : \mathbb{D}_{n_T} \times [w_T] \to \mathbb{F}\}$ of trace matrices (this $\mathcal{T}$ is not necessarily the same as the $\mathcal{T}$ mentioned in previous sections). We will describe the protocol to commit to $\mathcal{T}$ in a polynomial commitment $\mathsf{Com}_{\text{stack},k}(\mathcal{T})$ which depends on the WHIR folding parameter k.

The commitment $\mathsf{Com}_{\text{stack},k}(\mathcal{T})$ is the multivariate polynomial commitment to a *different* matrix $Q_{\mathcal{T}} : \mathbb{D}_{n_{\text{stack}}} \times [w_{\mathcal{T},\text{stack}}] \to \mathbb{F}$. We call this the *stacked trace matrix*, which we presently define. The stacked trace matrix and hence the stacked commitment depends on a protocol parameter n_{stack}. **The protocol requires that $n_T \leq n_{\text{stack}}$ for all hypercube dimensions**

n_T. We can first consider $\mathcal{T}$ as a map

$$\mathcal{T} : \bigsqcup_{T \in \mathcal{T}} \mathbb{D}_{n_T} \times [w_T] \to \mathbb{F}$$

where we recall that $-\ell \leq n_T$ may be negative. Let $\tilde{n}_T = \max(n_T, 0)$. Let $w_{\mathcal{T},\text{stack}} = \lceil (\sum_T w_T \cdot 2^{\ell+\tilde{n}_T})/2^{\ell+n_{\text{stack}}} \rceil$. Note that unlike the widths w_T, the number $w_{\mathcal{T},\text{stack}}$ depends on the hypercube dimensions n_T (and in turn on the heights of the matrices T). We define an injection

$$\iota : \bigsqcup_{T \in \mathcal{T}} \mathbb{D}_{n_T} \times [w_T] \hookrightarrow \bigsqcup_{T \in \mathcal{T}} \mathbb{D}_{\tilde{n}_T} \times [w_T] \to \mathbb{D}_{n_{\text{stack}}} \times [w_{\mathcal{T},\text{stack}}] \tag{3.1}$$

where the first map is induced from the canonical inclusion $\mathbb{D}_{n_T} \hookrightarrow \mathbb{D}_{\tilde{n}_T}$ and the second map is obtained from the product of identity in the D coordinate and the embedding of Lemma 2.8.1. Let $\mathcal{T}_T, \iota_T$ denote the restrictions of $\mathcal{T}, \iota$ to the T-th component. We define the stacked matrix

$$Q_{\mathcal{T}} : \mathbb{D}_{n_{\text{stack}}} \times [w_{\mathcal{T},\text{stack}}] \to \mathbb{F}$$

by $Q_{\mathcal{T}}(\boldsymbol{z}', j') = T(\boldsymbol{z}, j)$ if there exists $T, \boldsymbol{z}, j$ such that $\iota_T(\boldsymbol{z}, j) = (\boldsymbol{z}', j')$ or 0 otherwise[1]. Informally, $Q_{\mathcal{T}}$ is defined by "stacking" the columns of each T, where for $n_T \geq 0$ the column of height $2^{\ell+n_T}$ is stacked directly and for $n_T < 0$ the column is expanded to a column of height 2^ℓ using a stride of size 2^{-n_T}. While we defined $Q_{\mathcal{T}}$ with zero values outside of the image of ι, the protocol will not impose any conditions on the values of $Q_{\mathcal{T}}$ outside of the image of ι.

We make the following observation, which will be a key ingredient used in later polynomial opening proofs:

$$T(\boldsymbol{z}, j) = \sum_{\substack{\boldsymbol{z}' \in \mathbb{D}_{n_{\text{stack}}} \\ j' \in [w_{\mathcal{T},\text{stack}}]}} Q_{\mathcal{T}}(\boldsymbol{z}', j') \delta_{(\boldsymbol{z}', j'), \iota_T(\boldsymbol{z}, j)} \tag{3.2}$$

Equation (3.2) follows from the definition of $Q_{\mathcal{T}}$ and the injectivity of ι.

The commitment $\mathsf{Com}_{\text{stack},k}(\mathcal{T})$ is a certain Merkle tree based matrix commitment to the low-degree extension of $Q_{\mathcal{T}}$, which we review in the next section.

3.2.3. *Review of Reed-Solomon codes.* The polynomial commitment we use is designed for compatibility with the constrained Reed-Solomon code used in WHIR (cf. §3.7). It depends on a rate parameter $\rho < 1$. The blowup factor is $1/\rho$.

This section is general to any matrix Q so take $Q = Q_{\mathcal{T}}$ and $n = n_{\text{stack}}$. Let $\mathcal{L} \subset \mathbb{F}^\times$ be a smooth coset of order $|\mathbb{D}_n|/\rho = 2^{\ell+n}/\rho$. Classically, the Reed-Solomon code with field $\mathbb{F}$, evaluation domain $\mathcal{L} \subset \mathbb{F}$ and degree $2^{\ell+n}$ is

$$\mathsf{RS}[\mathbb{F}, \mathcal{L}, \ell + n] := \{g_{\mathsf{RS}} : \mathcal{L} \to \mathbb{F} \mid \exists \hat{g} \in \mathbb{F}^{<2^{\ell+n}}[X] \text{ s.t. } \forall x \in \mathcal{L},\, g_{\mathsf{RS}}(x) = \hat{g}(x)\}.$$

It was observed by [ZCF23] that this code is equivalently viewed as evaluations of multilinear polynomials:

$$\mathsf{RS}[\mathbb{F}, \mathcal{L}, \ell + n] := \{g_{\mathsf{RS}} : \mathcal{L} \to \mathbb{F} \mid \exists \hat{f}^{\mathsf{coeff}}_{\text{MLE}} \in \mathbb{F}^{<2}[X_1, \ldots, X_{\ell+n}] \text{ s.t.} \\ \forall x \in \mathcal{L},\, g_{\mathsf{RS}}(x) = \hat{f}^{\mathsf{coeff}}_{\text{MLE}}(x^{2^0}, x^{2^1}, \ldots, x^{2^{\ell+n-1}})\}.$$

[1] Another way to say this is that $Q_{\mathcal{T}}$ is the extension by zero $\iota_!(\mathcal{T})$ of $\mathcal{T}$.

Following [Hab24], we apply multilinear interpolation to use the *value-to-coefficient* identification between multilinear polynomials and their univariate representations:

$$\mathsf{RS}[\mathbb{F}, \mathcal{L}, \ell + n] := \Big\{ g_{\mathsf{RS}} : \mathcal{L} \to \mathbb{F} \mid \exists \hat{f}_{\mathrm{MLE}} \in \mathbb{F}^{<2}[X_1, \ldots, X_{\ell+n}] \text{ s.t.} \\ \forall x \in \mathcal{L},\ g_{\mathsf{RS}}(x) = \sum_{i=0}^{2^{\ell+n}-1} \hat{f}_{\mathrm{MLE}}(\mathrm{bin}_{\ell+n}(i)) \cdot x^i \Big\}$$

where we recall that $\mathrm{bin}_{\ell+n} : \{0, \ldots, 2^{\ell+n} - 1\} \to \mathbb{H}_{\ell+n}$ is the binary expansion bijection. To use Reed-Solomon codes in conjunction with the univariate skip technique, we make the further observation that

$$\mathsf{RS}[\mathbb{F}, \mathcal{L}, \ell + n] := \Big\{ g_{\mathsf{RS}} : \mathcal{L} \to \mathbb{F} \mid \exists \hat{f} \in \mathbb{F}^{<2^\ell,2,\ldots,2}[Z, X_1, \ldots, X_n] \text{ s.t.} \\ \forall x \in \mathcal{L},\ g_{\mathsf{RS}}(x) = \sum_{i=0}^{2^{\ell+n}-1} \hat{f}_{\mathrm{MLE}}(\mathrm{bin}_{\ell+n}(i)) \cdot x^i \Big\}$$

where $\mathbb{F}^{<2^\ell,2,\ldots,2}[Z, X_1, \ldots, X_n]$ refers to the space of multvariate polynomials of degree less than 2^ℓ in Z and linear in each X_i, and $\hat{f}_{\mathrm{MLE}} \in \mathbb{F}^{<2}[X_1, \ldots, X_{\ell+n}]$ is the unique multilinear polynomial such that $\hat{f}(Z, X_1, \ldots, X_n) = \hat{f}_{\mathrm{MLE}}(Z^{2^0}, \ldots, Z^{2^{\ell-1}}, X_1, \ldots, X_n)$ (in other words, we use a coefficient-to-coefficient identification in the Z variable). Note that this observation is simply a "mix" of the univariate and multilinear cases above, with the first ℓ hyperdimensions univariate and the remaining n hyperdimensions multilinear.

3.2.4. *Matrix commitment.* We will now define the polynomial commitment to Q as a certain matrix commitment based on Reed-Solomon codes. Let $q_1, \ldots, q_w$ denote the w columns of Q, so $q_j = Q(\bullet, j) : \mathbb{D}_n \to \mathbb{F}$. By Lemma 2.3.1, there exists $\hat{q}_j \in \mathbb{F}[Z, X_1, \ldots, X_n]$ such that $\hat{q}_j(z, \boldsymbol{x}) = q_j(z, \boldsymbol{x})$ for all $(z, \boldsymbol{x}) \in \mathbb{D}_n$. We define the Reed–Solomon codeword of q_j as

$$\mathrm{RS}(q_j) : \mathcal{L} \to \mathbb{F}, \quad \mathrm{RS}(q_j)(x) = \sum_{i=0}^{2^{\ell+n}-1} \hat{q}_{j,\mathrm{MLE}}(\mathrm{bin}_{\ell+n}(i)) \cdot x^i \text{ for } x \in \mathcal{L}. \tag{3.3}$$

where $\hat{q}_j \mapsto \hat{q}_{j,\mathrm{MLE}} \in \mathbb{F}[Z_1, \ldots, Z_\ell, X_1, \ldots, X_n]$ performs a coefficient-to-coefficient identification on $\hat{q}_j$ in the univariate Z coordinate to get a multilinear polynomial. The latter transformation can be computed computationally using inverse discrete Fourier transform followed by zeta transform, with inverse given by Mobius transform followed by Fourier transform.

We apply RS to each column q_j of Q to get the matrix $\mathrm{RS}(Q) : \mathcal{L} \times [w] \to \mathbb{F}$. We define the polynomial commitment $\mathsf{Com}_{\mathrm{stack},k}(Q)$ to be the Merkle root of the matrix $\mathrm{RS}(Q)$. The Merkle tree is defined by hashing 2^k strided rows of $\mathrm{RS}(Q)$ into a single leaf node using the hash function $\mathfrak{H}$: Each leaf node is itself the Merkle root of the row-wise hashes of 2^k rows of $\mathrm{RS}(Q)$, where the row indices are strided by $2^{\ell+n+\log_2(1/\rho)-k}$. The Merkle tree of $\mathrm{RS}(Q)$ has depth $\ell + n + \log_2(1/\rho) - k$. The row stride is chosen for compatibility with the folding step in WHIR.

The commitment $\mathsf{Com}_{\mathrm{stack},k}(Q)$ may be viewed as:

(i) the polynomial commitment to the prismalinear polynomials $\hat{q}_j$,
(ii) the polynomial commitment to associated multilinear polynomials $\hat{q}_{j,\mathrm{MLE}}$, or
(iii) the polynomial commitment to associated univariate polynomials $\hat{q}_{j,\mathrm{uni}}$.

We will use the second view later when discussing WHIR (§3.7).

Protocol 3.2.5 (Stacked polynomial commitment)**.** *Given a collection*

$$\mathcal{T} = \{T : \mathbb{D}_{n_T} \times [w_T] \to \mathbb{F}\}$$

of trace matrices of different heights, the trace matrices are stacked into a single stacked trace matrix $Q_{\mathcal{T}}$. The stacked commitment $\mathsf{Com}_{\mathrm{stack},k}(\mathcal{T})$ is defined to be the matrix commitment of the matrix $\mathrm{RS}(Q_{\mathcal{T}})$. This is a polynomial commitment to the prismalinear extensions $\hat{q}_j$ of the columns of $Q_{\mathcal{T}}$.

3.3. **ZeroCheck.** We provide a formulation of the ZeroCheck protocol tailored to our setting. We will state ZeroCheck without relating it to LogUp first, and then put them together in the protocol in §3.5.

Start with a collection of trace matrices $\mathcal{T} = \{(\mathbf{T}, A_{\mathbf{T}}, I_{\mathbf{T}})\}$ where each $\mathbf{T}$ is partitioned. Recall that the AIR $A_{\mathbf{T}}$ associated to a trace matrix $\mathbf{T}$ consists of a collection of (C, S) pairs where $C \in \mathbb{F}[U_1, \ldots, U_{w_{\mathbf{T}}}, V_1, \ldots, V_{w_{\mathbf{T}}}] = \mathbb{F}[\boldsymbol{U}, \boldsymbol{V}]$ is the constraint polynomial and S is a selector. Given the domain $\mathbb{D}_{n_{\mathbf{T}}}$, we can view the selector as a function $\mathsf{S}_{n_{\mathbf{T}}} : \mathbb{D}_{n_{\mathbf{T}}} \to \{0, 1\}$.

The condition that $\mathbf{T}$ satisfies the constraint (C, S) is equivalent to the condition that

$$C(\mathbf{T}(\boldsymbol{z}), \mathbf{T}(\mathrm{rot}(\boldsymbol{z}))) \cdot \mathsf{S}(\boldsymbol{z}) = 0, \quad \text{for all } \boldsymbol{z} \in \mathbb{D}_{n_{\mathbf{T}}}. \tag{3.4}$$

3.3.1. *Polynomial extension of selectors.* Fix $n \geq -\ell$. Recall from Definition 2.1.1 that the selectors we allow for fixed n are $\mathsf{All}, \mathsf{First}, \mathsf{Last}, \mathsf{Transition}$ as functions on $\mathbb{D}_n$. We provide explicit formulas for the prismalinear extensions of these selectors.

(i) $\widehat{\mathsf{All}}$ is the constant function 1.

(ii) $\widehat{\mathsf{First}}(\boldsymbol{Z}) = \frac{1}{2^\ell} \frac{Z^{2^\ell} - 1}{Z - 1} \cdot \prod_{i=1}^{n} (1 - X_i)$ if $n \geq 0$ or $\frac{1}{2^{\ell+n}} \frac{Z^{2^{\ell+n}} - 1}{Z - 1}$ if $n < 0$.

(iii) $\widehat{\mathsf{Last}}(\boldsymbol{Z}) = \frac{1}{2^\ell} \frac{(\omega_D Z)^{2^\ell} - 1}{\omega_D Z - 1} \cdot \prod_{i=1}^{n} X_i$ if $n \geq 0$ or $\frac{1}{2^{\ell+n}} \frac{(\omega_D^{(2^{-n})} Z)^{2^{\ell+n}} - 1}{\omega_D^{(2^{-n})} Z - 1}$ if $n < 0$.

(iv) $\widehat{\mathsf{Transition}}(\boldsymbol{Z}) = 1 - \widehat{\mathsf{Last}}(\boldsymbol{Z})$

Observe that $(\omega_D Z)^{2^\ell} = Z^{2^\ell}$.

3.3.2. *Constraints as polynomials.* We rewrite (3.4) as

$$C(\hat{\mathbf{T}}(\boldsymbol{z}), \hat{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{z})) \cdot \hat{\mathsf{S}}(\boldsymbol{z}) = 0, \quad \text{for all } \boldsymbol{z} \in \mathbb{D}_{n_{\mathbf{T}}}$$

where $\hat{\mathbf{T}}_{\mathrm{rot}} = \hat{\mathbf{T}} \star \hat{\kappa}_{\mathrm{rot}}$ is the prismalinear extension of $\mathbf{T}_{\mathrm{rot}}$ (see equation (2.4)). Let $\tilde{n}_{\mathbf{T}} = \max(n_{\mathbf{T}}, 0)$. Then using the lifts defined in §2.4, the above is equivalent to

$$C(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{z})) \cdot \widetilde{\mathsf{S}}(\boldsymbol{z}) = 0, \quad \text{for all } \boldsymbol{z} \in \mathbb{D}_{\tilde{n}_{\mathbf{T}}} \tag{3.5}$$

where the only difference is when $n_{\mathbf{T}} < 0$. We caution that when $n_{\mathbf{T}} < 0$, the lift $\widetilde{\mathbf{T}}_{\mathrm{rot}}$ is not the same as $\widetilde{\mathbf{T}} \star \hat{\kappa}_{\mathrm{rot}}$. We apply this lift so that the hyperprism $\mathbb{D}_{\tilde{n}_{\mathbf{T}}}$ always contains D as a factor, which is important for batch sumcheck below.

We now take the prismalinear extension of (3.5) above as a function on $\mathbb{D}_{\tilde{n}_{\mathbf{T}}}$ to get

$$\widetilde{C}_{\mathbf{T}}(\boldsymbol{Z}) = \sum_{\boldsymbol{z} \in \mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \mathsf{eq}(\boldsymbol{Z}, \boldsymbol{z}) \cdot C(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{z})) \cdot \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}(\boldsymbol{z})$$

The equation (3.4) is satisfied (i.e., $\mathbf{T}$ satisfies the constraint) if and only if the polynomial $\hat{C}_{\mathbf{T}} \in \mathbb{F}[\boldsymbol{Z}]$ is identifically zero.

Given a random $\boldsymbol{\xi} \in \mathbb{F}_{\mathrm{ext}}^{\tilde{n}_{\mathbf{T}}+1}$, we then have that (3.4) holds with high probability if $\widetilde{C}_{\mathbf{T}}(\boldsymbol{\xi}) = 0$. The latter is now the condition that

$$\sum_{\boldsymbol{z} \in \mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \mathsf{eq}(\boldsymbol{\xi}, \boldsymbol{z}) \cdot C(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{z})) \cdot \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}(\boldsymbol{z}) \stackrel{?}{=} 0. \tag{3.6}$$

The summand on the left hand side is a polynomial in $\boldsymbol{z}$ because $\mathsf{eq}, \widetilde{\mathbf{T}}, \widetilde{\mathbf{T}}_{\mathrm{rot}}, \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}$ are polynomials. Therefore we can apply sumcheck with univariate skip (Protocol 2.7.2) to reduce (3.6) to an evaluation claim.

3.3.3. *Algebraic batching of multiple constraints.* The condition for $\mathbf{T}$ to satisfy an AIR A (without interactions) is that $\mathbf{T}$ satisfies (C, S) for all $(C, \mathsf{S}) \in A$. This is equivalent to the vanishing of the polynomial $\widetilde{C}_{\mathbf{T}}$ for all $(C, \mathsf{S}) \in A$. We use the well-known technique of algebraic batching (cf. [Hab22, §3.3]) to combine the ZeroCheck protocol for these constraints into a single sumcheck.

Start with a random $\lambda \in \mathbb{F}_{\mathrm{ext}}$. By fixing a total ordering on A, fix a bijection $(C, \mathsf{S}) \mapsto \lambda^{(C,\mathsf{S})} : A \to \{\lambda^0, \dots, \lambda^{|A|-1}\}$. Define

$$\begin{aligned} \widetilde{\mathscr{C}}^{\lambda}_{\mathbf{T},A}(\boldsymbol{Z}) &= \sum_{(C,\mathsf{S})\in A} \lambda^{(C,\mathsf{S})} \cdot \widetilde{C}_{\mathbf{T}}(\boldsymbol{Z}) \\ &= \sum_{\boldsymbol{z}\in\mathbb{D}_n} \mathsf{eq}(\boldsymbol{Z}, \boldsymbol{z}) \sum_{(C,\mathsf{S})\in A} \lambda^{(C,\mathsf{S})} \cdot C(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{z})) \cdot \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}(\boldsymbol{z}) \end{aligned} \tag{3.7}$$

Given a random $\boldsymbol{\xi} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathbf{T}}+1}$ that is independently sampled from λ, then the condition that $\mathbf{T}$ satisfies A holds with high probability if $\widetilde{\mathscr{C}}_{\mathbf{T},A}(\boldsymbol{\xi}) = 0$, which reduces to a sumcheck as in (3.6).

We state the ZeroCheck protocol for $\mathcal{T}$ by applying the ZeroCheck protocol for each $(\mathbf{T}, A)$ in $\mathcal{T}$ and batching the sumchecks for each trace $\mathbf{T}$.

Protocol 3.3.4 (ZeroCheck, multiple AIRs)**.** *Let $n_{\mathcal{T}} = \max_{\mathbf{T}\in\mathcal{T}} \tilde{n}_{\mathbf{T}}$ where $\tilde{n}_{\mathbf{T}} = \max(n_{\mathbf{T}}, 0)$. Let $\lambda \in \mathbb{F}_{\mathrm{ext}}$ and $\boldsymbol{\xi} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$ be independently randomly sampled. The collection $\mathcal{T} = \{(\mathbf{T}, A, I)\}$ satisfies the AIR constraints for all trace matrices with high probability if for each $(\mathbf{T}, A) \in \mathcal{T}$, the sum*

$$\sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \mathsf{eq}(\boldsymbol{\xi}_{n_{\mathbf{T}}}, \boldsymbol{z}) \sum_{(C,\mathsf{S})\in A} \lambda^{(C,\mathsf{S})} \cdot C(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{z})) \cdot \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}(\boldsymbol{z}) \tag{3.8}$$

vanishes, where $\boldsymbol{\xi}_{\tilde{n}_{\mathbf{T}}}$ is the truncation of $\boldsymbol{\xi}$ to $\mathbb{F}_{\mathrm{ext}}^{\tilde{n}_{\mathbf{T}}+1}$. The batched sumcheck protocol can be applied to reduce the computation of the sum (3.8) to the evaluation of

$$\mathsf{eq}(\boldsymbol{\xi}_{\tilde{n}_{\mathbf{T}}}, \boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}) \sum_{(C,\mathsf{S})\in A} \lambda^{(C,\mathsf{S})} \cdot C(\widetilde{\mathbf{T}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}})) \cdot \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}) \tag{3.9}$$

for each $(\mathbf{T}, A)$ with respect to a shared random $\boldsymbol{r} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$. The random $\boldsymbol{r} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$ is sampled and used across the parallel sumchecks, where $\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}$ denotes its truncation to $\mathbb{F}_{\mathrm{ext}}^{\tilde{n}_{\mathbf{T}}+1}$.

The evaluation of (3.9) reduces to the evaluation of

$$\hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}) \text{ and } \hat{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{n_{\mathbf{T}}})$$

by requiring the verifier to directly evaluate (3.9) in terms of $\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}, \hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}), \hat{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{n_{\mathbf{T}}})$, where we let $\boldsymbol{r}_{n_{\mathbf{T}}} = r_0^{2^{-n_{\mathbf{T}}}}$ for $n_{\mathbf{T}} < 0$ so that $\widetilde{\mathbf{T}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}) = \hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}})$.

3.4. **Interactions via LogUp GKR.** We return to the proving context of a collection of trace matrices $\mathcal{T} = \{(\mathbf{T}, A_{\mathbf{T}}, I_{\mathbf{T}})\}$ where each $\mathbf{T}$ is partitioned.

3.4.1. *Review of $\mathbb{F}$-multiset balancing.* Recall the Definition 2.1.2 of interactions and buses. We review what it means for circuit buses to balance with respect to $\mathcal{T}$, with some rephrasing in terms of our hyperprism domains.

The set of possible messages is denoted $\mathbb{F}^+ = \bigsqcup_{i\geq 1} \mathbb{F}^i$ (disjoint union). An $\mathbb{F}$-*multiset* is a function

$$\mathscr{M} : \mathbb{F}^+ \to \mathbb{F}$$

that assigns an $\mathbb{F}$-valued "multiplicity" to each message $\mathbb{F}^+$.

For $(\mathbf{T} : \mathbb{D}_{n_\mathbf{T}} \times [w_\mathbf{T}] \to \mathbb{F}, A_\mathbf{T}, I_\mathbf{T}) \in \mathcal{T}$, we have the interactions $I_\mathbf{T} = \{(\hat{\boldsymbol{\sigma}}, \hat{m}, b)\}$. Recall the rotation map $\mathrm{rot} : \mathbb{D}_{n_\mathbf{T}} \to \mathbb{D}_{n_\mathbf{T}}$ defined in (2.2). Given $\boldsymbol{z} \in \mathbb{D}_{n_\mathbf{T}}$, we can evaluate the message $\hat{\boldsymbol{\sigma}}$ to get message value

$$\boldsymbol{\sigma}_\mathbf{T}(\boldsymbol{z}) := \hat{\boldsymbol{\sigma}}(\mathbf{T}(\boldsymbol{z}, \bullet), \mathbf{T}(\mathrm{rot}(\boldsymbol{z}), \bullet)) \in \mathbb{F}^{\mathsf{len}(\hat{\boldsymbol{\sigma}})}$$

where $\mathsf{len}(\hat{\boldsymbol{\sigma}})$ is the message length and we abuse notation to view $\mathbf{T}(\boldsymbol{z}, \bullet) : [w_\mathbf{T}] \to \mathbb{F}$ as an element of $\mathbb{F}^{w_\mathbf{T}}$. We similarly define the multiplicity value by

$$m_\mathbf{T}(\boldsymbol{z}) := \hat{m}(\mathbf{T}(\boldsymbol{z}, \bullet), \mathbf{T}(\mathrm{rot}(\boldsymbol{z}), \bullet)) \in \mathbb{F}.$$

We can now define the multiset $\mathscr{M}_{\mathcal{T},b} : \mathbb{F}^+ \to \mathbb{F}$ associated to the traces $\mathcal{T}$ on bus b by:

$$\mathscr{M}_{\mathcal{T},b}(\tau) = \sum_{(\mathbf{T},A,I)\in\mathcal{T}} \sum_{\substack{(\hat{\boldsymbol{\sigma}},\hat{m},b')\in I \\ b'=b}} \sum_{\boldsymbol{z}\in\mathbb{D}_{n_\mathbf{T}}} m_\mathbf{T}(\boldsymbol{z}) \cdot \delta_{\tau,\boldsymbol{\sigma}_\mathbf{T}(\boldsymbol{z})}, \quad \tau \in \mathbb{F}^+.$$

where $\delta_{\tau,\sigma_\mathbf{T}(\boldsymbol{z})}$ is the Kronecker delta function that is 1 if $\tau = \sigma_\mathbf{T}(\boldsymbol{z})$ and 0 otherwise. In words, the sum is over all traces, over all interactions for the associated AIR, and then over the hyperprism domain of the trace matrix.

We say that a bus b is *balanced* with respect to $\mathcal{T}$ if $\mathscr{M}_{\mathcal{T},b}$ is identically zero, i.e., $\mathscr{M}_{\mathcal{T},b}(\tau) = 0$ for all $\tau \in \mathbb{F}^+$. In order for the circuit $\mathcal{C}$ to be satisfied, we require all buses to be balanced, i.e., $\mathscr{M}_{\mathcal{T},b} = 0$ for all b.

3.4.2. *LogUp statement.* It is well-known to practitioners [PH23, Ope25a] that $\mathbb{F}$-multiset balancing can be cryptographically proven using LogUp. We summarize the procedure and the LogUp statement to prove below.

We first reduce the multi-bus balancing problem to a single bus by noting the injection

$$\mathbb{F}^+ \times (\mathbb{F} - \{0\}) \hookrightarrow \mathbb{F}^+$$

given by concatenation. This means that we can replace the message value $\boldsymbol{\sigma}_T(\boldsymbol{z})$ with $\boldsymbol{\sigma}_T(\boldsymbol{z}) \,\|\, b$ for a given bus b and ensure that messages from different buses cannot coincide. Hence vanishing of each multiset $\mathscr{M}_{\mathcal{T},b}$ over all buses b is equivalent to vanishing of the global multiset

$$\mathscr{M}_{\mathcal{T}}(\tau) = \sum_{(\mathbf{T},A,I)\in\mathcal{T}} \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \sum_{\boldsymbol{z}\in\mathbb{D}_{n_\mathbf{T}}} m_\mathbf{T}(\boldsymbol{z}) \cdot \delta_{\tau,\boldsymbol{\sigma}_\mathbf{T}(\boldsymbol{z}) \,\|\, b}, \quad \tau \in \mathbb{F}^+.$$

Define the polynomial hash $h_\beta : \mathbb{F}^+ \to \mathbb{F}_{\mathrm{ext}}$ by $h_\beta(\sigma_0, \ldots, \sigma_l) = \sigma_0 + \beta\sigma_1 + \ldots + \beta^l\sigma_l$. If we decompose $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_1, \ldots, \hat{\sigma}_{\mathsf{len}(\hat{\boldsymbol{\sigma}})})$ with each $\hat{\sigma}_j$ an $\mathbb{F}$-valued polynomial, then we have an $\mathbb{F}$-valued polynomial

$$h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b) := \beta^{\mathsf{len}(\hat{\boldsymbol{\sigma}})} b + \sum_{j=1}^{\mathsf{len}(\hat{\boldsymbol{\sigma}})} \beta^{j-1}\hat{\sigma}_j$$

in $2w_\mathbf{T}$ variables.

For the LogUp protocol, the prover transcript must sample two challenges $\alpha, \beta \in \mathbb{F}_{\mathrm{ext}}$.

Theorem 3.4.3 (LogUp, imprecise version)**.** *Let $\mathcal{T}$ be a collection of trace matrices for a circuit $\mathcal{C}$. For $\alpha, \beta \in \mathbb{F}_{\text{ext}}$ independent and uniformly random, if*

$$\sum_{(\mathbf{T},A,I)\in\mathcal{T}} \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \sum_{\boldsymbol{z}\in\mathbb{D}_{n_{\mathbf{T}}}} \frac{\hat{m}(\mathbf{T}(\boldsymbol{z}),\mathbf{T}(\text{rot}(\boldsymbol{z})))}{\alpha + h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\mathbf{T}(\boldsymbol{z}),\mathbf{T}(\text{rot}(\boldsymbol{z})))} = 0 \tag{3.10}$$

then $\mathscr{M}_{\mathcal{T}}$ vanishes (i.e., all buses are balanced) with high probability.

We analyze the soundness of Theorem 3.4.3 in Theorem 4.2.1.

3.4.4. *Fractional sumcheck via GKR.* First, observe we can switch the order of the last two sums in (3.10) to get

$$\sum_{(\mathbf{T},A,I)\in\mathcal{T}} \left(\sum_{\boldsymbol{z}\in\mathbb{D}_{n_{\mathbf{T}}}} \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \frac{\hat{m}(\mathbf{T}(\boldsymbol{z}),\mathbf{T}(\text{rot}(\boldsymbol{z})))}{\alpha + h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\mathbf{T}(\boldsymbol{z}),\mathbf{T}(\text{rot}(\boldsymbol{z})))} \right) = 0.$$

Let $\tilde{n}_{\mathbf{T}} = \max(n_{\mathbf{T}}, 0)$. Assuming that $\mathbb{F}$ is prime and hence 2 is invertible in $\mathbb{F}$, the above is equivalent to the condition that

$$\sum_{(\mathbf{T},A,I)\in\mathcal{T}} 2^{\min(n_{\mathbf{T}},0)} \left(\sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \frac{\hat{m}(\widetilde{\mathbf{T}}(\boldsymbol{z}),\widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z}))}{\alpha + h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\widetilde{\mathbf{T}}(\boldsymbol{z}),\widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z}))} \right) = 0 \tag{3.11}$$

where we use the lift from §2.4 in the case $n_{\mathbf{T}} < 0$.

The protocol for proving (3.11) will be a variation of the LogUp–GKR protocol described in [PH23, §3], which we now describe. The basic idea is to rewrite the sum (3.11) as a *fractional sumcheck* and then compute the fractional sumcheck by applying the GKR protocol to a layered circuit. We use a single layered circuit to handle the sum over all trace matrices.

In order to set up the fractional sumcheck and handle all trace matrices together, we construct an injection

$$\jmath : \bigsqcup_{(\mathbf{T},A,I)\in\mathcal{T}} \mathbb{D}_{\tilde{n}_{\mathbf{T}}} \times I \hookrightarrow \mathbb{H}_{\ell+n_{\text{LogUp}}}$$

where $n_{\text{LogUp}} = \lceil \log_2(\sum_{\mathcal{T}} 2^{\tilde{n}_{\mathbf{T}}}|I|) \rceil$ as follows:

Use Lemma 2.8.1 to get an injection $\jmath' : \bigsqcup_{\mathcal{T}} \mathbb{H}_{\tilde{n}_{\mathbf{T}}} \times I \hookrightarrow \mathbb{H}_{n_{\text{LogUp}}}$. Let $\omega_D \in D$ denote the generator of D. For $(\mathbf{T}, A, I) \in \mathcal{T}$, $(\omega_D^i, \boldsymbol{x}) \in \mathbb{D}_{\tilde{n}_{\mathbf{T}}}$ and $(\hat{\boldsymbol{\sigma}}, \hat{m}, b) \in I$, we define

$$\jmath_{\mathbf{T}}(\omega_D^i, \boldsymbol{x}, (\hat{\boldsymbol{\sigma}}, \hat{m}, b)) = (\text{bin}_\ell(i), \jmath'(\boldsymbol{x}, (\hat{\boldsymbol{\sigma}}, \hat{m}, b))) \tag{3.12}$$

where $\jmath_{\mathbf{T}}$ denotes the restriction of $\jmath$ to the $(\mathbf{T}, A, I)$-th component.

Define functions $p, q : \mathbb{H}_{\ell+n_{\text{LogUp}}} \to \mathbb{F}_{\text{ext}}$ as the extension by zero (resp. α) along $\jmath$ of the numerator (resp. denominator) of the summand in (3.11). More explicitly,

$$p(\boldsymbol{y}) = 2^{\min(n_{\mathbf{T}},0)} \hat{m}(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z}))$$
$$q(\boldsymbol{y}) = \alpha + h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z}))$$

if there exists $(\mathbf{T}, A, I) \in \mathcal{T}, \boldsymbol{z} \in \mathbb{D}_{\tilde{n}_{\mathbf{T}}}, (\hat{\boldsymbol{\sigma}}, \hat{m}, b) \in I$ such that $\jmath_{\mathbf{T}}(\boldsymbol{z}, (\hat{\boldsymbol{\sigma}}, \hat{m}, b)) = \boldsymbol{y}$ and $p(\boldsymbol{y}) = 0, q(\boldsymbol{y}) = \alpha$ otherwise[2].

[2]We set $q(\boldsymbol{y}) = \alpha$ as the default value to avoid divison by zero.

Let $\hat{p}, \hat{q} \in \mathbb{F}_{\text{ext}}[Y_1, \ldots, Y_{\ell+n_{\text{LogUp}}}]$ be the multilinear extensions of p and q. By expanding the equations for $\hat{p}, \hat{q}$ in terms of convolutions with the equality polynomial, we get

$$\hat{p}(\boldsymbol{Y}) = \sum_{(\mathbf{T},A,I)\in\mathcal{T}} 2^{\min(n_{\mathbf{T}},0)} \sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{Y}, \jmath_{\mathbf{T}}(\boldsymbol{z},(\hat{\boldsymbol{\sigma}},\hat{m},b))) \cdot \hat{m}(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z})) \tag{3.13}$$

$$\hat{q}(\boldsymbol{Y}) = \alpha + \sum_{(\mathbf{T},A,I)\in\mathcal{T}} \sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{Y}, \jmath_{\mathbf{T}}(\boldsymbol{z},(\hat{\boldsymbol{\sigma}},\hat{m},b))) \cdot h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z})) \tag{3.14}$$

Protocol 3.4.5 (LogUp, fractional sumcheck via GKR, multiple AIRs)**.** *The vanishing of* (3.10) *is equivalent to the vanishing of*

$$\sum_{\boldsymbol{y}\in\mathbb{H}_{\ell+n_{\text{LogUp}}}} \frac{\hat{p}(\boldsymbol{y})}{\hat{q}(\boldsymbol{y})} = 0. \tag{3.15}$$

There exists a layered circuit such that application of the GKR protocol to the layered circuit reduces the computation of the sum (3.15) *to the evaluation of* $\hat{p}(\boldsymbol{\xi})$ *and* $\hat{q}(\boldsymbol{\xi})$ *at a randomly sampled* $\boldsymbol{\xi} \in \mathbb{F}_{\text{ext}}^{\ell+n_{\text{LogUp}}}$.

To complete the LogUp protocol, we explain how to reduce the evaluation claims on the "input layer" $\hat{p}(\boldsymbol{\xi})$ and $\hat{q}(\boldsymbol{\xi})$ to polynomial opening claims on the trace polynomials and their rotational convolutions. The idea is simply to massage (3.13) and (3.14) for $\boldsymbol{Y} = \boldsymbol{\xi}$ until we can apply (non-fractional) sumcheck. Let $\boldsymbol{\xi} = \boldsymbol{\xi}_1 \,\|\, \boldsymbol{\xi}_2 \,\|\, \boldsymbol{\xi}_3$ for $\boldsymbol{\xi}_1 \in \mathbb{F}_{\text{ext}}^{\ell}, \boldsymbol{\xi}_2 \in \mathbb{F}_{\text{ext}}^{\tilde{n}_{\mathbf{T}}}, \boldsymbol{\xi}_3 \in \mathbb{F}_{\text{ext}}^{n_{\text{LogUp}}-\tilde{n}_{\mathbf{T}}}$. We use formula (3.12) and the property of $\jmath'$ from Lemma 2.8.1 to see that

$$\mathsf{eq}(\boldsymbol{\xi}, \jmath_{\mathbf{T}}(\omega_D^i, \boldsymbol{x}, (\hat{\boldsymbol{\sigma}},\hat{m},b))) = \mathsf{eq}(\boldsymbol{\xi}_1, \text{bin}_\ell(i))\, \mathsf{eq}(\boldsymbol{\xi}_2, \boldsymbol{x})\, \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}})$$

for some $\boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}} \in \mathbb{H}_{n_{\text{LogUp}}-\tilde{n}_{\mathbf{T}}}$. This element depends on $(\mathbf{T}, A, I)$ and $(\hat{\boldsymbol{\sigma}}, \hat{m}, b) \in I$ but we omitted some notation for brevity. We interpolate $(\omega_D^i, \boldsymbol{x}) \mapsto \mathsf{eq}(\xi_1, \text{bin}_\ell(i))\, \mathsf{eq}(\xi_2, \boldsymbol{x})$ into a prismalinear polynomial by interpolating over D to get

$$\mathsf{eq}^\sharp_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(Z, \boldsymbol{X}) := \left(\sum_{i\in[2^\ell]} \mathsf{eq}_D(Z, \omega_D^i)\, \mathsf{eq}_{\mathbb{H}_\ell}(\boldsymbol{\xi}_1, \text{bin}_\ell(i)) \right) \mathsf{eq}_{\mathbb{H}_{\tilde{n}_{\mathbf{T}}}}(\boldsymbol{\xi}_2, \boldsymbol{X}) \tag{3.16}$$

which is a polynomial in $\mathbb{F}[Z, \boldsymbol{X}]$.

We conclude that $\hat{p}(\boldsymbol{\xi})$ and $\hat{q}(\boldsymbol{\xi}) - \alpha$ can both be written in the form

$$\sum_{(\mathbf{T},A,I)\in\mathcal{T}} \left(\sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \mathsf{eq}^\sharp_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{z}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot \hat{C}(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z})) \right)$$

Recall that we constructed $\jmath$ so that $\boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}$ is efficiently computable by the verifier. We can evaluate sums of the above form using batch sumcheck:

Protocol 3.4.6 (LogUp, input layer evaluation via batch sumcheck)**.** *Fix* $\boldsymbol{\xi} = \boldsymbol{\xi}_1 \,\|\, \boldsymbol{\xi}_2 \,\|\, \boldsymbol{\xi}_3$ *for* $\boldsymbol{\xi}_1 \in \mathbb{F}_{\text{ext}}^{\ell}, \boldsymbol{\xi}_2 \in \mathbb{F}_{\text{ext}}^{\tilde{n}_{\mathbf{T}}}, \boldsymbol{\xi}_3 \in \mathbb{F}_{\text{ext}}^{n_{\text{LogUp}}-\tilde{n}_{\mathbf{T}}}$. *The evaluations of* $\hat{p}(\boldsymbol{\xi})$ *and* $\hat{q}(\boldsymbol{\xi}) - \alpha$ *are equivalent to the computations of*

$$\text{sum}_{\hat{p},\mathbf{T},I} = \sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \mathsf{eq}^\sharp_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{z}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot \hat{m}(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z}))$$

$$\text{sum}_{\hat{q},\mathbf{T},I} = \sum_{\boldsymbol{z}\in\mathbb{D}_{\tilde{n}_{\mathbf{T}}}} \mathsf{eq}^\sharp_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{z}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\widetilde{\mathbf{T}}(\boldsymbol{z}), \widetilde{\mathbf{T}}_{\text{rot}}(\boldsymbol{z}))$$

for each $(\mathbf{T}, A, I) \in \mathcal{T}$ *together with the computations of*

$$\sum_{(\mathbf{T},A,I)\in\mathcal{T}} 2^{\min(n_{\mathbf{T}},0)}\,\mathrm{sum}_{\hat{p},\mathbf{T},I} \;\text{ and }\; \sum_{(\mathbf{T},A,I)\in\mathcal{T}} \mathrm{sum}_{\hat{q},\mathbf{T},I} \tag{3.17}$$

The computations of (3.17) *are done directly by the verifier. The batched sumcheck protocol can be applied to reduce the computations of* $\mathrm{sum}_{\hat{p},\mathbf{T},I}$ *and* $\mathrm{sum}_{\hat{q},\mathbf{T},I}$ *for all* $(\mathbf{T}, A, I) \in \mathcal{T}$ *to the evaluations of*

$$\mathsf{eq}^{\sharp}_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot \hat{m}(\widetilde{\mathbf{T}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}})) \tag{3.18}$$

$$\mathsf{eq}^{\sharp}_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\widetilde{\mathbf{T}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}})) \tag{3.19}$$

for each $(\mathbf{T}, I)$ *with respect to a shared random* $\boldsymbol{r} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$. *The random* $\boldsymbol{r} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$ *is sampled and used across the parallel sumchecks, where* $\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}$ *denotes its truncation to* $\mathbb{F}_{\mathrm{ext}}^{\tilde{n}_{\mathbf{T}}+1}$.

The evaluations of (3.18) *and* (3.19) *reduce to the evaluation of*

$$\hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}) \text{ and } (\hat{\mathbf{T}} \star \hat{\kappa}_{\mathrm{rot}})(\boldsymbol{r}_{n_{\mathbf{T}}})$$

by requiring the verifier to directly evaluate (3.18) *and* (3.19) *in terms of* $\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}, \hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}), \hat{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{n_{\mathbf{T}}})$, *where we let* $\boldsymbol{r}_{n_{\mathbf{T}}} = r_0^{2^{-n_{\mathbf{T}}}}$ *for* $n_{\mathbf{T}} < 0$ *so that* $\widetilde{\mathbf{T}}(\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}) = \hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}})$.

We do not apply this protocol directly in the proof system. Instead, we combine it with the ZeroCheck protocol and run Protocol 3.5.1 below.

3.5. **Batch constraint sumcheck.** Observe that Protocol 3.3.4 (ZeroCheck) and Protocol 3.4.6 (LogUp input layer) both consist of batch sumchecks and the batching occurs over the same set of domains, namely those corresponding to $\mathcal{T}$. We combine them into a single protocol to optimize the batching.

Protocol 3.5.1 (Batch constraint sumcheck)**.** *Let* $n_{\mathcal{T}} = \max_{\mathbf{T}\in\mathcal{T}} \tilde{n}_{\mathbf{T}}$ *where* $\tilde{n}_{\mathbf{T}} = \max(n_{\mathbf{T}}, 0)$. *Let* $\lambda \in \mathbb{F}_{\mathrm{ext}}$ *and* $\boldsymbol{\xi} \in \mathbb{F}_{\mathrm{ext}}^{\ell+\max(n_{\mathcal{T}},n_{\mathrm{LogUp}})}$ *be independently randomly sampled. The ZeroCheck protocol on the collection* $\mathcal{T} = \{(\mathbf{T}, A, I)\}$ *together with the evaluations of the LogUp GKR input layer polynomials* $\hat{p}(\boldsymbol{\xi}), \hat{q}(\boldsymbol{\xi})$ *reduces to the following collection of sumchecks: for each* $(\mathbf{T}, A, I)$, *summations over* $\mathbb{D}_{\tilde{n}_{\mathbf{T}}}$ *of the multivariate polynomials*

$$\begin{aligned}
&\mathsf{eq}(\xi_1 \,\|\, \boldsymbol{\xi}_2, \boldsymbol{Z}) \sum_{(C,\mathsf{S})\in A} \lambda^{(C,\mathsf{S})} \cdot C(\widetilde{\mathbf{T}}(\boldsymbol{Z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{Z})) \cdot \widetilde{\mathsf{S}}_{n_{\mathbf{T}}}(\boldsymbol{Z}) \\
&\mathsf{eq}^{\sharp}_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{Z}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot \hat{m}(\widetilde{\mathbf{T}}(\boldsymbol{Z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{Z})) \\
&\mathsf{eq}^{\sharp}_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}(\boldsymbol{Z}) \sum_{(\hat{\boldsymbol{\sigma}},\hat{m},b)\in I} \mathsf{eq}(\boldsymbol{\xi}_3, \boldsymbol{b}_{\mathbf{T},\hat{\boldsymbol{\sigma}}}) \cdot h_\beta(\hat{\boldsymbol{\sigma}} \,\|\, b)(\widetilde{\mathbf{T}}(\boldsymbol{Z}), \widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{Z}))
\end{aligned} \tag{3.20}$$

where $\boldsymbol{\xi}_1 = (\xi_1, \ldots, \xi_\ell) \in \mathbb{F}_{\mathrm{ext}}^{\ell}, \boldsymbol{\xi}_2 = (\xi_{\ell+1}, \ldots, \xi_{\ell+\tilde{n}_{\mathbf{T}}}) \in \mathbb{F}_{\mathrm{ext}}^{\tilde{n}_{\mathbf{T}}}$ *and* $\mathsf{eq}^{\sharp}_{\boldsymbol{\xi}_1,\boldsymbol{\xi}_2}$ *is defined in 3.16.*

The sumchecks above are batched together across all $(\mathbf{T}, A, I) \in \mathcal{T}$ *to reduce the computation of the summations to the evaluations of the polynomials* (3.20) *above at* $\boldsymbol{r}_{\tilde{n}_{\mathbf{T}}}$ *with respect to a shared random* $\boldsymbol{r} \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$. *The evaluations of the polynomials* (3.20) *at* $\boldsymbol{r}$ *reduce to the evaluations of*

$$\hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}) \text{ and } \boldsymbol{t}^{\boldsymbol{r}}_{\mathrm{rot}},$$

where $\boldsymbol{t}^{\boldsymbol{r}}_{\mathrm{rot}} = \hat{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{n_{\mathbf{T}}})$ *if any expression in* (A, I) *uses a column rotation or* $\boldsymbol{t}^{\boldsymbol{r}}_{\mathrm{rot}} = \mathbf{0}$ *otherwise, by requiring the verifier to directly evaluate the rest.*

We note that $\hat{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{r}_{n_{\mathbf{T}}})$ is not needed in the verifier's evaluation above when (A, I) does not use column rotations, which more specifically means that for all $(C, \mathsf{S}) \in A, (\hat{\boldsymbol{\sigma}}, \hat{m}, b) \in I$ the polynomials $C, \hat{\boldsymbol{\sigma}}, \hat{m}$ do not use the second $\widetilde{\mathbf{T}}_{\mathrm{rot}}(\boldsymbol{Z})$ term. Regardless of whether rotations are used, the Fiat–Shamir transcript observes the $\boldsymbol{t}^{\boldsymbol{r}}_{\mathrm{rot}}$ term.

3.6. **Stacked opening reduction.** At this point, the protocol has reduced all computation to the evaluation of

$$\hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}) \text{ and } (\hat{\mathbf{T}} \star \hat{\kappa}_{\mathrm{rot}})(\boldsymbol{r}_{n_{\mathbf{T}}})$$

for each $\mathbf{T} \in \mathcal{T}$ with respect to a random $\boldsymbol{r} = (r_0, r_1, \ldots, r_{n_{\mathcal{T}}}) \in \mathbb{F}_{\mathrm{ext}}^{n_{\mathcal{T}}+1}$. We now describe the protocol to reduce these evaluation claims to polynomial opening claims of the stacked polynomials defined in §3.2. For expositional simplicity, we discuss the case where rotations are needed below. When rotations are not needed and $\boldsymbol{t}^{\boldsymbol{r}}_{\mathrm{rot}} = \mathbf{0}$ in Protocol 3.5.1, the corresponding evaluation claim is skipped below.

Recall that the $\mathbf{T} \in \mathcal{T}$ are partitioned. As described in §3.2, we commit to the trace matrices in $\mathcal{T}$ using multiple commitments

$$\mathcal{C} = \{\mathsf{Com}^{\mathrm{pre},1}_{\mathrm{stack},k}, \ldots, \mathsf{Com}^{\mathrm{pre},m_{\mathrm{pre}}}_{\mathrm{stack},k}, \mathsf{Com}^{\mathrm{common}}_{\mathrm{stack},k}, \mathsf{Com}^{\mathrm{cache},1}_{\mathrm{stack},k}, \ldots, \mathsf{Com}^{\mathrm{cache},m_{\mathrm{cache}}}_{\mathrm{stack},k}\}$$

For all subsequent discussion, all commitments can be treated uniformly (although practical implementations may take advantage of the explicit partition structure[3]). Let $\mathcal{C}$ denote the set of all commitments. We will view the partitioning of the columns of the trace matrices as a surjective map

$$c : \bigsqcup_{\mathbf{T} \in \mathcal{T}} [w_{\mathbf{T}}] \to \mathcal{C}.$$

For a given $\mathsf{Com} \in \mathcal{C}$, in this section we will use $\mathcal{T}_{\mathsf{Com}}$ to denote the set of matrices obtained by taking the sub-matrices of $\mathbf{T} \in \mathcal{T}$ corresponding to $c^{-1}(\mathsf{Com})$, excluding empty matrices. Let

$$\iota_{\mathsf{Com}} : \bigsqcup_{T \in \mathcal{T}_{\mathsf{Com}}} \mathbb{D}_{n_T} \times [w_T] \to \mathbb{D}_{n_{\mathrm{stack}}} \times [w_{\mathsf{Com}}]$$

denote the injection defining the stacking of hyperprisms from (3.1), with $w_{\mathsf{Com}} := w_{\mathcal{T}_{\mathsf{Com}},\mathrm{stack}}$ defined in §3.2 as a function of the trace heights and n_{stack}. To reduce notation, we use w_T to denote the width of the sub-matrix.

We wish to prove the evaluations of $\hat{t}_j(\boldsymbol{r}_{n_{\mathbf{T}}}), (\hat{t}_j \star \hat{\kappa}_{\mathrm{rot}})(\boldsymbol{r}_{n_{\mathbf{T}}})$ for all $\mathbf{T} \in \mathcal{T}$ and $j \in [w_{\mathbf{T}}]$. We proceed by grouping the column indices j by their commitment $c(j)$. Thus we fix $\mathsf{Com} \in \mathcal{C}$ and focus on evaluations for $T \in \mathcal{T}_{\mathsf{Com}}$. We switch to using $\hat{t}_j$ to denote the j-th column of $\hat{T}$ (as opposed to $\hat{\mathbf{T}}$) for $j \in [w_T]$. Let $\{\hat{q}_{\mathsf{Com},j'}\}_{j' \in [w_{\mathsf{Com}}]}$ denote the stacked polynomials corresponding to Com.

The idea is to use (3.2) to express the evaluations as sumchecks. Let $\tilde{n}_T = \max(n_T, 0)$. For $\boldsymbol{z}' \in \mathbb{D}_{n_{\mathrm{stack}}}$, let $\boldsymbol{z}' = \boldsymbol{z}'_{\tilde{n}_T} \,\|\, \boldsymbol{z}'_{>\tilde{n}_T}$ with $\boldsymbol{z}'_{\tilde{n}_T} \in \mathbb{D}_{\tilde{n}_T}$. We define

$$\mathsf{in}_{D,n_T}(Z) = \begin{cases} 2^{n_T} \frac{Z^{2^\ell}-1}{Z^{2^{\ell+n_T}}-1} & \text{if } n_T < 0 \\ 1 & \text{if } n_T \geq 0 \end{cases}$$

For $n_T < 0$, this is the univariate polynomial that equals 1 on $D^{(2^{n_T})}$ and 0 on the rest of D. Below we also use $\mathsf{eq}_{\mathbb{D}_{n_T}}$ to refer to $\mathsf{eq}_{D^{(2^{-n_T})}}$ as a polynomial in two variables when $n_T < 0$.

[3]The implementation may treat preprocessed and cached commitments in the same specialized way, which uses the fact that it is a commitment to a single matrix. At this point in the protocol, there is no distinction between preprocessed versus cached commitments.

Recall the property from Lemma 2.8.1 that $\iota_{\mathsf{Com}}(\boldsymbol{z}, j) = (\boldsymbol{z} \,\|\, \boldsymbol{b}_{T,j}, j_{T,j})$ for some efficiently computable $\boldsymbol{b}_{T,j} \in \mathbb{H}_{n_{\text{stack}}-\tilde{n}_T}$. Using this and combining (2.1) and (3.2), we observe that for variables $\boldsymbol{Z} = (Z, X_1, \ldots, X_{n_T})$,

$$\begin{aligned}\hat{t}_j(\boldsymbol{Z}) &= \sum_{\boldsymbol{z}\in\mathbb{D}_{n_T}} \sum_{\boldsymbol{z}'\in\mathbb{D}_{n_{\text{stack}}}} \hat{q}_{\mathsf{Com},j_{T,j}}(\boldsymbol{z}')\delta_{\boldsymbol{z}'_{\tilde{n}_T},\boldsymbol{z}} \cdot \mathsf{eq}_{\mathbb{H}_{n_{\text{stack}}-\tilde{n}_T}}(\boldsymbol{z}'_{>\tilde{n}_T}, \boldsymbol{b}_{T,j})\, \mathsf{eq}_{\mathbb{D}_{n_T}}(\boldsymbol{z}, \boldsymbol{Z}) \\ &= \sum_{\boldsymbol{z}'\in\mathbb{D}_{n_{\text{stack}}}} \hat{q}_{\mathsf{Com},j_{T,j}}(\boldsymbol{z}')\mathsf{in}_{D,n_T}(z'_0)\, \mathsf{eq}_{\mathbb{D}_{n_T}}(\boldsymbol{z}'_{\tilde{n}_T}, \boldsymbol{Z})\, \mathsf{eq}_{\mathbb{H}_{n_{\text{stack}}-\tilde{n}_T}}(\boldsymbol{z}'_{>\tilde{n}_T}, \boldsymbol{b}_{T,j}). \end{aligned} \tag{3.21}$$

A similar argument shows that

$$(\hat{t}_j \star \hat{\kappa}_{\text{rot}})(\boldsymbol{Z}) = \sum_{\boldsymbol{z}'\in\mathbb{D}_{n_{\text{stack}}}} \hat{q}_{\mathsf{Com},j_{T,j}}(\boldsymbol{z}')\mathsf{in}_{D,n_T}(z'_0)\hat{\kappa}_{\text{rot},\mathbb{D}_{n_T}}(\boldsymbol{z}'_{\tilde{n}_T}, \boldsymbol{Z})\, \mathsf{eq}_{\mathbb{H}_{n_{\text{stack}}-\tilde{n}_T}}(\boldsymbol{z}'_{>\tilde{n}_T}, \boldsymbol{b}_{T,j}). \tag{3.22}$$

We used eq with subscripts to clarify the domains, but we drop this below to simplify notation.

We have formulated the evaluations (3.21) and (3.22) as sumchecks over the same domain $\mathbb{D}_{n_{\text{stack}}}$. We can now apply the batch sumcheck protocol over all $\mathsf{Com} \in \mathcal{C}$, all $T \in \mathcal{T}_{\mathsf{Com}}$, and all $j \in [w_T]$.

Protocol 3.6.1 (Stacked opening reduction)**.** *Let $\mathcal{T}$ be a collection of partitioned trace matrices and let $\mathcal{C}$ be the set of commitments corresponding to the partitioning. Let $n_{\mathcal{T}} = \max_{\mathbf{T}\in\mathcal{T}} n_{\mathbf{T}}$. Let $\boldsymbol{r} \in \mathbb{F}_{\text{ext}}^{n_{\mathcal{T}}+1}$. The batched sumcheck protocol can be applied to reduce the evaluations of*

$$\hat{\mathbf{T}}(\boldsymbol{r}_{n_{\mathbf{T}}}) \text{ and } (\hat{\mathbf{T}} \star \hat{\kappa}_{\text{rot}})(\boldsymbol{r}_{n_{\mathbf{T}}}), \quad \mathbf{T} \in \mathcal{T}$$

to the evaluations of the polynomials

$$\begin{aligned}&\hat{q}_{\mathsf{Com},j_{T,j}}(\boldsymbol{u})\mathsf{in}_{D,n_T}(u_0)\, \mathsf{eq}(\boldsymbol{u}_{\tilde{n}_T}, \boldsymbol{r}_{n_T})\, \mathsf{eq}(\boldsymbol{u}_{>\tilde{n}_T}, \boldsymbol{b}_{T,j}) \\ &\hat{q}_{\mathsf{Com},j_{T,j}}(\boldsymbol{u})\mathsf{in}_{D,n_T}(u_0)\hat{\kappa}_{\text{rot}}(\boldsymbol{u}_{\tilde{n}_T}, \boldsymbol{r}_{n_T})\, \mathsf{eq}(\boldsymbol{u}_{>\tilde{n}_T}, \boldsymbol{b}_{T,j})\end{aligned}$$

for all $\mathsf{Com} \in \mathcal{C}$, $T \in \mathcal{T}_{\mathsf{Com}}$, and $j \in [w_T]$ for a shared random $\boldsymbol{u} \in \mathbb{F}_{\text{ext}}^{n_{\text{stack}}+1}$. These evaluations reduce to the evaluations of

$$\hat{q}_{\mathsf{Com},j'}(\boldsymbol{u})$$

for each $\mathsf{Com} \in \mathcal{C}$, $j' \in [w_{\mathsf{Com}}]$ by requiring the verifier to directly evaluate the rest.

For $\mathbf{T} \in \mathcal{T}$ where the rotation evaluation claim $(\hat{\mathbf{T}}\star\hat{\kappa}_{\text{rot}})(\boldsymbol{r}_{n_{\mathbf{T}}})$ is not needed, the corresponding sumcheck is skipped in the batch sumcheck (equivalently, the zero polynomial is included in the batch sumcheck).

Recall that in keeping with notation of §3.5, we let $\boldsymbol{r}_{n_{\mathbf{T}}} = r_0^{2^{-n_{\mathbf{T}}}}$ for $n_{\mathbf{T}} < 0$.

3.7. **WHIR.** Finally, we must prove the evaluation claims of the stacked polynomials $\hat{q}_{\mathsf{Com},j}(\boldsymbol{u})$ at a random $\boldsymbol{u} \in \mathbb{F}_{\text{ext}}^{n_{\text{stack}}+1}$. We treat each commitment separately, so we drop the subscript Com and consider the prismalinear polynomials $\hat{q}_j \in \mathbb{F}[Z, X_1, \ldots, X_{n_{\text{stack}}}]$ for $j \in [w_{\text{stack}}]$.

Recall from §3.2.3 that associated to each $\hat{q}_j$ we also have the multilinear polynomial $\hat{q}_{j,\text{MLE}} \in \mathbb{F}[X_1, \ldots, X_{n_{\text{stack}}+\ell}]$ and the Reed-Solomon codeword $\text{RS}(q_j) : \mathcal{L} \to \mathbb{F}$. The polynomials $\hat{q}_j$ and $\hat{q}_{j,\text{MLE}}$ satisfy the relation

$$\hat{q}_j(Z, \boldsymbol{X}) = \hat{q}_{j,\text{MLE}}(Z^{2^0}, Z^{2^1}, \ldots, Z^{2^{\ell-1}}, \boldsymbol{X}).$$

If $\boldsymbol{u} = (u_0, u_1, \ldots, u_{n_{\text{stack}}})$ then let

$$\tilde{\boldsymbol{u}} = (u_0, u_0^2, \ldots, u_0^{2^{\ell-1}}, u_1, \ldots, u_{n_{\text{stack}}}) \in \mathbb{F}_{\text{ext}}^{n_{\text{stack}}+\ell}.$$

We have reduced to proving the evaluation claims of multilinear polynomials $\hat{q}_{j,\mathrm{MLE}}$ at $\tilde{\boldsymbol{u}}$. This is now a classical batch polynomial opening problem. We use WHIR ([ACFY24]) as a multilinear polynomial commitment scheme (PCS) with algebraic batching to prove the claims.

Protocol 3.7.1 (Batch WHIR, single opening point)**.** *Sample a random $\mu \in \mathbb{F}_{\mathrm{ext}}$. Define the algebraic batching*

$$\mathrm{RS}(q)^{\mu} = \sum_j \mu^{j-1} \mathrm{RS}(q_j).$$

The evaluation claims $\hat{q}_j(\boldsymbol{u}) \stackrel{?}{=} v_j$ can be proven by running the WHIR protocol to establish proximity of $\mathrm{RS}(q)^{\mu}$ to the constrained Reed–Solomon code $\mathsf{CRS}[\mathbb{F}, \mathcal{L}, \ell + n_{\mathrm{stack}}, \hat{w}, \sigma]$ with $\hat{w}(Z, \boldsymbol{X}) = Z \cdot \mathsf{eq}(\boldsymbol{X}, \tilde{\boldsymbol{u}})$ and $\sigma = \sum_j \mu^{j-1} v_j$.

3.8. **SWIRL protocol in full.** We outline the full non-interactive STARK protocol here. We recall (cf. §2.6) this protocol is obtained by applying the BCS transform to an IOP. We state the protocol in terms of the Fiat–Shamir transcript, and we leave it to the reader to infer the corresponding IOP.

(i) (Key generation) The prover and verifier agree on the circuit $\mathcal{C} = \{(A, I)\}$ before starting the protocol. The prover generates commitments $\mathsf{Com}^{\mathrm{pre},1}_{\mathrm{stack},k}, \ldots, \mathsf{Com}^{\mathrm{pre},m_{\mathrm{pre}}}_{\mathrm{stack},k}$ to any preprocessed trace.

(ii) The prover starts with a collection of partitioned trace matrices $\mathcal{T} = \{(\mathbf{T}, A, I)\}$. The map $(\mathbf{T}, A, I) \mapsto (A, I)$ from $\mathcal{T} \to \mathcal{C}$ is injective but not necessarily surjective (i.e., there may be optional AIRs).

(iii) Transcript observes an unverified[4] hash of the verifying key to protect against weak Fiat–Shamir.

(iv) Transcript observes all public values from the proof.

(v) §3.2: The prover computes the stacked PCS commitments $\mathsf{Com}^{\mathrm{common}}_{\mathrm{stack},k}, \{\mathsf{Com}^{\mathrm{cache},j}_{\mathrm{stack},k}\}$ for $j = 1, \ldots, m_{\mathrm{cache}}$. The transcript observes these commitments. The prover computes the stacked PCS commitments as follows:
  (a) The prover computes the piecewise stacking maps $\iota_{\mathrm{common}}, \{\iota_{\mathrm{cache},j}\}$ and uses them to define the stacked trace matrices $Q_{\mathrm{common}}, \{Q_{\mathrm{cache},j}\}$.
  (b) The prover computes the Reed-Solomon (RS) codewords for the matrix column vectors to get matrices $\mathrm{RS}(Q_{\mathrm{common}}), \{\mathrm{RS}(Q_{\mathrm{cache},j})\}$.
  (c) The prover computes the Merkle trees and Merkle roots of the RS codeword matrices.

(vi) The fractional sumcheck protocol from Protocol 3.4.5 is applied. The transcript samples random $\alpha, \beta \in \mathbb{F}_{\mathrm{ext}}$ to be used in the denominator terms of the fractional sum. The GKR layered circuit is described in [PH23, Section 3.1]. The witness for the layered circuit consists of functions $p_j, q_j : \mathbb{H}_j \to \mathbb{F}_{\mathrm{ext}}$ for layers $j = 0, \ldots, \ell + n_{\mathrm{LogUp}}$. The witness functions are recursively defined starting from layer $\ell + n_{\mathrm{LogUp}}$ with $(p_{\ell+n_{\mathrm{LogUp}}}, q_{\ell+n_{\mathrm{LogUp}}})$ defined as the evaluations of $(\hat{p}, \hat{q})$ and proceeding down to layer 0 using the recursive definition[5]

$$p_{j-1}(\boldsymbol{y}) = p_j(0, \boldsymbol{y}) q_j(1, \boldsymbol{y}) + p_j(1, \boldsymbol{y}) q_j(0, \boldsymbol{y})$$
$$q_{j-1}(\boldsymbol{y}) = q_j(0, \boldsymbol{y}) q_j(1, \boldsymbol{y})$$

[4]Here unverified means from the perspective of the verifier. The hash may be externally verified prior to initiation of the protocol.

[5]The choice to evaluate $0, 1$ from the left makes no theoretical difference, but leads to a better memory layout in practical implementations where hypercube coordinates are represented as little endian integers.

The GKR protocol proceeds in rounds $j = 1, \ldots, \ell + n_{\text{LogUp}}$. In round j, the prover starts with the MLEs $\hat{p}_{j-1}, \hat{q}_{j-1}$ of p_{j-1}, q_{j-1}. The verifier has evaluation claims of $\hat{p}_{j-1}(\boldsymbol{\xi}^{(j-1)}), \hat{q}_{j-1}(\boldsymbol{\xi}^{(j-1)})$ for a randomly sampled $\boldsymbol{\xi}^{(j-1)} \in \mathbb{F}_{\text{ext}}^{j-1}$ from the last round. Note $\boldsymbol{\xi}^{(0)}$ is the empty vector and $\hat{p}_0, \hat{q}_0$ are constants. The value $\frac{p_0}{q_0}$ is the claimed fractional sum.

(a) In round j, the prover and verifier apply the batch sumcheck protocol to the MLEs $\hat{p}_{j-1}, \hat{q}_{j-1}$ using the equalities

$$\hat{p}_{j-1}(\boldsymbol{Y}) = \sum_{\boldsymbol{y} \in \mathbb{H}_{j-1}} \mathsf{eq}_{j-1}(\boldsymbol{Y}, \boldsymbol{y}) \cdot \big(\hat{p}_j(0, \boldsymbol{y})\hat{q}_j(1, \boldsymbol{y}) + \hat{p}_j(1, \boldsymbol{y})\hat{q}_j(0, \boldsymbol{y})\big)$$
$$\hat{q}_{j-1}(\boldsymbol{Y}) = \sum_{\boldsymbol{y} \in \mathbb{H}_{j-1}} \mathsf{eq}_{j-1}(\boldsymbol{Y}, \boldsymbol{y}) \cdot \big(\hat{q}_j(0, \boldsymbol{y})\hat{q}_j(1, \boldsymbol{y})\big)$$

In the batch sumcheck, the transcript samples randomness $\lambda_j \in \mathbb{F}_{\text{ext}}$ for batching and the protocol reduces the evaluation claims of $\hat{p}_{j-1}(\boldsymbol{\xi}^{(j-1)}), \hat{q}_{j-1}(\boldsymbol{\xi}^{(j-1)})$ to the evaluation claims of

$$\hat{p}_j(0, \boldsymbol{\rho}^{(j-1)}), \hat{p}_j(1, \boldsymbol{\rho}^{(j-1)}), \hat{q}_j(0, \boldsymbol{\rho}^{(j-1)}), \hat{q}_j(1, \boldsymbol{\rho}^{(j-1)}) \tag{3.23}$$

for a randomly sampled $\boldsymbol{\rho}^{(j-1)} \in \mathbb{F}_{\text{ext}}^{j}$.

(b) Observe that $\hat{p}_j(\bullet, \boldsymbol{\rho}^{(j-1)}), \hat{q}_j(\bullet, \boldsymbol{\rho}^{(j-1)})$ are linear polynomials. The transcript observes the claimed linear polynomials in terms of their evaluation claims (3.23).

(c) The transcript samples another random $\mu_j \in \mathbb{F}_{\text{ext}}$. The protocol uses μ_j to reduce the evaluation claims of $\hat{p}_j(0, \boldsymbol{\rho}^{(j-1)}), \hat{p}_j(1, \boldsymbol{\rho}^{(j-1)})$ to the evaluation claim of $\hat{p}_j(\boldsymbol{\xi}^{(j)})$ with $\boldsymbol{\xi}^{(j)} = (\mu_j, \boldsymbol{\rho}^{(j-1)})$. Similarly, it reduces the denominator evaluation claim to the evaluation claim of $\hat{q}_j(\boldsymbol{\xi}^{(j)})$. Now proceed to the next round of GKR.

(vii) Prover and verifier apply a *batch constraint sumcheck* for ZeroCheck and the evaluation claims of the input layer of the LogUp GKR circuit. Prover and verifier sample two random elements $\lambda, \mu \in \mathbb{F}_{\text{ext}}$ for batching purposes. The λ is used for algebraic batching of constraint polynomials per AIR. The μ is the batching factor in the batch sumcheck. The protocol can also use a single random element λ for both batching purposes, but we distinguish them to improve the soundness of the protocol. The batch sumcheck is applied as described in Protocol 3.5.1, where the random vector $\boldsymbol{\xi} \in \mathbb{F}_{\text{ext}}^{\ell + n_{\text{global}}}$ is set to equal the randomly sampled $\boldsymbol{\xi}^{(\ell + n_{\text{LogUp}})}$ from the last round of GKR together with additional sampled elements if $n_{\text{LogUp}} < n_{\text{global}}$.

In total the batch sumcheck batches $3|\mathcal{T}|$ polynomials. If we let $s_{p,\mathbf{T}}, s_{q,\mathbf{T}}, s_{\text{zc},\mathbf{T}}$ denote the sumcheck claims associated with a single trace $\mathbf{T} \in \mathcal{T}$ for the LogUp numerator claim, LogUp denominator claim, and ZeroCheck claim, then the total ordering of the batch sumcheck is $s_{p,\mathbf{T}_1}, s_{q,\mathbf{T}_1}, \ldots, s_{p,\mathbf{T}_{|\mathcal{T}|}}, s_{q,\mathbf{T}_{|\mathcal{T}|}}, s_{\text{zc},\mathbf{T}_1}, \ldots, s_{\text{zc},\mathbf{T}_{|\mathcal{T}|}}$ where the total ordering of $\mathcal{T}$ is in descending order of $n_{\mathbf{T}}$ (with tie breaks determined by ordering of the AIRs in the verifying key).

(viii) Apply Protocol 3.6.1 to reduce the evaluation claims of $\hat{\mathbf{T}}$ and $\hat{\mathbf{T}} \star \hat{\kappa}_{\text{rot}}$ at $\boldsymbol{r}_{n_{\mathbf{T}}}$ for each $\mathbf{T} \in \mathcal{T}$ to evaluation claims of $\hat{q}_{\mathsf{Com},j'}(\boldsymbol{u})$ for a random $\boldsymbol{u} \in \mathbb{F}_{\text{ext}}^{n_{\text{stack}}+1}$.

(ix) Apply Protocol 3.7.1 to prove the evaluation claims of $\hat{q}_{\mathsf{Com},j'}(\boldsymbol{u})$ via WHIR polynomial opening proofs with respect to the commitments

$$\{\mathsf{Com}_{\text{stack},k}^{\text{pre},j}\}_{j=1,\ldots,m_{\text{pre}}}, \mathsf{Com}_{\text{stack},k}^{\text{common}}, \{\mathsf{Com}_{\text{stack},k}^{\text{cache},j}\}_{j=1,\ldots,m_{\text{cache}}}$$

generated in steps (i),(iii). The polynomials $\hat{q}_{\mathsf{Com},j'}$ are algebraically batched together, across all commitments. The WHIR protocol is applied to the batched polynomial as described in [ACFY24, §2.1.3].

4. Security Analysis

We analyze the round-by-round soundness [CCH+19] and round-by-round proof of knowledge [CMS19] of the proof system as an interactive oracle proof (IOP). Fix a proximity parameter $\delta_0 \in (0, 1 - \sqrt{\rho})$ for the first round of the final WHIR invocation. Later rounds of that WHIR execution may use different proximity parameters, and these are absorbed into the term $\varepsilon_{\mathsf{WHIR}}$ below. In the list-decoding regime, the commitment tuple can leave multiple tuples of stacked polynomials viable. We write L_{PCS} for an upper bound on the number of tuples of stacked polynomials consistent with all commitments and within distance δ_0 of the corresponding first-round Reed–Solomon codes. This list-decoding ambiguity matters only when we combine the earlier algebraic reductions with the first round of the final proximity proof. For the initial LogUp fractional-sum round, different decoded trace tuples may induce different interaction multisets, so the corresponding soundness bound must be union-bounded over the list in the full protocol. Likewise, the batch-constraint and stacked-opening reductions are proved for a fixed tuple of trace polynomials and acquire a list-size factor only when lifted to the full IOP. Pure IP subprotocols once the prover's messages are fixed, such as the GKR rounds, are stated below without this extra factor. In the unique-decoding regime, $L_{\mathsf{PCS}} = 1$.

4.1. RBR soundness of batch sumcheck.

Since our protocol makes multiple uses of batch sumcheck, we start by reviewing the round-by-round (RBR) soundness of batch sumcheck as an interactive proof (IP).

Theorem 4.1.1 (Batched sumcheck, front-loaded, with univariate skip)**.** *Protocol 2.7.4 has round-by-round (knowledge) soundness with error*

$$\varepsilon = \max\left\{\frac{m-1}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{d_0}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{d}{|\mathbb{F}_{\mathrm{ext}}|}\right\}$$

where we use the notation of Protocol 2.7.4 and

- *m is the number of polynomials being batched,*
- *d_j (resp. d_0) is the maximum of $\deg_{X_j}(\hat{f}_i)$ (resp. $\deg_Z(\hat{f}_i)$) over all i,*
- *$d = \max_{j=1,\dots,n} d_j$.*

Proof. The front-loading of polynomials of different variables does not affect the soundness analysis, so we consider batch sumcheck over polynomials all of $n+1$ variables. Algebraic batching introduces an extra round with a newly sampled batching randomness λ. By the Schwartz–Zippel lemma applied to $\sum_{i=1}^{m} \lambda^{i-1} c_i \in \mathbb{F}_{\mathrm{ext}}[\lambda]$, we get error $\frac{m-1}{|\mathbb{F}_{\mathrm{ext}}|}$. The error for the subsequent sumcheck rounds follows from the RBR soundness error of standard sumcheck [CCH+19, LFKN92]. We emphasize the $\frac{d_0}{|\mathbb{F}_{\mathrm{ext}}|}$ term to highlight the contribution from the univariate skip round.

RBR knowledge soundness of algebraic batching is immediate since the sum claims are included in the transcript prior to batching. Round-by-round proof of knowledge [CMS19] of sumcheck is also immediate since the witness in each round is directly included in the transcript. □

4.2. **RBR soundness of the protocol.** We analyze the RBR soundness of the full protocol stage by stage.

Theorem 4.2.1 (Interactions via LogUp–GKR)**.** *Protocol 3.4.5 has round-by-round (knowledge) soundness with error*

$$\varepsilon_{\mathsf{logup}} = \max\{L_{\mathsf{PCS}} \cdot \varepsilon_{\mathsf{logup-frac}}, \varepsilon_{\mathsf{logup-gkr}}\} \tag{4.1}$$

$$\varepsilon_{\mathsf{logup-frac}} = \frac{(\max_{\hat{\boldsymbol{\sigma}}} \mathsf{len}(\hat{\boldsymbol{\sigma}}) + 1)(|\operatorname{supp} \mathscr{M}_{\mathcal{T}}| - 1)}{|\mathbb{F}_{\mathrm{ext}}|} \tag{4.2}$$

$$\varepsilon_{\mathsf{logup-gkr}} = \frac{3}{|\mathbb{F}_{\mathrm{ext}}|} \tag{4.3}$$

where $\max_{\hat{\boldsymbol{\sigma}}} \mathsf{len}(\hat{\boldsymbol{\sigma}})$ *is the maximum length of any message* $\hat{\boldsymbol{\sigma}}$ *in any interaction* I *in the circuit, and* $|\operatorname{supp} \mathscr{M}_{\mathcal{T}}|$ *is the cardinality of the support of the global LogUp multiset (i.e., it is the number of distinct interaction messages in* $\mathbb{F}^+$ *across all traces).*

Proof. We separate the protocol into: (i) an initial round where α, β are sampled and fractional sum claim is computed, and (ii) the rounds comprising the GKR protocol. For (i), [Ope25a, Theorem 3.4] gives the bound $\varepsilon_{\mathsf{logup-frac}}$ for one fixed interaction multiset. In the list decoding setting relevant to the full SWIRL protocol, the commitments may be consistent with up to L_{PCS} candidate trace tuples, and hence up to L_{PCS} candidate interaction multisets. The verifier accepts if the sampled (α, β) makes the fractional-sum identity hold for any one of these multisets, so this term contributes at most $L_{\mathsf{PCS}} \cdot \varepsilon_{\mathsf{logup-frac}}$ by a union bound over the list. For (ii), once the prover's GKR messages are fixed, the GKR protocol soundness is entirely transcript defined without reliance on the list decoding, so its RBR soundness error is $\varepsilon_{\mathsf{logup-gkr}}$ by the batch sumcheck protocol [CCH+19], where the sumcheck polynomial has per-variable degree 3 and 2 polynomials are batched. □

Theorem 4.2.2 (Batch constraint sumcheck)**.** *For a constraint–selector pair* $(C, \mathsf{S}) \in A$*, let* $d_{(C,\mathsf{S})}$ *denote the sum of the degree of* C *and the maximum degree of* S *in any variable. For interaction* $(\hat{\boldsymbol{\sigma}}, \hat{m}, b) \in I$*, let* $d_{(\hat{\boldsymbol{\sigma}},\hat{m},b)} = \max(\deg(\hat{\boldsymbol{\sigma}}), \deg(\hat{m}))$*. Let* $d_{(A,I)}$ *denote the maximum of all* $d_{(C,\mathsf{S})}$ *and* $d_{(\hat{\boldsymbol{\sigma}},\hat{m},b)}$ *over all constraints and interactions that appear in* (A, I)*.*

Protocol 3.5.1 has round-by-round (knowledge) soundness with error

$$\varepsilon_{\mathsf{bc}} = \max\left\{\frac{2^\ell - 1 + n_{\mathcal{T}}}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{N_{\mathcal{C}} - 1}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{3|\mathcal{T}| - 1}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{(d_{\mathcal{C}} + 1)(2^\ell - 1)}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{d_{\mathcal{C}} + 1}{|\mathbb{F}_{\mathrm{ext}}|}\right\} \tag{4.4}$$

where

- $N_{\mathcal{C}} = \max_{A \in \mathcal{C}} |A|$ *is the maximum number of constraints in any individual AIR* $A = \{(C, \mathsf{S})\}$ *in the circuit* $\mathcal{C}$*,*
- $|\mathcal{T}| \le |\mathcal{C}|$ *is the number of nonempty trace matrices,*
- $d_{\mathcal{C}}$ *is the maximum of* $d_{(A,I)}$ *over all* $(A, I) \in \mathcal{C}$*,*

Proof. The RBR soundness error is the maximum over the errors in the following groups of rounds that make up Protocol 3.5.1.

ZeroCheck reduction: the transcript samples $\boldsymbol{\xi} \in \mathbb{F}_{\mathrm{ext}}^{\ell + \max(n_{\mathcal{T}}, n_{\mathrm{LogUp}})}$. For ZeroCheck we only consider the element $\boldsymbol{\xi}' = \xi_1 \,\|\, \boldsymbol{\xi}_2 \in \mathbb{F}_{\mathrm{ext}}^{1+n_{\mathcal{T}}}$. Suppose there exists $(\mathbf{T}, A) \in \mathcal{T}$ and a constraint $(C, \mathsf{S}) \in A$ that is not satisfied by the trace $\mathbf{T}$. Then the polynomial $\widetilde{C}_{\mathbf{T}}(\boldsymbol{Z})$ is not identically zero. Note that $\widetilde{C}_{\mathbf{T}}(\boldsymbol{Z})$ is prismalinear in $\boldsymbol{Z}$. By the Schwartz–Zippel lemma, the probability that $\widetilde{C}_{\mathbf{T}}(\boldsymbol{\xi}') = 0$ is

$$\le \frac{2^\ell - 1 + n_{\mathcal{T}}}{|\mathbb{F}_{\mathrm{ext}}|}.$$

This contributes the error for this round.

Constraint algebraic batching: suppose that we have $\widetilde{C}_{\mathbf{T}}(\boldsymbol{Z})$ not identically zero as above, for a particular $(C, \mathsf{S}) \in A$. Furthermore $\widetilde{C}_{\mathbf{T}}(\boldsymbol{\xi}') \neq 0$. Then the probability that randomly sampled λ can result in algebraically batched $\sum_{(C,S)\in A} \lambda^{(C,S)} \widetilde{C}_{\mathbf{T}}(\boldsymbol{\xi}') = 0$ is $\leq \frac{|A|-1}{|\mathbb{F}_{\mathrm{ext}}|}$ again by Schwartz–Zippel. Taking the maximum over all AIRs $A \in \mathcal{C}$, we conclude that the error for this round is

$$\leq \frac{N_{\mathcal{C}} - 1}{|\mathbb{F}_{\mathrm{ext}}|}.$$

The remaining rounds comprise the batch sumcheck protocol applied to $3|\mathcal{T}|$ polynomials. Applying Theorem 4.1.1 to the polynomials as stated in Protocol 3.5.1, we get the error term

$$\max\left\{\frac{3|\mathcal{T}|-1}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{(d_{\mathcal{C}}+1)(2^\ell-1)}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{d_{\mathcal{C}}+1}{|\mathbb{F}_{\mathrm{ext}}|}\right\}.$$

Here we use the fact that $\deg_{X_j}(\widetilde{\mathscr{C}}^{\lambda}_{\mathbf{T},A}(\boldsymbol{Z})) \leq d_{\mathcal{C}}+1$, where the $+1$ comes from the eq term. □

Theorem 4.2.3 (Stacked opening reduction)**.** *Protocol 3.6.1 has round-by-round (knowledge) soundness with error*

$$\varepsilon_{\mathsf{stack}} = \max\left\{\frac{\sum_{\mathbf{T}\in\mathcal{T}} 2w_{\mathbf{T}} - 1}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{2\cdot(2^\ell-1)}{|\mathbb{F}_{\mathrm{ext}}|}, \frac{2}{|\mathbb{F}_{\mathrm{ext}}|}\right\} \tag{4.5}$$

where $\sum_{\mathbf{T}\in\mathcal{T}} w_{\mathbf{T}}$ is the total number of columns across all partitioned trace matrices in $\mathcal{T}$.

Proof. The RBR soundness is a direct application of Theorem 4.1.1. The number of polynomials to batch corresponds to 2 times the number of columns due to the need to handle both $\hat{\mathbf{T}}$ and $\hat{\mathbf{T}}_{\mathrm{rot}}$. The sumcheck polynomial has degree $2\cdot(2^\ell-1)$ in Z and degree 2 in X_j because $\hat{q}_{\mathsf{Com},j_{T,j}}$ is prismalinear and we observe that $\mathsf{in}_{D,n_T}(z_0')\,\mathsf{eq}_{\mathbb{D}_{n_T}}(\boldsymbol{z}'_{\tilde{n}_T}, \boldsymbol{Z})\,\mathsf{eq}_{\mathbb{H}_{n_{\mathsf{stack}}-\tilde{n}_T}}(\boldsymbol{z}'_{>\tilde{n}_T}, \boldsymbol{b}_{T,j})$ and $\mathsf{in}_{D,n_T}(z_0')\hat{\kappa}_{\mathrm{rot},\mathbb{D}_{n_T}}(\boldsymbol{z}'_{\tilde{n}_T}, \boldsymbol{Z})\,\mathsf{eq}_{\mathbb{H}_{n_{\mathsf{stack}}-\tilde{n}_T}}(\boldsymbol{z}'_{>\tilde{n}_T}, \boldsymbol{b}_{T,j})$ both have degree $2^\ell-1$ in Z and degree 1 in X_j. □

Theorem 4.2.4 (Batch WHIR)**.** *Assume the first round of the WHIR protocol is instantiated with proximity parameter δ_0. Let $w_{\mathcal{T},\mathrm{stack}}$ be the number of codewords batched in Protocol 3.7.1. Let $N_{\mathsf{BCHKS}}(w_{\mathcal{T},\mathrm{stack}}, \rho, \delta_0)$ denote the exceptional-set bound for μ-power batching of $w_{\mathcal{T},\mathrm{stack}}$ Reed–Solomon codewords. In the list-decoding regime, this is the bound from* [BCH+25, Theorem 4.5]*; in the unique-decoding regime, we may take*

$$N_{\mathsf{BCHKS}}(w_{\mathcal{T},\mathrm{stack}}, \rho, \delta_0) = (w_{\mathcal{T},\mathrm{stack}} - 1)|\mathcal{L}|.$$

Let $\varepsilon_{\mathsf{WHIR}}$ denote the maximum of the round-by-round soundness errors obtained from [ACFY24, Theorem 5.2] *for the resulting WHIR instance, with the chosen proximity parameters in each WHIR round. Protocol 3.7.1 has round-by-round (knowledge) soundness with error*

$$\varepsilon_{\mathsf{bWHIR}} = \max\left\{\frac{N_{\mathsf{BCHKS}}(w_{\mathcal{T},\mathrm{stack}}, \rho, \delta_0)}{|\mathbb{F}_{\mathrm{ext}}|}, \varepsilon_{\mathsf{WHIR}}\right\}. \tag{4.6}$$

In the list-decoding regime, let $m \geq 1$ be an integer and define

$$\delta_0 := 1 - \sqrt{\rho}\left(1 + \frac{1}{2m}\right)$$

and $\eta := 1 - \sqrt{\rho} - \delta_0$ (which implies $m = \frac{\sqrt{\rho}}{2\eta}$) and

$$D_X := \left(m + \frac{1}{2}\right) |\mathcal{L}| \sqrt{\rho}$$
$$D_Y := \left(m + \frac{1}{2}\right) \frac{1}{\sqrt{\rho}}$$
$$D_Z := \max\left\{ D_Y, \frac{1}{3}\left(m + \frac{1}{2}\right)^2 \frac{1}{\rho} \right\}$$

which are the values from [BCH+25, Lemma 3.1] with proximity radius δ_0 (denoted γ in [BCH+25]), $n = |\mathcal{L}|$, and $k = \rho|\mathcal{L}|$, where $\mathcal{L}$ is the Reed–Solomon codeword domain. By [BCH+25, Equation (13) and Theorem 4.5], we may take

$$N_{\mathsf{BCHKS}}(w_{\mathcal{T},\mathrm{stack}}, \rho, \delta_0) = (w_{\mathcal{T},\mathrm{stack}} - 1)\left(2D_X D_Y^2 D_Z + (\delta_0|\mathcal{L}| + 1)D_Y\right) \tag{4.7}$$

in Theorem 4.2.4, where we removed the requirement that $m \geq 3$ by ensuring that $D_Z \geq D_Y$, as noted after the statement of [Lemma 3.1]BCHKS25.

Since the Guruswami–Sudan interpolant has Y-degree at most D_Y, we may also take

$$L_{\mathsf{PCS}} \leq D_Y = \frac{m + \frac{1}{2}}{\sqrt{\rho}}.$$

The error $\varepsilon_{\mathsf{WHIR}}$ of [ACFY24, Theorem 5.2] relies on the error of the function $\mathsf{Gen}(l; \alpha) = (1, \alpha, \ldots, \alpha^{l-1})$ as a proximity generator with mutual correlated agreement for Reed–Solomon codes. Mutual correlated agreement is established in [BCGM26, Theorem 9.2] with an error bound obtained from [BCI+20, Theorem 5.1]. The proof of mutual correlated agreement can be applied directly with the improved bound from [BCH+25, Theorem 4.5].

Proof. We perform the μ-power batching of the codewords *before* instantiating WHIR. In the list-decoding regime, for a fixed opening claim the relevant constrained Reed–Solomon code in the first WHIR round is an affine translate of a linear subspace, so [BCH+25, Theorem 4.5] yields that if the tuple of codewords is not jointly close to that code then the number of $\mu \in \mathbb{F}_{\mathrm{ext}}$ for which the batched codeword is nevertheless δ_0-close is at most $N_{\mathsf{BCHKS}}(w_{\mathcal{T},\mathrm{stack}}, \rho, \delta_0)$. In the unique-decoding regime, the same batching step is controlled by the standard correlated-agreement bound, which gives the exceptional-set size $(w_{\mathcal{T},\mathrm{stack}} - 1)|\mathcal{L}|$. Thus the batching round contributes at most $\frac{N_{\mathsf{BCHKS}}(w_{\mathcal{T},\mathrm{stack}}, \rho, \delta_0)}{|\mathbb{F}_{\mathrm{ext}}|}$ in either regime. After fixing a good batching challenge, the remaining rounds are exactly a WHIR execution with the chosen proximity parameters in each round. Their round-by-round soundness is given by [ACFY24, Theorem 5.2]; in the list-decoding regime, the required mutual correlated agreement hypothesis is supplied by the formal Reed–Solomon powers-generator theorem [BCGM26, Theorem 9.2]. □

Theorem 4.2.5 (SWIRL IOP)**.** *The interactive oracle proof described in §3.8 has round-by-round soundness with error*

$$\varepsilon^{\mathsf{rbr}}_{\mathsf{SWIRL}} := \max\left\{\varepsilon_{\mathsf{logup}}, L_{\mathsf{PCS}} \cdot \varepsilon_{\mathsf{bc}}, L_{\mathsf{PCS}} \cdot \varepsilon_{\mathsf{stack}}, \varepsilon_{\mathsf{bWHIR}}\right\}. \tag{4.8}$$

Moreover it has round-by-round knowledge soundness with the same error.

Proof. Theorems 4.2.2 and 4.2.3 are also stated for a fixed tuple of trace polynomials. In the full protocol, the first round of the final WHIR proximity proof only implies that the committed oracle is within distance δ_0 of one of at most L_{PCS} candidate tuples of stacked polynomials, equivalently one of at most L_{PCS} candidate tuples of trace polynomials. For each such tuple, the same Schwartz–Zippel arguments used in Theorems 4.2.2 and 4.2.3 apply, and the verifier

accepts if any tuple in the list explains the prover's answers. Therefore these two error terms also acquire a factor of L_{PCS} by a union bound over the list-decoding output. The WHIR term $\varepsilon_{\mathsf{bWHIR}}$ already incorporates the list-decoding regime via Theorem 4.2.4. □

Corollary 4.2.6 (Fiat–Shamir security of SWIRL)**.** *Let Π_{SWIRL} denote the public-coin IOP of Theorem 4.2.5. Let $\gamma_{\mathsf{SWIRL}}(x)$ and $q_{\mathsf{WHIR}}(x)$ denote, respectively, the proof length and verifier query complexity of Π_{SWIRL} on input x. Let $Q_{\mathsf{RO}} \in \mathbb{N}$ be a bound on the total number of random-oracle queries made by an adversary against the BCS-transformed protocol, and let*

$$\mathcal{H} : \{0,1\}^* \to \{0,1\}^{\kappa_{\mathsf{fs}}}$$

be a random oracle.

Then the non-interactive protocol of §3.8, obtained from Π_{SWIRL} by the BCS / Fiat–Shamir transform, has adaptive soundness error and adaptive knowledge error

$$\varepsilon^{\mathsf{fs}}_{\mathsf{SWIRL}}(x; Q_{\mathsf{RO}}, \kappa_{\mathsf{fs}}) = Q_{\mathsf{RO}} \cdot \varepsilon^{\mathsf{rbr}}_{\mathsf{SWIRL}} + \frac{3(Q^2_{\mathsf{RO}} + 1)}{2^{\kappa_{\mathsf{fs}}}} \tag{4.9}$$

against classical adversaries making at most Q_{RO} queries to $\mathcal{H}$.

Moreover, the transformed protocol has adaptive soundness error and adaptive knowledge error

$$\varepsilon^{\mathsf{fs-q}}_{\mathsf{SWIRL}}(x; Q_{\mathsf{RO}}, \kappa_{\mathsf{fs}}) = \Theta\big(Q_{\mathsf{RO}} \cdot \varepsilon^{\mathsf{fs}}_{\mathsf{SWIRL}}(x; Q_{\mathsf{RO}}, \kappa_{\mathsf{fs}})\big)\,. \tag{4.10}$$

against quantum adversaries making at most $Q_{\mathsf{RO}} - O\big(q_{\mathsf{WHIR}}(x) \log \gamma_{\mathsf{SWIRL}}(x)\big)$ queries.

Proof. By Theorem 4.2.5, the IOP Π_{SWIRL} has round-by-round soundness and round-by-round knowledge soundness error at most $\varepsilon^{\mathsf{rbr}}_{\mathsf{SWIRL}}$. Applying the standard BCS transformation theorem for public-coin IOPs [BSCS16, CMS19, Blo23] with proof length $\gamma_{\mathsf{SWIRL}}(x)$, verifier query complexity $q_{\mathsf{WHIR}}(x)$, random-oracle query budget Q_{RO}, and output length κ_{FS} gives adaptive soundness error

$$Q_{\mathsf{RO}} \cdot \varepsilon^{\mathsf{rbr}}_{\mathsf{SWIRL}} + \frac{3(Q^2_{\mathsf{RO}} + 1)}{2^{\kappa_{\mathsf{FS}}}}$$

and the same bound for adaptive knowledge error against classical adversaries. The same theorem gives the stated quantum-random-oracle bound against $Q_{\mathsf{RO}} - O(q_{\mathsf{WHIR}}(x) \log \gamma_{\mathsf{SWIRL}}(x))$ query quantum adversaries. □

References

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed–solomon proximity testing with super-fast verification. Cryptology ePrint Archive, Paper 2024/1586, 2024.

[BCGM26] Sarah Bordage, Alessandro Chiesa, Ziyi Guan, and Ignacio Manzur. All polynomial generators preserve distance with mutual correlated agreement. Cryptology ePrint Archive, 2026.

[BCH+25] Eli Ben-Sasson, Dan Carmon, Ulrich Haböck, Swastik Kopparty, and Shubhangi Saraf. On proximity gaps for reed-solomon codes. Electronic Colloquium on Computational Complexity, Report 169, 2025.

[BCI+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. Cryptology ePrint Archive, Paper 2020/654, 2020.

[BDT24] Suyash Bagad, Yuval Domb, and Justin Thaler. The Sum-Check Protocol over Fields of Small Characteristic. *IACR Cryptol. ePrint Arch.*, 2024:1046, 2024.

[Blo23] Block, Alexander R. and Garreta, Albert and Katz, Jonathan and Thaler, Justin and Tiwari, Pratyush Ranjan and Zając, Michał. Fiat-shamir security of fri and related snarks. In *Advances in Cryptology – ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4–8, 2023, Proceedings, Part II*, page 3–40, Berlin, Heidelberg, 2023. Springer-Verlag.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 31–60, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 1082–1090, New York, NY, USA, 2019. Association for Computing Machinery.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography*, pages 1–29, Cham, 2019. Springer International Publishing.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4), September 2015.

[Gru24] Angus Gruen. Some improvements for the PIOP for ZeroCheck. Cryptology ePrint Archive, Paper 2024/108, 2024.

[Hab22] Ulrich Haböck. A summary on the FRI low degree test. Cryptology ePrint Archive, Paper 2022/1216, 2022.

[Hab24] Ulrich Haböck. Basefold in the list decoding regime. Cryptology ePrint Archive, Paper 2024/1571, 2024.

[Irr25] Irreducible Team. Binius Blueprint: Batch Evaluation. https://web.archive.org/web/20250217104020/https://www.binius.xyz/blueprint/cryptography/evaluation/, 2025. No longer available.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, October 1992.

[Ope25a] OpenVM Contributors. On the Soundness of Interactions via LogUp. https://github.com/openvm-org/stark-backend/blob/main/docs/Soundness_of_Interactions_via_LogUp.pdf, 2025.

[Ope25b] OpenVM Contributors. OpenVM Whitepaper. https://openvm.dev/whitepaper.pdf, 2025.

[PH23] Shahar Papini and Ulrich Haböck. Improving logarithmic derivative lookups using GKR. Cryptology ePrint Archive, Paper 2023/1284, 2023.

[Val24] Valida Team. Valida. https://github.com/valida-xyz/valida, 2024.

[ZCF23] Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. Cryptology ePrint Archive, Paper 2023/1705, 2023.