Doubly Efficient Interactive Proofs over Infinite and Non-Commutative Rings

Eduardo Soria-Vazquez

Technology Innovation Institute, UAE.
ORCID: 0000-0002-4882-0230
eduardo.soria-vazquez@tii.ae

Abstract. We introduce the first proof system for layered arithmetic circuits over an arbitrary ring R that is (possibly) non-commutative and (possibly) infinite, while only requiring black-box access to its arithmetic and a subset $A \subseteq R$. Our construction only requires limited commutativity and regularity properties from A, similar to recent work on efficient information theoretic multi-party computation over non-commutative rings by Escudero and Soria-Vazquez (*CRYPTO 2021*), but furthermore covering infinite rings.

We achieve our results through a generalization of GKR-style interactive proofs (Goldwasser, Kalai and Rothblum, *Journal of the ACM*, 2015). When A is a subset of the center of R, generalizations of the sum-check protocol and other building blocks are not too problematic. The case when the elements of A only commute with each other, on the other hand, introduces a series of challenges. In order to overcome those, we need to introduce a new definition of polynomial ring over a non-commutative ring, the notion of *left* (and *right*) multi-linear extensions, modify the layer consistency equation and adapt the sum-check protocol.

Despite these changes, our results are compatible with recent developments such as linear time provers. Moreover, for certain rings our construction achieves provers that run in *sublinear* time in the circuit size. We obtain such result both for known cases, such as matrix and polynomial rings, as well as new ones, such as for some rings resulting from Clifford algebras. Besides efficiency improvements in computation and/or round complexity for several instantiations, the core conclusion of our results is that state of the art doubly efficient interactive proofs do not require much algebraic structure. This enables *exact* rather than *approximate* computation over infinite rings as well as "agile" proof systems, where the black-box choice of the underlying ring can be easily switched through the software life cycle.

1 Introduction

Interactive proofs (IPs) are a natural extension of the standard notion of a mathematical proof, where the *verifier* checking a proof is allowed to interrogate the *prover* who is providing it. They were introduced by Goldwasser, Micali and Rackoff [GMR89] in the 1980s and they soon made a huge impact in complexity theory. IPs have also been influential to practical proof systems, for which a lot of progress took place during the last decade. Usually, in those schemes, the prover tries to convince a verifier about the correctness of the evaluation of a circuit consisting of addition and multiplication gates. Moreover, the arithmetic of this circuit is often over a finite field, no matter how well represented under these constraints is the original computation whose correctness is being checked.

In 2008, Goldwasser, Kalai and Rothblum (GKR) presented the first *doubly-efficient* interactive proof [GKR15], where the prover is only required to perform a polynomial amount of work in the size of the (layered, over a finite field) arithmetic circuit and the verifier only needs to be quasi-linear in the same parameter. The prover's effort was later improved to quasi-linear [CMT12] and finally linear [XZZ$^+$19] for the same family of circuits in 2019. Recently, the restriction to layered circuits was removed [ZLW$^+$21] without affecting the linear complexity of the prover and only a slight increase in the verifier's work for non-layered circuits. In this work we are interested in a different kind of generalization of the GKR protocol. Namely, we set out to answer the following question:

> "Let C be a layered arithmetic circuit over a ring R. What algebraic properties does R need to satisfy in order to construct a doubly-efficient IP for C's correct evaluation, without emulating R's arithmetic?"

The most relevant part of our quest is that of avoiding the emulation of R's arithmetic, which we refer to as being *black-box* over R. We answer this question in a partial but constructive way by providing a doubly-efficient IP for rings R that are possibly *non-commutative* as well as *infinite*.

The *black-box* nature of our constructions has a theoretical interest, in the tradition of finding lower bounds and reducing (cryptographic) assumptions. Namely, it helps us understand what are the *minimum* algebraic properties that need to be assumed for proof systems and their underlying techniques to go through, and how does this affect their complexity. Whereas this path has been more explored in the context of Multi-Party Computation (MPC, see e.g. [CFIK03, CDI$^+$13, ACD$^+$19, DLS20, ES21] just to name a few), it has been strangely overlooked in the context of proof and argument systems, with notable exceptions [AIK10, HR18, CCKP19, GNS21, BCFK21, BCS21]. The main take-away of our work is that, when it comes to GKR-style protocols and their complexity, the algebraic properties of the ring do not matter much as long as it contains a big enough *set* with "good enough" regular and commutative properties. Interestingly, since infinite rings are allowed, this is a superset of the rings for which we know how to build efficient information-theoretic MPC protocols in a black-box manner [ES21].

Besides the theoretical aspects of our work, we expect its generality to find applications in practice. Practically relevant infinite rings (such as the integers) and fields (such as rational or real numbers) as well as non-commutative rings (such as matrices and quaternions) did not fit previous systems. Their arithmetic had to be emulated (at best) or approximated (at worst) when compiled into circuits over either finite fields or finite commutative rings [CCKP19]. Avoiding this compilation step can bring improvements in several fronts. First of all, removing this stage simplifies the practitioners' work, who can now be *agile* with respect to the choice of rings that are more commonplace than finite fields. If, after deployment, they need to provide a new proof system with a different underlying arithmetic, they could simply change the underlying data type that represents the ring, rather than having to develop an ad-hoc compiler. Moreover, working natively over such data types (algebraic structures) allows them to easily use existing software libraries for those, since their arithmetic does not need to be compiled into circuits. This, in turn, results in circuits with significantly less gates, which can ultimately result in better concrete efficiency in terms of computation and round complexity. Finally, the soundness error of our black-box IP can also benefit from working over these rings.

Related work. In [AIK10] Applebaum, Ishai and Kushilevitz show how to construct a verifiable computation protocol out of message authentication codes (MACs) and randomized encodings (REs). For their construction to be a *proof* rather than an *argument* system, it would need to use information-theoretic MACs and statistically secure REs. It is a longstanding open problem whether such statistical REs could efficiently support layered arithmetic circuits over the non-commutative and infinite rings that we support, or even finite fields. In particular, such REs would imply efficient constant-round statistically secure multi-party computation protocols for such circuits, which in turn solves an open problem about locally decodable codes of quasi-polynomial parameters [IK04].

In 2013, Meir [Mei13] demonstrated how the IP = PSPACE result can be proven using error-correcting codes (ECCs) that are more general than low degree polynomials. Along the way, the sum-check protocol is generalized to work with tensor products of linear ECCs. While that work is also interested in reducing algebraic assumptions, the results and ECCs are defined over finite fields. Whereas Meir's goal is to provide a new proof for a complexity theory result, we want to expand the amount of rings that one can use in a black-box way for GKR-style protocols and we care about concrete efficiency.

1.1 Technical overview

The GKR protocol [GKR15] is a doubly-efficient interactive proof for the evaluation of a layered arithmetic circuit, which consists of addition and multiplication gates of fan-in two. Parties move from the output (0-th layer) to the input layer (D-th layer) one layer at a time. Each gate in the i-th layer is supposed to take inputs from two wires in layer $i+1$, and so the output wires of the i-th layer gates are checked to be consistent with the ones in the preceding layer. Let $V^{(i)} : \{0,1\}^{s_i} \to \mathbb{F}$ be the function that maps the string x to the value of the x-th wire in layer i. Thus, layer i has (up to) 2^{s_i} wires. Furthermore, let $\mathtt{add}^{(i+1)} : \{0,1\}^{s_i} \times \{0,1\}^{s_{i+1}} \times \{0,1\}^{s_{i+1}} \to \{0,1\}$ be the function satisfying $\mathtt{add}^{(i+1)}(z,x,y) = 1$ if the z-th wire on layer i is the addition of the x-th and y-th wires in layer $i+1$, otherwise $\mathtt{add}^{(i+1)}(z,x,y) = 0$. Define $\mathtt{mult}^{(i+1)}$ analogously. If we use $\hat{f} \in \mathbb{F}[\vec{\mathtt{X}}]$ to denote a (low degree) multivariate polynomial such that for all $a \in \{0,1\}^s$, $\hat{f}(a) = f(a)$, we can express layer consistency as follows:

$$\hat{V}^{(i)}(\vec{\mathtt{Z}}) = \sum_{x,y \in \{0,1\}^{s_{i+1}}} \Big(\widehat{\mathtt{mult}}^{(i+1)}(\vec{\mathtt{Z}},x,y) \cdot \big(\hat{V}^{(i+1)}(x) \cdot \hat{V}^{(i+1)}(y)\big) + \\ + \widehat{\mathtt{add}}^{(i+1)}(\vec{\mathtt{Z}},x,y) \cdot \big(\hat{V}^{(i+1)}(x) + \hat{V}^{(i+1)}(y)\big)\Big). \tag{1}$$

The advantage of using the polynomial extensions $\hat{V}^{(i)}, \hat{V}^{(i+1)}, \widehat{\mathtt{mult}}^{(i+1)}$ and $\widehat{\mathtt{add}}^{(i+1)}$ is that the previous equation can be easily checked using the sum-check protocol [LFKN92]. Originally, as well as for most of its subsequent literature, the GKR protocol only worked for circuits over finite fields. Chen et al. [CCKP19] showed how to extend this result to finite commutative rings as long as the points used to define the polynomial extensions and the random challenges from the verifier belong to a set $A = \{a_1, \dots, a_n\}$ where $\forall i \neq j, a_i - a_j$ is not a zero divisor. In our work, we denote such A a *regular difference* set.

As we realized, it turns out that removing the finiteness assumption from [CCKP19] does not introduce any additional problems. Even if R is infinite, we only need a finite regular difference set A. From a practical point of view, one could envision different mechanisms to deal with issues such as sending messages of unknown length (for example, using the first bit of every message block to denote whether the message has been fully sent) or e.g. irrational numbers (the same way we write π rather than $3.14159\dots$).

On the other hand, when R is not commutative, we are presented with several issues. First, the definition of a polynomial ring with coefficients in R is not straightforward. One easily finds obstacles related to whether polynomial evaluation is a ring homomorphism (i.e., whether $f(a) + g(a) = (f+g)(a)$ and $f(a) \cdot g(a) = (f \cdot g)(a)$) or other crucial results, such as Euclidean division or bounding the number of roots of a polynomial. Nevertheless, if we further restrict the regular difference set A to be contained in the center of the ring (i.e., $\forall r \in R, a \in A, a \cdot r = r \cdot a$), then Equation (1) (and multi-linear extensions, the sum-check protocol, etc.) behave as we would expect. In this scenario, we use the most common definitions for polynomial over non-commutative rings (Definition 10, the same as in [QBC13, ES21]). We discuss this case in Section 4.

The most challenging part of our work comes from relaxing the commutativity requirement on A, so that rather than $A \subset Z(R)$, we only ask that $\forall a_i, a_j \in A, a_i \cdot a_j = a_j \cdot a_i$. This was also the most difficult family of rings in [ES21], where Escudero and Soria-Vazquez showed how to build efficient information-theoretic MPC protocols with black-box access to such a ring[1]. Employing the same polynomial ring definition as in that work fails in our context. This poses the question of whether there are inherently more *algebraic* limitations for doubly-efficient IPs than there are for information theoretic MPC, potentially ruling out these "less commutative" rings where $A \not\subset Z(R)$. Fortunately, we overcome most problems by putting forward a new polynomial ring definition (Definition 13) in Section 3, the notion of *sandwich* (and *toast*) polynomials (Section 3.1) and reworking many basic algebraic results related to these new polynomials. In Section 3.2 we show that there is no unique notion of multi-linear extension (MLE) in this setting, so we have to define both *left* and *right* MLEs. This part of our work is lengthier, but it is insightful and it highly simplifies every following step towards our doubly-efficient IPs. Equipped with these results, in Section 5 we show how to modify the layer consistency equation so that it becomes a sandwich polynomial. We need to do this carefully, so that it is a toast polynomial on every indeterminate. Finally, we need to provide a new sum-check protocol for this layer consistency equation (Section 5.3), which we show how to run in linear time by the prover in Section 5.4.

1.2 Instantiations and applications

The main practical appeal of our constructions, being *black-box* on the choice of the underlying ring, is their versatility. They provide a single system that can fit many instances and applications at a reasonable cost, as they preserve the state of the art complexity of their finite field counterparts. Practitioners need to worry less: First, they can work with existing software libraries for whichever ring R becomes relevant in their application. Second, expressing computation directly over R, rather than emulating its arithmetic, simplifies readability and any compilation steps into the arithmetic circuit. Besides this flexibility, we highlight a few interesting instantiations for R and their associated applications.

Improved efficiency, sublinear provers. In Section 4.1 we show how, for rings R which are free modules of rank d over a smaller ring S, our doubly-efficient IP over R can have reduced costs for the prover. We can even have the prover to run in *sublinear* time in the size of the layered arithmetic circuit over R. Examples include matrix and polynomial rings, as well as (large enough) Clifford algebras.

Geometric algebra. Clifford algebras have many applications in physics and engineering [BS10], such as computer vision [RAS08]. One of the most familiar Clifford algebras are quaternion rings. Quaternions are particularly advantageous for describing rotations in a three-dimensional space, both in terms of efficiency and in order to avoid (potentially devastating) gimbal locks, i.e. the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism. Gimbal locks happen when

[1] In fact, in [ES21] they only show how to work with *finite* rings in that family. An example interesting ring in this setting is $\mathcal{M}_{n\times n}(\mathbb{F}_2)$, which has $\mathbb{F}_{2^n}$ as a subfield.

the axes of two of the three gimbals end up parallel to each other, restricting the system to rotations in a two-dimensional space. In robotics, they are sometimes referred to as "wrist flips" or "wrist singularities". Besides avoiding gimbal lock, quaternions are also more compact and numerically stable than rotation matrices. This results in their application to the domains of computer graphics, robotics and aerospace, including satellite navigation. *Dual* quaternions are an even better tool to describe rigid body dynamics, which is interesting in the same settings.

Exact computation. This application is enabled since we can compute black-box over infinite rings. We can, for example, work directly with the field of (computable) real numbers, rather than using fixed/floating point arithmetic to represent elements of the field. The latter only provide *approximate* computation, and errors are likely to occur due to the lack of precision. Perhaps contrary to popular belief, the magnitude of these errors can be quite significant. Let us illustrate this with an example from [GNSW07], the logistic map defined by $x_0 = 0.5$ and $x_{n+1} = 3.999 \cdot x_n \cdot (1 - x_n)$. Using IEEE 754 floating point numbers, the 10000th element of the sequence is 0.780738. Using exact computation, on the other hand, the output (rounded to six digits of precision *after performing the computation*) is 0.354494. We recommend [GNSW07] to any reader interested in an overview of the different approaches to computing on real numbers.

Approximating infinite arithmetic requires not only costly subcircuits for rounding and a careful analysis as in [CCKP19]. It also requires to consider the use of words and adjusting the circuit (or the underlying finite ring) depending on the bit length of circuit inputs[2]. Since the amount of multiplications depends on the word size and the cost of each multiplication on the underlying algebraic structure, this becomes an optimization problem for practitioners. What is more, fixed/floating point numbers do not have a ring structure, in particular they lack the distributive property. Hence, the optimization algorithm providing the circuit description becomes a new attack vector in the broader system, allowing whoever provides it to bias results towards their interest.

Due to the above and further issues, critical applications might benefit from performing exact rather than approximate computation over the reals, for which there was no proof system before our work. For example, implementations of differential privacy have been found to be insecure due to the use of a floating point representation when using the Laplace [Mir12] or Gaussian mechanism [JMRO22]. Even for non-critical applications, the enormous simplification in terms of circuit description (which for exact arithmetic is independent of the inputs) is, in the last instance, of practical interest.

Symbolic computation. Many software products for symbolic computation such as Magma, Maple, Sage, Wolfram, etc. run cloud services, where the clients need not run these computations locally. Since the appeal of these systems is the many algebraic structures they support, it is interesting for any proof system on top of them to be as generic as ours on the choice of the underlying ring.

2 Preliminaries

Notation. We use $[i, j]$, where $i < j$, to represent the set of positive integers $\{i, i+1, \ldots, j\}$, and simply $[n]$ to represent $\{1, 2, \ldots, n\}$. Sometimes, we may use arrows to denote vectors, e.g. $\vec{b} = (b_1, \ldots, b_n)$. For a "sub-interval" of the elements of a vector, we might denote use $\vec{b}_{[i,j]} = (b_i, \ldots, b_j)$.

[2] That is, if that information (e.g. the number of decimals for real numbers) can be upper-bounded in advance. In particular, arbitrary computation on irrational numbers such as e and π cannot be emulated over a fixed finite field in a exact way.

2.1 Interactive proofs and the GKR protocol

In order to capture more naturally our results, we present these definitions in terms of the prover $\mathcal{P}$ trying to convince a verifier $\mathcal{V}$ that the application of an arithmetic circuit C over a ring R on some input `inp` results on a specific output `out`, where inputs and outputs are elements of R.

Definition 1. *Let C be an arithmetic circuit over a ring R. A pair of interactive machines $\langle \mathcal{P}, \mathcal{V} \rangle$ is an ϵ-sound interactive proof (IP) for C if, on a claimed output* `out` *by $\mathcal{P}$:*

- *Completeness: For every* `inp` *s.t. $C($*`inp`*$) =$* `out`*, it holds that $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle($*`inp`*$) =$* `accept`*$] = 1$.*
- *ϵ-Soundness: For any* `inp` *s.t. $C($*`inp`*$) \neq$* `out`*, and any $\mathcal{P}^*$, it holds that $\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle($*`inp`*$) =$* `accept`*$] \leq \epsilon$.*

We say that an interactive proof has the succinct property if the running time of $\mathcal{V}$ and the total communication between $\mathcal{P}$ and $\mathcal{V}$ is $\mathsf{poly}(|x|, \log(|C|))$.

The sum-check protocol Given an n-variate polynomial $f : \mathbb{F}^n \to \mathbb{F}$, the sum-check protocol [LFKN92] allows a verifier to outsource the computation of $\sum_{\vec{b} \in \{0,1\}^n} f(\vec{b})$ to a prover. If the verifier was to do this on their own, it would take them $O(2^n)$ time. Let d be an upper bound on the degree of each individual variable of f. The sum-check protocol is an n-round interactive proof for this task, where both the proof size and the verifier's work is $O(n \cdot d)$ and the soundness error is $\epsilon = n \cdot d \cdot |\mathbb{F}|^{-1}$. Its full description appears Figure 1.

Algorithm `Sumcheck_Finite_Field`(f)

Input: $f \in \mathbb{F}[\mathtt{X}_1, \ldots, \mathtt{X}_n]$.
Output: Accept if $S = \sum_{\vec{b} \in \{0,1\}^n} f(\vec{b})$, or reject otherwise.

- In the first round, $\mathcal{P}$ sends the univariate polynomial given by:
$$f_1(\mathtt{X}_1) = \sum_{b_2, \ldots, b_n \in \{0,1\}} f(\mathtt{X}_1, b_2, \ldots, b_n),$$
$\mathcal{V}$ checks whether $f_1(0) + f_1(1) = S$. If so, $\mathcal{V}$ chooses a random element $r_1 \in \mathbb{F}$ and sends it to $\mathcal{P}$.
- For rounds $2 \leq i \leq n-1$, $\mathcal{P}$ sends the univariate polynomial given by:
$$f_i(\mathtt{X}_i) = \sum_{b_{i+1}, \ldots, b_n \in \{0,1\}} f(r_1, \ldots, r_{i-1}, \mathtt{X}_i, b_{i+1}, \ldots, b_n),$$
$\mathcal{V}$ checks whether $f_i(0) + f_i(1) = f_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in \mathbb{F}$ and sends it to $\mathcal{P}$.
- In the n-th round, $\mathcal{P}$ sends the univariate polynomial given by:
$$f_n(\mathtt{X}_n) = f(r_1, \ldots, r_{n-1}, \mathtt{X}_n),$$
$\mathcal{V}$ checks whether $f_n(0) + f_n(1) = f_{n-1}(r_{n-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_n \in \mathbb{F}$. Given oracle access to the evaluation of $f(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ at $(r_1, \ldots, r_n)$, $\mathcal{V}$ accepts if and only if $f_n(r_n) = f(r_1, \ldots, r_{n-1}, r_n)$.

Fig. 1. Sum-check protocol over a finite field.

The GKR protocol The basics of how circuits and wire values are represented in the GKR protocol have been explained at the beginning of Section 1.1. Here we give a bit more details about how Equation (1) is combined with the sum-check protocol and how to progress from the output to the input layer. $\mathcal{P}$ first sends the claimed output to $\mathcal{V}$, consisting of 2^{s_0} different values. $\mathcal{V}$ defines a multi-linear polynomial $\hat{V}^{(0)} : \mathbb{F}^{s_0} \to \mathbb{F}^{s_0}$ which extends $V^{(0)}$, samples a random $\gamma \in \mathbb{F}^{s_0}$ and sends it to $\mathcal{P}$. Both parties then evaluate $\hat{V}^{(0)}(\gamma)$ and run a sum-check protocol on Eq. (1) for $i = 0$ and evaluated at γ. Let $f_i(\vec{\mathtt{Z}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}})$ be the function such that Eq. (1) is $\hat{V}^{(i)}(\vec{\mathtt{Z}}) = \sum_{x,y \in \{0,1\}^{s_{i+1}}} f_i(\vec{\mathtt{Z}}, x, y)$. At the end of the protocol, $\mathcal{V}$ needs to compute $f_0(\gamma, \chi, \psi)$, where $\chi, \psi \in \mathbb{F}^{s_1}$ are two random values produced throughout the sum-check execution. Whereas $\mathcal{V}$ can evaluate $\widehat{\mathtt{add}}^{(1)}(\gamma, \chi, \psi)$ and $\widehat{\mathtt{mult}}^{(1)}(\gamma, \chi, \psi)$ on their own, it has to ask $\mathcal{P}$ for $\hat{V}^{(1)}(\chi), \hat{V}^{(1)}(\psi)$, since those evaluations require the knowledge of the wire values on layer 1. This way, a claim about the output layer has been reduced to two claims about layer one, $\hat{V}^{(1)}(\chi)$ and $\hat{V}^{(1)}(\psi)$. $\mathcal{V}$ and $\mathcal{P}$ could run one sum-check protocol for each of those claims using Eq. (1) for $i = 1$, but the number of sum-check executions would eventually become exponential in the depth of the circuit by following such a route. In order to avoid this, both claims are combined into a single claim. The full description of the GKR protocol appears in Figure 2. Below, we state the complexity and soundness of its current most efficient version.

Theorem 1 ([XZZ+19]). *Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be a depth-D layered arithmetic circuit. The GKR protocol is an interactive proof for C with soundness error $O(D \log |C| / |\mathbb{F}|)$. Its communication and round complexity is $O(D \log |C|)$. The prover complexity is $O(|C|)$ and the verifier complexity is $O(n + k + D \log |C| + T)$, where T is the optimal time to evaluate every $\widehat{\mathtt{add}}^{(i)}, \widehat{\mathtt{mult}}^{(i)}$ wiring predicate. For log-space uniform circuits, $T = \mathsf{poly}\log(|C|)$.*

2.2 Algebraic background

We turn now to recall some useful results from ring theory. Unless otherwise specified, whenever we talk about a ring R we mean a ring with identity $1 \neq 0$, for which we assume neither commutativity nor finiteness. This generality requires to be careful about the potential lack of commutativity and the presence of zero divisors and non-invertible (regular) elements. We start by recapping some basic definitions and results in non-commutative algebra.

Definition 2. *Let R be a ring. An element $a \in R$ is a unit if there exists $b \in R$ such that $a \cdot b = b \cdot a = 1$. The set of all units is denoted by R^*.*

Definition 3. *An element $a \in R \setminus \{0\}$ is a left (resp. right) zero divisor if $\exists\, b \in R \setminus \{0\}$ such that $a \cdot b = 0$ (resp. $b \cdot a = 0$).*

Lemma 1. *Let R be a finite ring. Then all elements of $R \setminus \{0\}$ are either a unit or a zero divisor.*

Sets of elements whose pairwise differences are either regular or invertible will play a crucial role in our constructions.

Definition 4. *Let $A = \{a_1, \ldots, a_n\} \subset R$. We say that A is a regular difference set, or R.D. set for short, if $\forall i \neq j, a_i - a_j \in R$ is not a zero divisor. We define the* regularity constant *of R to be the maximum size of an R.D. set in R.*

Algorithm `GKR_Finite_Field`

Let $\mathbb{F}$ be a finite field. Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be a layered arithmetic circuit over $\mathbb{F}$ with depth D. Without loss of generality, we assume that n and k are powers of 2.
Input: Circuit input `inp` and claimed output `out`.
Output: Accept or reject.

- Compute $\hat{V}^{(0)}(\mathtt{X})$ as the multi-linear extension (MLE) of `out`. $\mathcal{V}$ chooses a random $\gamma \in \mathbb{F}^{s_0}$ and sends it to $\mathcal{P}$. Both parties compute $\hat{V}^{(0)}(\gamma)$.
- Run a sum-check protocol on Equation (1) for $i = 0$ and evaluated at γ. Let $\chi^{(0)}, \psi^{(0)} \in \mathbb{F}^{s_1}$ denote the challenge vectors corresponding to the $\vec{\mathtt{X}}$ and $\vec{\mathtt{Y}}$ variables within that execution. $\mathcal{P}$ sends $\hat{V}^{(1)}(\chi^{(0)})$ and $\hat{V}^{(1)}(\psi^{(0)})$ to $\mathcal{V}$.
- $\mathcal{V}$ queries their oracles for $\widehat{\mathtt{mult}}^{(1)}(\gamma, \chi^{(0)}, \psi^{(0)})$ and $\widehat{\mathtt{add}}^{(1)}(\gamma, \chi^{(0)}, \psi^{(0)})$, so as to check that $\widehat{\mathtt{add}}^{(1)}(\gamma, \chi^{(0)}, \psi^{(0)}) \cdot (\hat{V}^{(1)}(\chi^{(0)}) + \hat{V}^{(1)}(\psi^{(0)})) + \widehat{\mathtt{mult}}^{(1)}(\gamma, \chi^{(0)}, \psi^{(0)}) \cdot \hat{V}^{(1)}(\chi^{(0)}) \cdot \hat{V}^{(1)}(\psi^{(0)})$ equals the last message of the sumcheck execution.
- For circuit layers $i = 1, \ldots, D-1$:
  - $\mathcal{V}$ samples $\alpha^{(i)}, \beta^{(i)} \in \mathbb{F}$ and sends them to $\mathcal{P}$. $\mathcal{P}$ and $\mathcal{V}$ run a sumcheck protocol on:

$$\alpha^{(i)}\hat{V}^{(i)}(\chi^{(i-1)}) + \beta^{(i)}\hat{V}^{(i)}(\psi^{(i-1)}) = \sum_{x,y \in \{0,1\}^{s_{i+1}}}$$
$$\Big(\quad (\alpha^{(i)} + \beta^{(i)}) \cdot \widehat{\mathtt{mult}}^{(i+1)}(\chi^{(i-1)}, x, y) \cdot \hat{V}^{(i+1)}(x) \cdot \hat{V}^{(i+1)}(y) +$$
$$+ (\alpha^{(i)} + \beta^{(i)}) \cdot \widehat{\mathtt{add}}^{(i+1)}(\psi^{(i-1)}, x, y) \cdot (\hat{V}^{(i+1)}(x) + \hat{V}^{(i+1)}(y)) \quad \Big)$$

  Let $\chi^{(i)}, \psi^{(i)}$ denote the challenge vectors corresponding to the $\vec{\mathtt{X}}$, and $\vec{\mathtt{Y}}$ variables within that execution. At the end of the protocol, $\mathcal{P}$ sends $\hat{V}^{(i+1)}(\chi^{(i)})$ and $\hat{V}^{(i+1)}(\psi^{(i)})$ to $\mathcal{V}$, so that $\mathcal{V}$ can check the validity of the last message in the sumcheck execution. If the check passes, they proceed to the $(i+1)$-th layer, otherwise, $\mathcal{V}$ outputs reject and aborts.
- At the input layer D, $\mathcal{V}$ has received two claims $\hat{V}^{(D)}(\chi^{(D-1)})$ and $\hat{V}^{(D)}(\psi^{(D-1)})$. $\mathcal{V}$ queries the evaluation oracles of $\hat{V}^{(D)}$ at $\chi^{(D-1)}$ and $\psi^{(D-1)}$ in order to check that they equal the sumcheck claims. If they do, $\mathcal{V}$ outputs accept, otherwise, $\mathcal{V}$ outputs reject.

Fig. 2. The GKR protocol over a finite field.

Definition 5. *Let $A = \{a_1, \ldots, a_n\} \subset R$. We say that A is an exceptional set if $\forall i \neq j, a_i - a_j \in R^*$. We define the* Lenstra constant *of R to be the maximum size of an exceptional set in R.*

When R is a finite ring, as a consequence of Lemma 1, every R.D. set is an exceptional set. Nevertheless, there are infinite rings, such as $\mathbb{Z}$, which have an infinite regularity constant whereas their Lenstra constant is 2.

Besides "how regular" or "how invertible" are certain subsets of ring elements, we might also be interested in "how commutative" they are. The following definitions and results look into this.

Definition 6. *The center of a ring R, denoted by $Z(R)$ consists of the elements $a \in Z(R)$ such that $\forall b \in R$, $a \cdot b = b \cdot a$.*

Definition 7 ([QBC13]). *Let $A = \{a_1, \ldots, a_n\} \subset R$. We say that A is a commutative set if $\forall a_i, a_j \in A, a_i \cdot a_j = a_j \cdot a_i$.*

Definition 8. *Let R be a ring and $A \subset R$. The centralizer of the set A in R is:*

$$C_R(A) = \{b \in R : b \cdot a = a \cdot b, \forall a \in A\}.$$

Centralizers are also referred to as *commutants* sometimes in the literature. Note that $C_R(R) = Z(R)$. We will often refer to the following, trivial lemma.

Lemma 2. *Let R be a ring and $A \subset R$ a commutative set. Then $C_R(A) \supseteq (A \cup Z(R))$. Furthermore, if $A \subseteq Z(R)$, then $C_R(A) = R$.*

3 Polynomials over non-commutative rings

There is no unique choice for how to define a polynomial ring with coefficients on a non-commutative ring R. Usually, as in [QBC13, ES21], univariate polynomials are defined in such a way that "the indeterminate commutes with coefficients", so as to *uniquely* express any polynomial $f \in R[\mathtt{X}]_{\leq d}$ as the formal sum $f(\mathtt{X}) = \sum_{i=0}^{d} f_i \mathtt{X}^i$, where $f_i \in R$. In the language of centralizers, this approach enforced $C_{R[\mathtt{X}]}(\{\mathtt{X}\}) = R[\mathtt{X}]$. In the multivariate case, one can choose whether to define the ring so that indeterminates commute with each other or not. For rings where $A \subseteq Z(R)$, we will stick with the former case and refer to it as a *ring of non-commutative polynomials*. Before giving a formal definition of this approach, we recall the definition of a monoid.

Definition 9. *A set $\mathbf{M}$ equipped with a binary operation $\cdot : \mathbf{M} \times \mathbf{M} \to \mathbf{M}$ is a monoid if the operation is associative ($\forall a, b, c \in \mathbf{M}, (a \cdot b) \cdot c = a \cdot (b \cdot c)$) and there is an identity element ($\exists e \in \mathbf{M} :\ \forall a \in \mathbf{M}, a \cdot e = e \cdot a = a$).*

Definition 10. *Let $(R, +, *)$ be a ring and let $\Sigma = \{\mathtt{X}_1, \ldots, \mathtt{X}_n\}$. Let Σ^* be the free commutative monoid generated by Σ, i.e. the monoid whose binary operation is the concatenation of finite strings and the letters of the alphabet Σ commute with each other. The* ring of non-commutative polynomials *$R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ is the monoid ring of Σ^* over R. Explicitly, $a \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ is of the form $a = \sum_{m \in \Sigma^*} a_m m$, where $a_m \in R$ and there is only a finite amount of $a_m \neq 0$. Addition and multiplication are defined as follows:*

– *Addition:* $a + b = \sum_{m \in \Sigma^*} (a_m + b_m) m$

– *Multiplication:* $a \cdot b = \sum_{m_1, m_2 \in \Sigma^*} (a_{m_1} \cdot b_{m_2}) m_1 m_2$.

Furthermore, for any set $S \subseteq R$*, we define* $S[\mathtt{X}_1, \ldots, \mathtt{X}_n]_{\leq d}$ *to be the subset of polynomials in* $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ *of degree at most* d *whose coefficients belong to* S*.*

The previous definition has many advantages, but it requires to be careful about polynomial evaluation, which we next show to be a ring homomorphism if and only if the evaluation points belong to $Z(R)$. For example, consider polynomials $f(\mathtt{X}) = f_0 + f_1\mathtt{X}$ and $g(\mathtt{X}) = g_1\mathtt{X} + g_2\mathtt{X}^2$. We would have that $h(\mathtt{X}) = f(\mathtt{X}) \cdot g(\mathtt{X}) = f_0 g_1 \mathtt{X} + (f_0 g_2 + f_1 g_1)\mathtt{X}^2 + f_1 g_2 \mathtt{X}^3$. Unless α commutes with g_1 and g_2, this results in $h(\alpha) \neq f(\alpha) \cdot g(\alpha)$.

Lemma 3. *Let* $A = \{\alpha_i\}_{i=1}^n \subset R$ *be a commutative set and let* $\alpha = (\alpha_1, \ldots, \alpha_n)$*. Denote by* $\mathtt{Ev}_\alpha : R[\mathtt{X}_1, \ldots, \mathtt{X}_n] \to R$ *the map that takes a polynomial* $f \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ *to its evaluation*[3] *at* α*, by replacing each appearance of* $\mathtt{X}_i$ *with* α_i *and applying the product operation of* R*. Then:*

1. $\forall f, g \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$, $\mathtt{Ev}_\alpha(f) + \mathtt{Ev}_\alpha(g) = \mathtt{Ev}_\alpha(f + g)$.
2. $\mathtt{Ev}_\alpha(f) \cdot \mathtt{Ev}_\alpha(g) = \mathtt{Ev}_\alpha(f \cdot g)$ *holds* $\forall f, g \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ *if and only if* $A \subseteq Z(R)$*.*

Proof. We first observe that the condition of $A = \{\alpha_i\}_{i=1}^n \subset R$ being a commutative set is necessary for the map to be well defined, since we did not define an order for the variables $\{\mathtt{X}_i\}_{i=1}^n \subset \Sigma$ and the symbols of the alphabet commute with each other. Let $f = \sum_{m \in \Sigma^*} f_m m$, $g = \sum_{m \in \Sigma^*} g_m m$ as per Definition 3. Note that $\mathtt{Ev}_\alpha(\sum_{m \in \Sigma^*} a_m m) = \sum_{m \in \Sigma^*} a_m \cdot \mathtt{Ev}_\alpha(m)$, from which we prove:

1. $\mathtt{Ev}_\alpha(f + g) = \mathtt{Ev}_\alpha(\sum_{m \in \Sigma^*} (f_m + g_m) m) = \sum_{m \in \Sigma^*} (f_m + g_m) \cdot \mathtt{Ev}_\alpha(m) = \sum_{m \in \Sigma^*} f_m \cdot \mathtt{Ev}_\alpha(m) + \sum_{m \in \Sigma^*} g_m \cdot \mathtt{Ev}_\alpha(m) = \mathtt{Ev}_\alpha(f) + \mathtt{Ev}_\alpha(g)$.
2. First, assume $A \subseteq Z(R)$. Then $\mathtt{Ev}_\alpha(f \cdot g) = \mathtt{Ev}_\alpha(\sum_{m_1, m_2 \in \Sigma^*} (f_{m_1} \cdot g_{m_2}) m_1 m_2) = \sum_{m_1, m_2 \in \Sigma^*} (f_{m_1} \cdot g_{m_2} \cdot \mathtt{Ev}_\alpha(m_1 m_2)) = \sum_{m_1, m_2 \in \Sigma^*} (f_{m_1} \cdot g_{m_2} \cdot \mathtt{Ev}_\alpha(m_1) \cdot \mathtt{Ev}_\alpha(m_2))$. Since $A \subseteq Z(R)$ and $Z(R)$ is a ring, we have that $\mathtt{Ev}_\alpha(m_1), \mathtt{Ev}_\alpha(m_2) \in Z(R)$. Hence we can conclude that $\sum_{m_1, m_2 \in \Sigma^*} (f_{m_1} \cdot g_{m_2} \cdot \mathtt{Ev}_\alpha(m_1) \cdot \mathtt{Ev}_\alpha(m_2)) = \sum_{m_1, m_2 \in \Sigma^*} (f_{m_1} \cdot \mathtt{Ev}_\alpha(m_1) \cdot g_{m_2} \cdot \mathtt{Ev}_\alpha(m_2)) = (\sum_{m_1 \in \Sigma^*} f_{m_1} \cdot \mathtt{Ev}_\alpha(m_1)) \cdot (\sum_{m_2 \in \Sigma^*} g_{m_2} \cdot \mathtt{Ev}_\alpha(m_2)) = \mathtt{Ev}_\alpha(f) \cdot \mathtt{Ev}_\alpha(g)$.
   For the other implication, we will prove the contrapositive. Assume $\exists \alpha_i \in A$ such that $\alpha_i \notin Z(R)$, we will show that $\exists f, g \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ such that $\mathtt{Ev}_\alpha(f) \cdot \mathtt{Ev}_\alpha(g) \neq \mathtt{Ev}_\alpha(f \cdot g)$. Since $\alpha_i \notin Z(R)$, then $C_R(\{\alpha_i\}) \neq R$. Let $f_1, g_1 \in R \setminus C_R(\{\alpha_i\})$. Consider the polynomials $f = f_1 \mathtt{X}_i$ and $g = g_1 \mathtt{X}_i$, we have that $\mathtt{Ev}_\alpha(f) \cdot \mathtt{Ev}_\alpha(g) = f_1 \cdot \alpha_i \cdot g_1 \cdot \alpha_i \neq f_1 \cdot g_1 \cdot \alpha_i^2 = \mathtt{Ev}_\alpha(f \cdot g)$. ∎

A different way to define the polynomial ring arises from treating the indeterminate $\mathtt{X}$ as a formal, non-commuting symbol. Polynomial addition works as usual, whereas the product looks more similar to string concatenation (e.g. for the same $f(\mathtt{X})$ and $g(\mathtt{X})$ of the example in the previous paragraph, $f(\mathtt{X}) \cdot g(\mathtt{X}) = f_0 \cdot g_1 \mathtt{X} + f_1 \mathtt{X} g_1 \mathtt{X} + f_0 \cdot g_2 \mathtt{X}^2 + f_1 \mathtt{X} g_2 \mathtt{X}^2$). Looking at the problem again in terms of centralizers, in this approach we enforce $C_{R[\mathtt{X}]}(\{\mathtt{X}\}) = \emptyset$. The advantage of this strategy[4] is that polynomial evaluation at any $\alpha \in R$ becomes a ring homomorphism, in contrast with Definition 10 where that is only true if $\alpha \in Z(R)$ (Lemma 3). On the other hand, not being able to simplify polynomial expressions as in Definition 10 not only results in lengthier polynomials, but it also eliminates the possibility to prove many useful, typical results about polynomials. We will refer to this construction as the *ring of totally non-commutative polynomials.*

[3] Throughout the text, we implicitly refer to $\mathtt{Ev}_\alpha(f)$ whenever we write either $f(\alpha)$ or $f(\alpha_1, \ldots, \alpha_n)$ and $f \in R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$.

[4] This advantage would also apply to a slight variant where the ring would be constructed so that $C_{R[\mathtt{X}]}(\mathtt{X}) = Z(R)$.

Definition 11. *Let $(R, +, \odot)$ be a ring and let $\Sigma = R \cup \{\mathtt{X}_1, \ldots, \mathtt{X}_n\}$. Let $*$ denote the string concatenation operation. Let $\mathbf{M}$ be the monoid generated by Σ according to the non-commutative binary operation $\cdot : \mathbf{M} \times \mathbf{M} \to \mathbf{M}$, which we restrict to finite strings and where:*

$$r \cdot s = \begin{cases} r \odot s, & \text{if } r, s \in R \\ r * s, & \text{if } (r \in R \wedge s \in \{\mathtt{X}_i\}_{i=1}^n) \vee (r \in \{\mathtt{X}_i\}_{i=1}^n \wedge s \in R) \vee (r, s \in \{\mathtt{X}_i\}_{i=1}^n) \end{cases} \tag{2}$$

The ring of totally non-commutative polynomials $R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!]$ *consists of elements $a \in R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!]$ of the form $a = \sum_{m \in \mathbf{M}} a_m \cdot m$, where $a_m \in R$ and there is only a finite amount of $a_m \neq 0$. Addition (inherited from R) and multiplication (inherited from $\mathbf{M}$) are defined as follows:*

- *Addition: $a + b = \sum_{m \in \mathbf{M}} (a_m + b_m) \cdot m$*
- *Multiplication: $a \cdot b = \sum_{m_1, m_2 \in \mathbf{M}} a_{m_1} \cdot m_1 \cdot b_{m_2} \cdot m_2$.*

Lemma 4. *$R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!]$ is a ring.*

Proof. $(R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!], +)$ being an abelian group follows from inspection. Next, $(R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!], \cdot)$ is a monoid, since $m_1 \cdot b_{m_2} \cdot m_2 \in \mathbf{M}$ and only a finite number of a_{m_1} are non-zero. The distributive property follows from making string concatenation (the operation $*$) distribute over addition. ■

Note 1. In terms of notation, the ring of totally non-commutative polynomials $R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!]$ should not be confused with a formal power series ring over R.

Note 2. The only difference between $(\mathbf{M}, \cdot)$ in Definition 11 and the free (non-commutative) monoid over Σ is that, in $\mathbf{M}$, strings containing sub-strings of the form $\mathtt{X}_i * a * b * \mathtt{X}_j$ do not exist, since in $\mathbf{M}$ those are "simplified" to $\mathtt{X}_i * c * \mathtt{X}_j$ where $c = a \odot b$.

Note 3. $R[\![\mathtt{X}_1, \ldots, \mathtt{X}_n]\!]$ allows for limited simplifications in polynomial expressions. Namely, those inherited from the associative and distributive properties if we go from the "outside" to the "inside" of a monomial. For example, it is true that $\mathtt{X}r\mathtt{X}^3 + \mathtt{X}s\mathtt{X}^3 = \mathtt{X}(r + s)\mathtt{X}^3$, but for any $m_1 \neq m_2 \in \mathbf{M}$, we cannot simplify beyond $\mathtt{X}r\mathtt{X}^3 m_1 + \mathtt{X}s\mathtt{X}^3 m_2 = \mathtt{X}(r\mathtt{X}^3 m_1 + s\mathtt{X}^3 m_2)$, since $m_1 \neq m_2$ is "blocking" any simplification from the right end (besides any common factor at the right end of m_1, m_2).

The rationale behind defining the ring of non-commutative polynomials as in Definition 10, as they do in e.g. [QBC13, ES21], is that in those works unique polynomial interpolation is the most crucial property. In our case, the most important requirement is that polynomial evaluation is a ring homomorphism, so that we can meaningfully apply a sumcheck protocol to the layer consistency Equation (1). Unfortunately, the approach from Definition 11, where $C_{R[\mathtt{X}]}(\mathtt{X}) = \emptyset$, does not serve us either, since the layer consistency equation is *cubic*. As it will become clearer later on, when proving results about Euclidean division (Theorem 2) and Lemma 7, we cannot bound the number of roots of the product of three polynomials by simply following that route. Such bound is in turn necessary to establish the soundness error of our new interactive proofs.

Due to the above, we introduce a novel polynomial ring definition, somewhere between Definitions 10 and 11. We define the *polynomial ring with evaluation set A*, $R_A[\mathtt{X}]$, by taking into account the specific set of points $A \subset R$ on which polynomials will be ever evaluated. Rather than constructing the ring so that $C_{R[\mathtt{X}_1,\ldots,\mathtt{X}_n]}(\{\mathtt{X}_i\}_{i=1}^n) = R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ (as in Defn. 10) or such that $C_{R[\![\mathtt{X}_1,\ldots,\mathtt{X}_n]\!]}(\{\mathtt{X}_i\}_{i=1}^n) = \emptyset$ (as in Defn. 11), we will enforce $C_{R_A[\mathtt{X}_1,\ldots,\mathtt{X}_n]}(\{\mathtt{X}_i\}_{i=1}^n) = C_R(A) \cup \{\mathtt{X}_i\}_{i=1}^n$.

Hence, in $R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$, indeterminates commute with each other and "an indeterminate commutes with a coefficient $c \in R$ if and only if $c \in C_R(A)$". Formally, we construct $R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$ by taking the quotient of the ring of totally non-commutative polynomials $R[\![\mathtt{X}_1, \dots, \mathtt{X}_n]\!]$ with a "commutator ideal" I_A that enforces our precise commutativity requirements.

Definition 12. *Let $R[\![\mathtt{X}_1, \dots, \mathtt{X}_n]\!]$ and let $A \subset R$. For $i, j = 1, \dots n$, let $S_{i,j} = \{\mathtt{X}_i\mathtt{X}_j - \mathtt{X}_j\mathtt{X}_i\}$ and $S_i = \{\mathtt{X}_i c - c\mathtt{X}_i : c \in C_R(A)\}$. The two-sided ideal generated by $\bigcup_{i=1}^{n} \bigcup_{j=1}^{i-1} S_{i,j} \cup S_i$ is the* commutator ideal of A, *which we denote by I_A.*

For our specific goals, we will impose that the set A is commutative (see Definition 7). This is to ensure that $C_R(A) \supseteq (A \cup Z(R))$.

Definition 13. *Let $A \subset R$ be a commutative set and let I_A be the commutator ideal it defines. We define the* ring of polynomials with evaluation set A *to be $R_A[\mathtt{X}_1, \dots, \mathtt{X}_n] = R[\![\mathtt{X}_1, \dots, \mathtt{X}_n]\!]/I_A$. For any set $S \subseteq R$, we define $S_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$ to be the subset of polynomials in $R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$ whose coefficients belong to S.*

Claim. Let $S \subseteq R$. Any $f \in S_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$ can be uniquely expressed as

$$f = \sum_{k=1}^{s} \Big(\prod_{\ell=1}^{m} \big(r_{k,\ell} \prod_{i=1}^{n} (\mathtt{X}_i^{d_{i,\ell}}) \big) \Big),$$

where $r_{k,\ell} \in S \cup \{1\}, d_{i,\ell} \in \mathbb{Z}$ and m is the maximum length of any monomial in f. We consider $\mathtt{X}_i^0$ to be the empty string, for any $i = 1, \dots, n$.

By Lemma 2, when $A \subseteq Z(R)$ we have that $C_R(A) = R$, in which case the commutator ideal enforces $C_{R_A[\mathtt{X}_1,\dots,\mathtt{X}_n]}(\{\mathtt{X}_i\}_{i=1}^n) = R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$. Intuitively, in this situation, the polynomial ring $R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$ behaves *the same way* as $R[\mathtt{X}_1, \dots, \mathtt{X}_n]$ in Definition 10: $R[\mathtt{X}_1, \dots, \mathtt{X}_n]$ also satisfies that $C_{R[\mathtt{X}_1,\dots,\mathtt{X}_n]}(\{\mathtt{X}_i\}_{i=1}^n) = R[\mathtt{X}_1, \dots, \mathtt{X}_n]$ and, when the evaluation points are in $A \subseteq Z(R)$, evaluating polynomials from $R[\mathtt{X}_1, \dots, \mathtt{X}_n]$ is a ring homomorphism too (Lemma 3). The two following results show in more detail how both definitions are interchangeable for our purposes when $A \subseteq Z(R)$.

Proposition 1. *Let A be a commutative set and let $S \subseteq C_R(A)$. Let ϕ be:*

$$\phi : S_A[\mathtt{X}_1, \dots, \mathtt{X}_n] \to R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$$
$$f = \sum_{k=1}^{s} \Big(\prod_{\ell=1}^{m} \big(r_{k,\ell} \prod_{i=1}^{n} (\mathtt{X}_i^{d_{i,\ell}}) \big) \Big) \mapsto \phi(f) = \sum_{k=1}^{s} \Big(\prod_{\ell=1}^{m} (r_{k,\ell}) \prod_{i=1}^{n} (\mathtt{X}_i^{\sum_{\ell=1}^{m} d_{i,\ell}}) \Big),$$

the map which uses the commutator ideal I_A to "regroup indeterminates $\mathtt{X}_i$ on the right hand side, and the product of the coefficients in S on the left hand side". Let $\psi : \phi(S_A[\mathtt{X}_1, \dots, \mathtt{X}_n]) \to R[\mathtt{X}_1, \dots, \mathtt{X}_n]$ be the map that reinterprets the resulting polynomial as an element of $R[\mathtt{X}_1, \dots, \mathtt{X}_n]$. $\forall f \in S_A[\mathtt{X}_1, \dots, \mathtt{X}_n]$, let $F = \psi \circ \phi(f)$. Then, $\forall a \in A^n$, it holds that $F(a) = f(a)$.

Proof. First, observe that $\phi(f)(a) = f(a)$, since the elements of both $\{\mathtt{X}_i\}_{i=1}^n$ and A commute with $\{r_{k,\ell}\}_{k\in[s],\ell\in[m]}$ (because $r_{k,\ell} \in (S \cup \{1\}) \subset C_R(A)$). Second, $\psi \circ \phi(f)(a) = \phi(f)(a)$, since the evaluation map for a polynomial in $R[\mathtt{X}_1, \dots, \mathtt{X}_n]$ or a polynomial in $\phi(S_A[\mathtt{X}_1, \dots, \mathtt{X}_n])$ has the exact same description. ■

Corollary 1. *Let $A \subseteq Z(R)$ and let ϕ and ψ be the maps from Proposition 1. Then, $\forall f \in R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n], \forall a \in A^n$, it holds that $\psi \circ \phi(f)(a) = f(a)$. In other words, when $A \subseteq Z(R)$, reinterpreting polynomials from $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ as polynomials from $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ does not affect the results of polynomial evaluation.*

Proof. If $A \subset Z(R)$, then $C_R(A) = R$ (Lemma 2). Take $S = R$ in Prop. 1. ■

For the sake of generality, we will state and prove our results using the more general ring $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ from Definition 13. Nevertheless, when $A \subset Z(R)$, it is conceptually simpler to treat polynomials as elements from $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ (Definition 10). For basic algebraic results that we will present in Section 3.1, such as Euclidean division or the number of roots of a polynomial, simplified statements and proofs for $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ can be found in e.g. [ES21]. For more involved results, such as the generalization of the sumcheck protocol to non-commutative rings, multi-linear extensions and ultimately our protocol for doubly-efficient interactive proofs over general rings, we will provide remarks along the way stating how they are simplified when $A \subset Z(R)$ and how this affects their complexities.

3.1 Sandwich polynomials

In the previous block of results, we have seen that when $A \subset Z(R)$, the ring $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ from Definition 13 behaves *the same way* as $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ in Definition 10. It is when A is merely a commutative set –and hence $C_{R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]}(\mathtt{X}_i) \supseteq (A \cup Z(R) \cup \{\mathtt{X}_j\}_{j=1}^n)$– that our new Definition 13 will be necessary to enable our GKR-style protocol over a ring $R \supset A$. This extends the applicability of our results, since for example the ring $\mathcal{M}_{n\times n}(\mathbb{Z}/2^k\mathbb{Z})$ contains a commutative exceptional set of size 2^m [ES21, Proposition 3], whereas the biggest regular difference set contained in $Z(\mathcal{M}_{n\times n}(\mathbb{Z}/2^k\mathbb{Z}))$ has size two.

Our protocols will be concerned with a particular *subset* of the polynomials in $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]$, concretely the ones for which monomials have a single coefficient, possibly "surrounded" by indeterminates on both sides. We will refer to these as *sandwich polynomials*, metaphorically thinking of the indeterminates as bread and the coefficient as the content[5]. The goal of this subsection is to generalize the Schwartz-Zippel lemma to these polynomials, to the extent that it is possible.

Definition 14 (Sandwich polynomials). *Let A be a commutative subset of a ring R and let $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ be the ring of polynomials with evaluation set A. Let $i = (i_1, \ldots, i_n)$, $j = (j_1, \ldots, j_n)$. We define the set of* sandwich polynomials *over R with left-degree at most d' and right-degree at most d to be:*

$$R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]_{\le d', \le d} = \{f(\mathtt{X}_1, \ldots, \mathtt{X}_n) = \sum_{i \in [0,d']^n, j \in [0,d]^n} \mathtt{X}_n^{i_n} \cdot \ldots \cdot \mathtt{X}_1^{i_1} f_{i,j} \mathtt{X}_1^{j_1} \cdot \ldots \cdot \mathtt{X}_n^{j_n} \mid f_{i,j} \in R\}$$

The subset of polynomials with right-degree exactly d, $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]_{\le d', d} \subset R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]_{\le d', \le d}$, *is given by further imposing that, for every $\mathtt{X}_k$, the polynomial must have at least one monomial of right-degree d in $\mathtt{X}_k$. Formally: $\forall k \in [n]\ \exists i \in [0, d']^n, j_1, \ldots, j_{k-1}, j_{k+1}, \ldots, j_n \in [0, d]$ such that*

[5] The reader might find funny to think about multiplication as "stacking sandwiches" and addition as putting sandwiches next to each other. The commutativity of indeterminates with elements in $C_R(A)$, simplifications enabled by the distributive property and other results in this section provide some (metaphorical) food for thought!

$f_{i,(j_1,\dots,j_{k-1},d,j_{k+1},\dots,j_n)} \neq 0$. *The subset of polynomials with left-degree* exactly d', $R_A[\mathtt{X}_1,\dots,\mathtt{X}_n]_{d',\leq d}$, *is defined analogously.*

Furthermore, for any set $S \subseteq R$, we define $S_A[\mathtt{X}_1,\dots,\mathtt{X}_n]_{\leq d',\leq d}$ as the subset of polynomials in $R_A[\mathtt{X}_1,\dots,\mathtt{X}_n]_{\leq d',\leq d}$ whose coefficients $f_{i,j}$ all belong to S. Polynomials of exact degrees are defined as in the previous paragraph.

Definition 15 (Toast polynomials). *Let $\vec{\mathtt{X}} = (\mathtt{X}_1,\dots,\mathtt{X}_n)$. A sandwich polynomial f is a* left *(resp.* right*)* toast polynomial *if it is of right (resp. left) degree zero, i.e. $f \in R_A[\vec{\mathtt{X}}]_{\leq d',0}$ (resp. $f \in R_A[\vec{\mathtt{X}}]_{0,\leq d}$). If we do not want to specify the position of the indeterminate, we may simply refer to it as a* toast polynomial.

In the previous definitions, it is important to note that a polynomial in $R_A[\mathtt{X}_1,\dots,\mathtt{X}_n]_{\leq d',\leq d}$ has at most $((d'+1)\cdot(d+1))^n$ monomials, i.e. for fixed powers $i \in [0,d']^n, j \in [0,d]^n$, an expression of the form $\sum_\ell \mathtt{X}_n^{i_n} \cdot \ldots \cdot \mathtt{X}_1^{i_1} f_{i,j}^{(\ell)} \mathtt{X}_1^{j_1} \cdot \ldots \cdot \mathtt{X}_n^{j_n}$ is simplified into $\mathtt{X}_n^{i_n} \cdot \ldots \cdot \mathtt{X}_1^{i_1} f_{i,j} \mathtt{X}_1^{j_1} \cdot \ldots \cdot \mathtt{X}_n^{j_n}$, where $f_{i,j} = \sum_\ell f_{i,j}^{(\ell)}$. Furthermore, when we talk about polynomials of *exact* right degree d or left degree d', we assume that all possible simplifications have taken place. In particular[6], for $f \in R_A[\mathtt{X}]_{\leq d',d}$, we assume that $f_{i,d}$ is not simplified away with terms of the form $\mathtt{X}^{i+k} f_{i+k,d-k} \mathtt{X}^{d-k}$, where $k \in \{1,\dots,d\}$, when $f_{i,d}, f_{i+k,d-k} \in C_R(A)$.

Lemma 5. *Let $\vec{\mathtt{X}} = (\mathtt{X}_1,\dots,\mathtt{X}_n)$. For $\ell = 1,\dots m$, let $f^{(\ell)} \in R_A[\vec{\mathtt{X}}]_{\leq d'_f,\leq d_f}$, $a^{(\ell)} \in C_R(A)[\vec{\mathtt{X}}]_{\leq d'_a,\leq d_a}$ and $b^{(\ell)} \in C_R(A)[\vec{\mathtt{X}}]_{\leq d'_b,\leq d_b}$. Let $g = \sum_{\ell=1}^m a^{(\ell)} f^{(\ell)} b^{(\ell)}$. Then:*

$$g \in R_A[\vec{\mathtt{X}}]_{\leq (d'_a+d_a+d'_f),\leq(d'_b+d_b+d_f)}.$$

Proof. Let $i^{(f)} = (i_1^{(f)},\dots i_n^{(f)}), j^{(f)} = (j_1^{(f)},\dots j_n^{(f)})$ and so on for $i^{(a)}, j^{(a)}$ and $i^{(b)}, j^{(b)}$. For $\ell = 1,\dots,m$, let:

$$f^{(\ell)}(\mathtt{X}_1,\dots,\mathtt{X}_n) = \sum_{i^{(f)}\in[0,d'_f]^n} \sum_{j^{(f)}\in[0,d_f]^n} \mathtt{X}_n^{i_n^{(f)}} \cdot \ldots \cdot \mathtt{X}_1^{i_1^{(f)}} f_{i^{(f)},j^{(f)}}^{(\ell)} \mathtt{X}_1^{j_1^{(f)}} \cdot \ldots \cdot \mathtt{X}_n^{j_n^{(f)}}$$

$$a^{(\ell)}(\mathtt{X}_1,\dots,\mathtt{X}_n) = \sum_{i^{(a)}\in[0,d'_a]^n} \sum_{j^{(a)}\in[0,d_a]^n} \mathtt{X}_n^{i_n^{(a)}} \cdot \ldots \cdot \mathtt{X}_1^{i_1^{(a)}} a_{i^{(a)},j^{(a)}}^{(\ell)} \mathtt{X}_1^{j_1^{(a)}} \cdot \ldots \cdot \mathtt{X}_n^{j_n^{(a)}}$$

$$b^{(\ell)}(\mathtt{X}_1,\dots,\mathtt{X}_n) = \sum_{i^{(b)}\in[0,d'_b]^n} \sum_{j^{(b)}\in[0,d_b]^n} \mathtt{X}_n^{i_n^{(b)}} \cdot \ldots \cdot \mathtt{X}_1^{i_1^{(b)}} b_{i^{(b)},j^{(b)}}^{(\ell)} \mathtt{X}_1^{j_1^{(b)}} \cdot \ldots \cdot \mathtt{X}_n^{j_n^{(b)}}$$

By reasoning about the commutator ideal I_A (Definition 12), we can rewrite

$$a^{(\ell)}(\mathtt{X}_1,\dots,\mathtt{X}_n) = \sum_{i^{(a)}\in[0,d'_a]^n} \sum_{j^{(a)}\in[0,d_a]^n} a_{i^{(a)},j^{(a)}}^{(\ell)} \mathtt{X}_1^{i_1^{(a)}+j_1^{(a)}} \cdot \ldots \cdot \mathtt{X}_n^{i_n^{(a)}+j_n^{(a)}}$$

$$b^{(\ell)}(\mathtt{X}_1,\dots,\mathtt{X}_n) = \sum_{i^{(b)}\in[0,d'_b]^n} \sum_{j^{(b)}\in[0,d_b]^n} \mathtt{X}_n^{i_n^{(b)}+j_n^{(b)}} \cdot \ldots \cdot \mathtt{X}_1^{i_1^{(b)}+j_1^{(b)}} b_{i^{(b)},j^{(b)}}^{(\ell)}.$$

[6] We give this example in the univariate case in order to avoid heavier notation.

Let $\sum_{i\in I, j\in J}$ replace the notation $\sum_{i^{(a)}\in[0,d'_a]^n}\sum_{j^{(a)}\in[0,d_a]^n}\sum_{i^{(f)}\in[0,d'_f]^n}\sum_{j^{(f)}\in[0,d_f]^n}\sum_{i^{(b)}\in[0,d'_b]^n}\sum_{j^{(b)}\in[0,d_b]^n}$. Then, by substituting on $g = \sum_{\ell=1}^{m} a^{(\ell)} f^{(\ell)} b^{(\ell)}$, we obtain:

$$g = \sum_{\ell=1}^{m}\sum_{i\in I, j\in J} a^{(\ell)}_{i^{(a)},j^{(a)}} \mathtt{X}_n^{i_n^{(a)}+j_n^{(a)}+i_n^{(f)}}\cdot\ldots\cdot \mathtt{X}_1^{i_1^{(a)}+j_1^{(a)}+i_1^{(f)}} f^{(\ell)}_{i^{(f)},j^{(f)}} \mathtt{X}_1^{i_1^{(b)}+j_1^{(b)}+i_1^{(f)}}\cdot\ldots\cdot$$
$$\cdot \mathtt{X}_n^{i_n^{(b)}+j_n^{(b)}+i_n^{(f)}} b^{(\ell)}_{i^{(b)},j^{(b)}} = \sum_{\ell=1}^{m}\sum_{i\in I, j\in J} \mathtt{X}_n^{i_n^{(a)}+j_n^{(a)}+i_n^{(f)}}\cdot\ldots\cdot \mathtt{X}_1^{i_1^{(a)}+j_1^{(a)}+i_1^{(f)}} a^{(\ell)}_{i^{(a)},j^{(a)}}\cdot$$
$$f^{(\ell)}_{i^{(f)},j^{(f)}}\cdot b^{(\ell)}_{i^{(b)},j^{(b)}} \mathtt{X}_1^{i_1^{(b)}+j_1^{(b)}+i_1^{(f)}}\cdot\ldots\cdot \mathtt{X}_n^{i_n^{(b)}+j_n^{(b)}+i_n^{(f)}}$$

Where in the first equality we used the fact that $\mathtt{X}_\alpha\cdot\mathtt{X}_\beta = \mathtt{X}_\beta\cdot\mathtt{X}_\alpha$ for all $\alpha,\beta\in[n]$ and, for the second one, the fact that $a^{(\ell)}_{i^{(a)},j^{(a)}}$ and $b^{(\ell)}_{i^{(b)},j^{(b)}}$ are in $C_R(A)$ and hence commute with the indeterminates. The resulting expression is clearly an element of $R_A[\mathtt{X}_1,\ldots,\mathtt{X}_n]_{\le(d'_a+d_a+d'_f),\le(d'_b+d_b+d_f)}$. ■

Lemma 6. *Let $f\in R_A[\mathtt{X}_1,\ldots,\mathtt{X}_n]_{\le d',\le d}$ and let $a_\ell\in A$. Then, $\forall\ell\in\{1,\ldots,n\}$:*

$$f(\mathtt{X}_1,\ldots\mathtt{X}_{\ell-1},a_\ell,\mathtt{X}_{\ell+1},\ldots,\mathtt{X}_n)\in R_A[\mathtt{X}_1,\ldots\mathtt{X}_{\ell-1},\mathtt{X}_{\ell+1},\ldots,\mathtt{X}_n]_{\le d',\le d}.$$

Proof. Since A is a commutative set (required by the definition of $R_A[\mathtt{X}_1,\ldots,\mathtt{X}_n]$), we have that $A\subset C_R(A)$. Making use of the commutator ideal, we "push" the powers of a_ℓ to the middle of the polynomial, obtaining:

$$f(\mathtt{X}_1,\ldots\mathtt{X}_{\ell-1},a_\ell,\mathtt{X}_{\ell+1},\ldots,\mathtt{X}_n) =$$
$$= \sum_{i\in[0,d']^n, j\in[0,d]^n} \mathtt{X}_n^{i_n}\ldots\mathtt{X}_{\ell+1}^{i_{\ell+1}}\cdot\mathtt{X}_{\ell-1}^{i_{\ell-1}}\ldots\mathtt{X}_1^{i_1}\cdot a_\ell^{i_\ell}\cdot f_{i,j}\cdot a_\ell^{j_\ell}\cdot\mathtt{X}_1^{j_1}\ldots\mathtt{X}_{\ell-1}^{j_{\ell-1}}\cdot\mathtt{X}_{\ell+1}^{j_{\ell+1}}\ldots\mathtt{X}_n^{j_n}.$$ ■

The advantage of sandwich polynomials is that they can be divided by monic polynomials in $S_A[\mathtt{X}]$, where $S = C_R(A)$ (recall notation from Definition 13).

Theorem 2 (Euclidean division). *Let $f(\mathtt{X})\in R_A[\mathtt{X}]_{d',d}$ be a non-zero sandwich polynomial and let $g(\mathtt{X})\in C_R(A)_A[\mathtt{X}]_{0,m}$ be a monic polynomial[7]. There exist unique sandwich polynomials $q_\ell(\mathtt{X}), r_\ell(\mathtt{X})$ (resp. $q_r(\mathtt{X}), r_r(\mathtt{X})$) such that $f(\mathtt{X}) = q_\ell(\mathtt{X})\cdot g(\mathtt{X}) + r_\ell(\mathtt{X})$ (resp. $f(\mathtt{X}) = g(\mathtt{X})\cdot q_r(\mathtt{X}) + r_r(\mathtt{X})$), where $q_\ell(\mathtt{X})\in R_A[\mathtt{X}]_{\le d',\le d-m}$ and $r_\ell(\mathtt{X})\in R_A[\mathtt{X}]_{\le d',\le m-1}$ (resp. $q_r(\mathtt{X})\in R_A[\mathtt{X}]_{\le d-m,\le d}$, $r_r(\mathtt{X})\in R_A[\mathtt{X}]_{\le m-1,\le d}$).*

Proof. We only prove the result for $q_\ell(\mathtt{X})$ and $r_\ell(\mathtt{X})$. The result for $q_r(\mathtt{X})$ and $r_r(\mathtt{X})$ follows in a similar manner. We first prove the statement regarding the existence of $q_\ell(\mathtt{X})$ and $r_\ell(\mathtt{X})$ by induction on the right degree of f. Consider a degree-zero f. If $m=0$, then $g(\mathtt{X})=1$ and we may take $q_\ell(\mathtt{X}) = f(\mathtt{X})$ and $r_\ell(\mathtt{X}) = 0$. If $m\ge 1$, then we may take $q_\ell(\mathtt{X}) = 0$ and $r_\ell(\mathtt{X}) = f(\mathtt{X})$.

Suppose the statement is true for polynomials f of right degree D. We will now show that the statement also holds for f of right degree $D+1$. If $D+1<m$, we take $q_\ell(\mathtt{X}) = 0$ and $r_\ell(\mathtt{X}) = f(\mathtt{X})$. Otherwise, let $f(\mathtt{X}) = \sum_{i=0}^{d'}\sum_{j=0}^{D+1}\mathtt{X}^i f_{i,j}\mathtt{X}^j$.

[7] I.e. $g(\mathtt{X}) = \mathtt{X}^m + \sum_{\ell=0}^{m-1} g_\ell\mathtt{X}^\ell$. Note that since $g_\ell\in C_R(A)$ $\forall\ell\in[0,m-1]$, it is also true that $g(\mathtt{X})\in C_R(A)_A[\mathtt{X}]_{m,0}$, that $g(\mathtt{X})\in C_R(A)_A[\mathtt{X}]_{m-1,1}$, etc.

We define $\tilde{f}(\mathtt{X}) = f(\mathtt{X}) - \hat{q}(\mathtt{X}) \cdot g(\mathtt{X})$, where $\hat{q}(\mathtt{X}) = \sum_{i=0}^{d'} \mathtt{X}^i f_{i,D+1} \mathtt{X}^{D+1-m}$. As the right degree of $\tilde{f}$ is smaller or equal than D, by the induction hypothesis we have that $\tilde{f}(\mathtt{X}) = \tilde{q}_\ell(\mathtt{X}) \cdot g(\mathtt{X}) + \tilde{r}_\ell(\mathtt{X})$, where the right degree of $\tilde{r}_\ell(\mathtt{X})$ is strictly smaller than m. We can conclude the existence part of the proof by setting $q_\ell(\mathtt{X}) = \tilde{q}_\ell(\mathtt{X}) + \hat{q}(\mathtt{X})$ (which is a sandwich polynomial by virtue of Lemma 5, and it is in $R_A[\mathtt{X}]_{\leq d', \leq D+1-m}$) and $r_\ell(\mathtt{X}) = \tilde{r}_\ell(\mathtt{X}) \in R_A[\mathtt{X}]_{\leq d', \leq m-1}$, as shown by the following equalities:

$$f(\mathtt{X}) = \tilde{f}(\mathtt{X}) + \hat{q}(\mathtt{X}) \cdot g(\mathtt{X}) = \tilde{q}_\ell(\mathtt{X}) \cdot g(\mathtt{X}) + \tilde{r}_\ell(\mathtt{X}) + \hat{q}(\mathtt{X}) \cdot g(\mathtt{X}) = (\tilde{q}_\ell(\mathtt{X}) + \hat{q}(\mathtt{X})) \cdot g(\mathtt{X}) + \tilde{r}_\ell(\mathtt{X}).$$

Regarding uniqueness, suppose that $f(\mathtt{X}) = q_\ell(\mathtt{X}) \cdot g(\mathtt{X}) + r_\ell(\mathtt{X})$ and $f(\mathtt{X}) = q'_\ell(\mathtt{X}) \cdot g(\mathtt{X}) + r'_\ell(\mathtt{X})$, where $r_\ell(\mathtt{X}), r'_\ell(\mathtt{X}) \in R_A[\mathtt{X}]_{\leq d', \leq m-1}$. This implies that $(q'_\ell(\mathtt{X}) - q_\ell(\mathtt{X})) \cdot g(\mathtt{X}) = r_\ell(\mathtt{X}) - r'_\ell(\mathtt{X})$, but if $q_\ell(\mathtt{X}) \neq q'_\ell(\mathtt{X})$ we would have that the right degree of $(q'_\ell(\mathtt{X}) - q_\ell(\mathtt{X})) \cdot g(\mathtt{X})$ is at least m, which is a contradiction with the fact that $r_\ell(\mathtt{X}) - r'_\ell(\mathtt{X}) \in R_A[\mathtt{X}]_{\leq d', \leq m-1}$. ■

Given the previous theorem, we can now prove the following result with respect to the maximum number of roots of toast polynomials on their evaluation set A, when A is not only commutative but also regular difference (Definition 4).

Lemma 7. *Let A be a commutative, regular difference set of R and let $f \in R_A[\mathtt{X}]_{0,\leq d}$ (resp. $\tilde{f} \in R_A[\mathtt{X}]_{\leq d,0}$) be a non-zero toast polynomial. Then f (resp. $\tilde{f}$) has at most d roots in A.*

Proof. We reason by induction on the right-degree d of the non-zero polynomial f. The procedure is analogous for $\tilde{f} \in R_A[\mathtt{X}]_{\leq d,0}$, so we omit that part of the proof. The statement is clear when $d = 0$. Assume the result true for $d-1$ and let us look at $f \in R_A[\mathtt{X}]_{0,d}$. If f does not have any roots, or if it only has one root, then the result holds (since we have considered $d = 0$ as a separate case). Else, let $a, b \in A$ be two different roots of $f(\mathtt{X})$. As $g(\mathtt{X}) = \mathtt{X} - a$ is a monic polynomial in $C_R(A)_A[\mathtt{X}]$, by Theorem 2 there exist a sandwich polynomial $q(\mathtt{X}) \in R_A[\mathtt{X}]_{0,\leq d-1}$ and $r \in R$ (since the reminder has both left and right degree zero) such that $f(\mathtt{X}) = q(\mathtt{X}) \cdot g(\mathtt{X}) + r$.

Since we have that $0 = f(a) = q(a) \cdot g(a) + r = q(a) \cdot 0 + r$, we conclude that $r = 0$. From this, it follows that $0 = f(b) = q(b) \cdot (b - a)$. Since A is a regular difference set, $b - a$ is not a zero divisor, so it has to be that $q(b) = 0$. By the inductive hypothesis, $q(\mathtt{X}) \in R[\mathtt{X}]_{0,\leq d-1}$ has at most $d-1$ roots in A, so we can conclude that $f(\mathtt{X})$ has at most d roots in A. ■

Whereas, given Theorem 2 and Lemma 7, one could hope to be able to bound the number of roots of any polynomial $f \in R_A[\mathtt{X}]_{\leq d', \leq d}$, we were unable to prove such a result. This is due to the fact that the Euclidean division of sandwich polynomials by a polynomial $g_i(\mathtt{X}) = (X - \alpha_i)$, where α_i is a root of f, provides us with a remainder that is of degree zero *only on the side from which $g_i(\mathtt{X})$ is dividing.* If α_1 is a root of f, by calling Theorem 2 so that $g_1(\mathtt{X})$ "divides on the right", we can prove that $f(\mathtt{X}) = f_1(\mathtt{X})(\mathtt{X} - \alpha_1) + r_1(\mathtt{X})$, where $f_1 \in R_A[\mathtt{X}]_{\leq d', \leq d-1}, r_1 \in R_A[\mathtt{X}]_{\leq d',0}$. If we divide $r_1(\mathtt{X})$ by $g_1(\mathtt{X})$ on the left, we get to $f(\mathtt{X}) = f_1(\mathtt{X})(\mathtt{X} - \alpha_1) + (\mathtt{X} - \alpha_1) f_2(\mathtt{X})$, where $f_2 \in R_A[\mathtt{X}]_{\leq d'-1,0}$. The problem now is that, if $\alpha_2 \in A$ is another root, we find no way forward from the expression $0 = f(\alpha_2) = f_1(\alpha_2)(\alpha_2 - \alpha_1) + (\alpha_2 - \alpha_1) f_2(\alpha_2)$. Alternative strategies also beared no positive results. This important limitation will condition the generalization of almost every building block of our doubly-efficient IP over non-commutative rings when A is merely commutative, rather than $A \subseteq Z(R)$.

Lemma 8 is a generalization of the Schwartz-Zippel lemma to *toast* polynomials over possibly infinite, possibly non-commutative rings.

Lemma 8 (Schwartz-Zippel Lemma). *Let $A \subseteq R$ be a finite, commutative regular difference set. Let $\vec{\mathtt{X}} = (\mathtt{X}_1, \dots, \mathtt{X}_n)$ and let $f \in (R_A[\vec{\mathtt{X}}]_{\leq d,0} \cup R_A[\vec{\mathtt{X}}]_{0,\leq d})$ be a non-zero toast polynomial. Then:*

$$\Pr_{\vec{a} \leftarrow A^n}[f(\vec{a}) = 0] \leq \frac{n \cdot d}{|A|}$$

Proof. We prove by induction on the number of variables n. For simplicity, we will assume that $f \in R_A[\vec{\mathtt{X}}]_{\leq d,0}$, the reasoning is symmetric when $f \in R_A[\vec{\mathtt{X}}]_{0,\leq d}$. The case $n = 1$ follows from Lemma 7.

Assume the result holds for polynomials in $R_A[\mathtt{X}_1, \dots, \mathtt{X}_{n-1}]_{\leq d,0}$. For the n-variate case, given any $f \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_n]_{\leq d,0}$, we have that $f(\vec{\mathtt{X}}) = \sum_{\ell=0}^{d} \mathtt{X}_n^{\ell} \cdot g_\ell(\mathtt{X}_1, \dots, \mathtt{X}_{n-1})$, where $g_\ell \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_{n-1}]_{\leq d,0}$.

Let $\vec{a} = (a_1, \dots, a_n)$ and let $d_{\mathtt{X}_n}$ denote the degree of f in $\mathtt{X}_n$, i.e. the biggest $d_{\mathtt{X}_n} \leq d$ such that $g_{d_{\mathtt{X}_n}}$ is not the zero polynomial. Denote by E_1 the event $g_{d_{\mathtt{X}_n}}(\vec{a}) = 0$. Since it is a non-zero toast polynomial, by the induction hypothesis

$$\Pr_{(a_1, \dots, a_{n-1}) \leftarrow A^{n-1}}[E_1] \leq \frac{(n-1) \cdot d}{|A|}.$$

For the event $\neg E_1$, i.e. when $g_{d_{\mathtt{X}_n}}(\vec{a}) \neq 0$, let us define $f_{\neg E_1}(\mathtt{X}_n) = \sum_{\ell=0}^{d_{\mathtt{X}_n}} \mathtt{X}_n^{\ell} \cdot g_\ell(a_1, \dots, a_{n-1})$. Notice that $f_{\neg E_1} \in R_A[\mathtt{X}_n]_{\leq d_{\mathtt{X}_n},0}$ and $f_{\neg E_1}(a_n) = f(\vec{a})$. Then:

$$\Pr_{\vec{a} \leftarrow A}[f(\vec{a}) = 0 | \neg E_1] = \Pr_{\vec{a} \leftarrow A}[f_{\neg E_1}(a_n) = 0 | \neg E_1] \leq d_{\mathtt{X}_n}/|A|,$$

where the last inequality follows from Lemma 7. Given the previous bounds, we conclude our proof by substituting in the following series of inequalities:

$$\begin{aligned}\Pr[f(\vec{a}) = 0] &= \Pr[f(\vec{a}) = 0 | \neg E_1] \cdot \Pr[\neg E_1] + \Pr[f(\vec{a}) = 0 | E_1] \cdot \Pr[E_1] \\ &\leq \Pr[f(\vec{a}) = 0 | \neg E_1] + \Pr[E_1] \leq \frac{(n-1)d}{|A|} + \frac{d_{\mathtt{X}_n}}{|A|} \leq \frac{n \cdot d}{|A|}.\end{aligned}$$ ■

Lagrange interpolation for sets of points $(x_i, y_i) \in R^2$ can be computed, as long as all the x_i are part of the same commutative exceptional set $A \subset R$.

Proposition 2. *Let $A = \{x_1, \dots, x_{d+1}\} \subset R$ be a commutative exceptional set and let $B = \{y_1, \dots, y_{d+1}\} \subset R$. Then there exists a unique polynomial $f \in R_A[\mathtt{X}]_{0,\leq d}$ (resp. $g \in R_A[\mathtt{X}]_{\leq d,0}$) such that $f(x_i) = y_i$ (resp. $g(x_i) = y_i$) for $i = 1, \dots, d+1$. Furthermore, if $A \cup B$ constitutes a commutative set, $f = g$.*

Proof. Let $L_i(\mathtt{X}) = \prod_{j \neq i}(\mathtt{X} - x_j) \in A_A[\mathtt{X}]$. Observe that for all $j = 1, \dots, d+1$ it holds that $L_i(x_j) \in R^*$, since $(x_i - x_j) \in R^*$. It is easy to verify that the two following polynomials show the existence of solutions:

$$f(\mathtt{X}) = \sum_{i=1}^{d+1} y_i L_i(x_i)^{-1} L_i(\mathtt{X}); \qquad g(\mathtt{X}) = \sum_{i=1}^{d+1} L_i(\mathtt{X}) L_i(x_i)^{-1} y_i$$

The uniqueness of f (resp. g) is a consequence of Lemma 7. The fact that $f(X) = g(X)$ when $A \cup B$ constitutes a commutative set follows from inspection. ■

3.2 Multi-Linear Extensions over non-commutative rings

Multi-linear extensions were introduced in [BFL91] and extensively used in [CMT12] as an improvement over the low-degree extensions used in [GKR15]. Here, we generalize their definition to toast polynomials (Definition 15).

Lemma 9. *Let A be a regular difference, commutative set s.t. $\{0,1\} \subset A \subset R$. Given a function $V : \{0,1\}^m \to R$, there exist unique multilinear polynomials $\hat{V}_L \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]_{\leq 1,0}$ and $\hat{V}_R \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]_{0,\leq 1}$ extending V, i.e. $\hat{V}_L(a) = V(a) = \hat{V}_R(a)$ for all $a \in \{0,1\}^m$. We call $\hat{V}_L$ (resp. $\hat{V}_R$) the left (resp. right) multilinear extension of V, which we will abbreviate by LMLE (resp. RMLE).*

When $A \subset Z(R)$, or when $V : \{0,1\}^m \to C_R(A)$, it furthermore holds that $\hat{V}_L(\mathtt{X}_1, \dots, \mathtt{X}_m) = \hat{V}_R(\mathtt{X}_1, \dots, \mathtt{X}_m)$, in which case we will simply refer to the multilinear extension (MLE) of V and denote it by $\hat{V}(\mathtt{X}_1, \dots, \mathtt{X}_m)$.

Proof. Let us start by showing the existence of left (resp.) right MLEs. Let $a, b \in \{0,1\}^m$, where $a = (a_1, \dots, a_m), b = (b_1, \dots, b_m)$. Define $\chi_b(\mathtt{X}_1, \dots, \mathtt{X}_m) = \prod_{i=1}^m (\mathtt{X}_i(2b_i - 1) + 1 - b_i)$, which satisfies that $\chi_b(a) = 1$ if $a = b$ and $\chi_b(a) = 0$ otherwise. Observe that the coefficients of each monomial of $\chi_b(\mathtt{X}_1, \dots, \mathtt{X}_m)$ are in $Z(R)$. Hence, we can e.g. consider that either $\chi_b \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]_{\leq m,0}$ or $\chi_b \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]_{0,\leq m}$ according to the situation. The left multilinear extension of V is $\hat{V}_L(\mathtt{X}_1, \dots, \mathtt{X}_m) = \sum_{b \in \{0,1\}^m} \chi_b(\mathtt{X}) \cdot V(b)$ and the right multilinear extension of V is $\hat{V}_R(\mathtt{X}_1, \dots, \mathtt{X}_m) = \sum_{b \in \{0,1\}^m} V(b) \cdot \chi_b(\mathtt{X})$.

We next prove the uniqueness of LMLEs (the same reasoning applies to RMLEs). Assume $f, g \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]_{\leq 1,0}$ are LMLEs of V such that $f \neq g$ and let $h = f - g$. Since $f \neq g$, then $h \in R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]_{\leq 1,0}$ is not the zero polynomial. Let $S = \{(d_1, \dots, d_m) \in \{0,1\}^m : h_{(d_1,\dots,d_m)} \neq 0\}$ denote the set of indices of non-zero monomials in h and let $(D_1, \dots, D_m) \in \underset{(d_1,\dots,d_m) \in S}{\arg\min} \left(\sum_{i=1}^m d_i\right)$ be the index of any of the monomials with the least amount of indeterminates. Denote $D = (D_1, \dots, D_m)$. We are about to show that $h(D) \neq 0$. First of all, the evaluation at D of the D-th monomial is the coefficient of such monomial, which is non-zero by the definition of S. Furthermore, every other monomial evaluates to zero according to how D was chosen (any monomial with an indeterminate X_j such that $D_j = 0$ will evaluate to zero). But $h(D) \neq 0$ is in contradiction with the facts that $h(D) = f(D) - g(D)$ and that both f and g are LMLEs of V, i.e. $f(D) = g(D) = V(D)$. Hence, it has to be that $f = g$.

Finally, let us look at the simplification to MLE. When $A \subset Z(R)$, since $C_R(A) = R$, using the commutator ideal of $R_A[\mathtt{X}_1, \dots, \mathtt{X}_m]$ we have

$$\hat{V}_L(\mathtt{X}_1, \dots, \mathtt{X}_m) = \sum_{b \in \{0,1\}^m} \chi_b(\mathtt{X}) \cdot V(b) = \sum_{b \in \{0,1\}^m} V(b) \cdot \chi_b(\mathtt{X}) = \hat{V}_R(\mathtt{X}_1, \dots, \mathtt{X}_m)$$

For the case when $V : \{0,1\}^m \to C_R(A)$, we reason analogously. ∎

4 Doubly-efficient IP over non-commutative rings: Regular difference set contained in $Z(R)$

We can now introduce our first generalization of the GKR protocol [GKR15] to layered arithmetic circuits over possibly infinite, possibly non-commutative rings. In this variant, we assume that A

is a regular difference set such that $A \subset Z(R)$. This greatly simplifies our protocol compared with the one we will present in Section 5, where we only assume that A is commutative.

As most building blocks work essentially as in the finite commutative ring case [CCKP19], we only give a high level overview of this simpler variant. Since $A \subset Z(R)$, all polynomials can be expressed as elements from $R[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ (the ring in Definition 10) rather than $R_A[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ (see Corollary 1). This simpler polynomial ring definition is good enough in this case, since polynomial evaluation at elements in A is a ring homomorphism (Lemma 3) and furthermore we can bound the number of roots of these polynomials in A. The latter has been proved in [QBC13, ES21], where they use exceptional rather than R.D. sets, but such assumption can be weakened for that result. Furthemore, we have unique MLEs rather than LMLEs and RMLEs (see Lemma 9) and both the layer consistency equation (Eq. (1)) and sum-check protocol generalize naturally.

The only state-of-the-art tool for doubly-efficient IPs that requires more care in this scenario is the linear time prover sum-check protocol presented in Libra [XZZ+19]. These are all problems which we encounter and solve in the more restrictive case of Section 5 and hence they can also be solved at least as efficiently here. We give an overview of the issues for readers familiarized with [XZZ+19]. First of all, if the Lenstra constant of R is not at least 3, the prover needs to send intermediate polynomials by giving away their coefficients, rather than their evaluation at three different points. Additionally, for non-commutative rings, we need to be careful about terms that get reordered in [XZZ+19]. Namely, during "phase two" of their sum-check algorithm, the authors rewrite expressions of the form $\sum_{x,y \in H^m} f_1(g,x,y) f_2(x) f_3(y)$ as $\sum_{x \in H^m} f_2(x) \cdot h_g(x)$, where $h_g(x) = \sum_{y \in H^m} f_1(g,x,y) f_3(y)$. We cannot assume this in our setting, since $f_2(x) \in R$ might not commute with $f_1(g,x,y)$. Nevertheless, if we rewrite Eq. (1) so that the the wiring predicates multiply on the right rather than the left, almost no modifications are needed. Otherwise, if we want to keep Eq. (1) as is (because we want to hardcode multiplication by constants not in $Z(R)$ on the left), we can still exploit the sparsity of wiring predicates to enable a linear time prover, as we demonstrate in the more complex case of Section 5.4.

Theorem 3. *Let R be a ring and $A \subset Z(R)$ a regular difference set. Let $C : R^n \to R^k$ be a depth-D layered arithmetic circuit. There is an interactive proof for C with soundness error $O(D \log |C| / |A|)$. Its round complexity is $O(D \log |C|)$ and it communicates $O(D \log |C|)$ elements in R. In terms of operations in R, the prover complexity is $O(|C|)$ and the verifier complexity is $O(n + k + D \log |C| + T)$, where T is the optimal time to evaluate every wiring predicate. For log-space uniform circuits, $T = \mathsf{poly}\log(|C|)$ and hence the IP is succinct.*

4.1 Improved efficiency

The generality of our construction opens up possibilities for concrete efficiency improvements. One such example is the case when the ring R over which the circuit is defined can be seen as a free module of rank d over a ring S with a R.D. set $A \subseteq Z(S)$. Namely, as long as a product of elements in R takes more than d products in S, we have achieved our goal: Once the circuit has been evaluated, all operations the prover performs are the (sum of) evaluation of polynomials in $R[\mathtt{X}]$ at random elements from A. Polynomial evaluation is (the sum of) the product of elements of R with elements of S. Hence, if the ratio between the product of two elements in R and the product of an element of R with an element of S is bigger than the constants hidden in the $O(|C|)$ complexity of the prover, this results in a *sublinear* time prover! We are only aware of two previous examples in the literature where the prover is sublinear in the size of the circuit: Matrix multiplication [Fre79, Tha13] and Fast Fourier Transforms (FFT) [LXZ21].

In [LXZ21], the authors provide a sum-check protocol for FFTs where the prover only needs to do additional $O(d)$ work to produce a proof for a vector of size d. This is sublinear, since the FFT complexity is $O(d \log d)$. If FFTs are used for fast polynomial multiplication, we also obtain sublinear time provers by taking R to be the polynomial ring and S its coefficient ring. Multiplying two degree-d polynomials requires either $O(d^2)$ or $O(d \log d)$ operations in S (since in practice, for smaller values of d the former approach might be preferable). Multiplying such a polynomial with an element of S, on the other hand, requires exactly d operations in S, which is a gap of either $O(d)$ or $O(\log d)$ between both approaches. Thus, with our protocol we obtain a sublinear prover for polynomial multiplication *regardless* of whether FFT is actually employed in practice.

For a matrix ring $R = \mathcal{M}_{n\times n}(S)$, we have that R is a free module of rank n^2 over S. Since the best matrix multiplication algorithms we know require way more than n^2 operations, we once again obtain a sublinear prover by applying the observation at the beginning of this subsection. All in all, when taking every other complexity metric into account, Thaler's optimal MATMUL protocol [Tha13] is still preferable to our approach in terms of concrete efficiency. Nevertheless, we find interesting the extent to which our construction is versatile: We can obtain sublinear provers as simple, natural instantiations, rather than having to design a specific protocol. Furthermore, as far as we know, ours is only the third conceptually different method allowing for sublinear provers when dealing with matrix multiplication [Fre79, Tha13].

Even if they do not necessarily achieve sublinear time provers, many other rings R benefit from the improvement implied from being a rank-d module over a ring S. This is the case of many Clifford algebras, whose applications we discussed in Section 1.2. For example, let $R = H(S)$ denote the quaternions with coefficients over a ring S. We have that $Z(R) = Z(S)$, and R is a free module of rank 4 over S. Multiplying two elements in R requires at least 7 multiplications in S for a commutative ring S (or at least 8 in the non-commutative case) [HL75]. Dual quaternions are of rank 8 over their coefficient ring S, whereas their product consists on three quaternion products. Hence, if S is commutative, the prover would roughly obtain a factor of $21/8 = 2.625$ improvement compared with running over [CCKP19] over S.

Theoretical improvements. If we furthermore assumed that addition and multiplication of elements of the chosen non-commutative ring R can be performed at unit cost, we can obtain a series of theoretical results. Even though one could imagine to have specific hardware for that goal, these observations remain mostly theoretical, as they require to work with exponential size rings.

First of all, in [HY11] Hrubeš and Yehudayoff show that given a polynomial f (over a ring S) of degree d in n variables, there is a non-commutative extension ring R such that $S \subset Z(R)$ and f has a a formula of size $O(dn)$ over R. On the other hand, if S is an algebraically closed field, no commutative extension ring R can reduce the formula or circuit complexity of f. These would all seem good news for us: Non-commutativity might be a requirement, the resulting formula is really small and furthermore R could potentially be a free module over S. Unfortunately, the dimension of this ring extension is roughly n^d.

In [SS10], Schott and Staples show how many **NP**-complete and $\sharp$**P**-complete problems can be moved to class **P** if addition and multiplication in a Clifford algebra can be assumed to have unit cost. These include: Hamiltonian cycle problem, set covering problem, counting the edge-disjoint cycle decompositions of a finite graph, computing the permanent of an arbitrary matrix, computing the girth and circumference of a graph, and finding the longest path in a graph. Thus, in this model of computation, Theorem 3 provides us with a doubly-efficient IP for those languages[8]. Remember that, since for a language to have a doubly-efficient IP it has to belong to **BPP**, this was out of reach in the non-algebraic complexity world! Once again, the problem is that the algebra has an exponential dimension.

5 Doubly-efficient IP over non-commutative rings: Commutative, regular difference set

We can now introduce our most general doubly-efficient IP. Its description and properties follow nicely from the definitional work we did in Section 3 and the results we proved for toast and sandwich polynomials. It works for rings that are possibly infinite and non-commutative as long as they contain a commutative R.D. set A such that $A \subset R$. At several points, and merely as a writing simplification, we will add the condition $\{0,1\} \subset A$. If we remove that assumption, we can work with polynomials in $R_{A\cup\{0,1\}}[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2,\leq 2}$ rather than $R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2,\leq 2}$, since if A is a commutative set, so is $A \cup \{0,1\}$. Still, in order to avoid complicating the notation, we state our results in the simpler scenario.

An example ring to which this section applies is $R = \mathcal{M}_{n\times n}(\mathbb{Z}/p^k\mathbb{Z})$ for a prime p. Since we can embed the Galois Ring $S = \mathsf{GR}(p^k, n)$ into R and S contains an exceptional set A of size p^n, we can pick that same A as our commutative, R.D. set [ES21]. On the other hand, we have that $Z(R) = \{a \cdot Id : a \in \mathbb{Z}/p^k\mathbb{Z}\}$, so the biggest regular difference set *contained* in $Z(R)$ is of size p. Hence, for small values of p, Section 4 might not be enough for soundness.

Our construction follows the archetype of [GKR15], which we described in Sections 1.1 and 2.1. Let us now provide an overview of the required changes. First of all, we must adapt the layer consistency equation (Eq. (1)), as we describe in Section 5.1. Second, we need a way to avoid an exponential amount of sum-check executions as we progress through the layers of the circuit. We show two ways of dealing with this, one in Section 5.2 and another in Appendix B. These new equations require to design new sum-check protocols, which appear in Section 5.3 and Appendix B. Finally, in Section 5.4, we provide detailed algorithms for the prover to run in linear time when executing such sum-check protocols, as in Libra [XZZ+19]. Our final protocol appears in Figure 4 (Section 5.5).

5.1 A new layer consistency equation

Let us look at Equation (1). The first problem that we encounter when trying to generalize it to this setting, is that we are no longer able to define MLEs of the $V_i : \{0,1\}^{s_i} \to R$ functions which map $b \in \{0,1\}^{s_i}$ to the b-th wire in the i-th layer. Instead, we need to content ourselves with either LMLEs or RMLEs for those functions (see Section 3.2). A natural impulse would be to settle for e.g. the LMLE $\hat{V}_L^{(i)}(\vec{\mathtt{Z}})$ and express the consistency with layer $i+1$ as follows, where $\widehat{\mathtt{add}}^{(i+1)}(\vec{\mathtt{Z}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}), \widehat{\mathtt{mult}}^{(i+1)}(\vec{\mathtt{Z}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}) \in R_A[\vec{\mathtt{X}}, \vec{\mathtt{Y}}, \vec{\mathtt{Z}}]_{\leq 1,0}$, $\hat{V}_L^{(i+1)}(\vec{\mathtt{X}}) \in R_A[\vec{\mathtt{X}}]_{\leq 1,0}$ and $\hat{V}_R^{(i+1)}(\vec{\mathtt{Y}}) \in$

[8] In all precision, Theorem 3 only deals with layered arithmetic circuits, but our result can be generalized to general arithmetic circuits the same way as in [ZLW+21].

$R_A[\vec{\mathtt{Y}}]_{0,\leq 1}$:

$$\hat{V}_L^{(i)}(\vec{\mathtt{Z}}) = \sum_{x,y\in\{0,1\}^{s_{i+1}}} \Big(\widehat{\mathtt{mult}}^{(i+1)}(\vec{\mathtt{Z}},x,y)\cdot\big(\hat{V}_L^{(i+1)}(x)\cdot\hat{V}_R^{(i+1)}(y)\big) + \\ + \widehat{\mathtt{add}}^{(i+1)}(\vec{\mathtt{Z}},x,y)\cdot\big(\hat{V}_L^{(i+1)}(x)+\hat{V}_R^{(i+1)}(y)\big)\Big). \tag{3}$$

Both the good and the bad thing about this attempt is that the right hand side is a *sandwich* polynomial in $R_A[\vec{\mathtt{X}},\vec{\mathtt{Y}},\vec{\mathtt{Z}}]_{\leq 1,\leq 1}$, since the coefficients of the wiring predicates belong to $Z(R)$. Having such a sandwich is problematic when defining a sum-check protocol, which would progress through univariate polynomials in each indeterminate by partially evaluating the right hand side of Eq. (3). More specifically, the problem is with the $\mathtt{Y}_j$ indeterminates (for $j = 1,\dots,s_{i+1}$), as the partial evaluations sent by the prover would be *sandwich* polynomials $R_A[\mathtt{Y}_j]_{\leq 1,\leq 1}$. Since our Schwartz-Zippel lemma (Lemma 8) only copes with *toast* polynomials, this will not provide us with a sound protocol.

In order to have toast polynomials at every step of the sum-check protocol, we replace $\widehat{\mathtt{add}}^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{Y}})$, $\widehat{\mathtt{mult}}^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{Y}}) \in R_A[\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{Y}}]_{\leq 1,0}$ in Equation (3) with $\widehat{\mathtt{add}}_{\mathtt{L}}^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{W}}), \widehat{\mathtt{mult}}_{\mathtt{L}}^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{W}}) \in R_A[\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{W}}]_{\leq 1,0}$, still evaluating $\vec{\mathtt{W}}$ in y. This seemingly minor change requires to develop a new sum-check protocol which ensures that $\mathcal{P}$ evaluates $\vec{\mathtt{Y}}$ and $\vec{\mathtt{W}}$ in the same y. We postpone its description to Section 5.3. For the layer consistency equations featuring $\hat{V}_R^{(i)}(\vec{\mathtt{Z}}) \in R_A[\vec{\mathtt{Z}}]_{0,\leq 1}$, we will also do a change of variables, as we describe in the following lemma. For easier reference, as it will be important to keep track of which variables appear in every polynomial and whether it is a MLE, LMLE or RMLE, we display that information in Table 1.

Lemma 10. *Let $\vec{\mathtt{Z}} = (\mathtt{Z}_1,\dots,\mathtt{Z}_{s_i})$. Toast multilinear polynomials $\hat{V}_L^{(i)} \in R_A[\vec{\mathtt{Z}}]_{\leq 1,0}$ and $\hat{V}_R^{(i)} \in R_A[\vec{\mathtt{Z}}]_{0,\leq 1}$ are equal to the following expressions:*

$$\hat{V}_L^{(i)}(\vec{\mathtt{Z}}) = \sum_{x,y\in\{0,1\}^{s_{i+1}}} \Big(\widehat{\mathtt{mult}}_L^{(i+1)}(\vec{\mathtt{Z}},x,y)\cdot\big(\hat{V}_L^{(i+1)}(x)\cdot\hat{V}_R^{(i+1)}(y)\big) + \\ + \widehat{\mathtt{add}}_L^{(i+1)}(\vec{\mathtt{Z}},x,y)\cdot\big(\hat{V}_L^{(i+1)}(x)+\hat{V}_R^{(i+1)}(y)\big)\Big). \tag{4}$$

$$\hat{V}_R^{(i)}(\vec{\mathtt{Z}}) = \sum_{x,y\in\{0,1\}^{s_{i+1}}} \Big(\big(\hat{V}_L^{(i+1)}(x)\cdot\hat{V}_R^{(i+1)}(y)\big)\cdot\widehat{\mathtt{mult}}_R^{(i+1)}(\vec{\mathtt{Z}},x,y) + \\ + \big(\hat{V}_L^{(i+1)}(x)+\hat{V}_R^{(i+1)}(y)\big)\cdot\widehat{\mathtt{add}}_R^{(i+1)}(\vec{\mathtt{Z}},x,y)\Big). \tag{5}$$

Where, in Eq. (4), $\widehat{\mathtt{add}}_L^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{W}}), \widehat{\mathtt{mult}}_L^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{X}},\vec{\mathtt{W}}) \in R_A[\vec{\mathtt{X}},\vec{\mathtt{W}},\vec{\mathtt{Z}}]_{\leq 1,0}$ and in Eq. (5), $\widehat{\mathtt{add}}_R^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{U}},\vec{\mathtt{Y}})$, $\widehat{\mathtt{mult}}_R^{(i+1)}(\vec{\mathtt{Z}},\vec{\mathtt{U}},\vec{\mathtt{Y}}) \in R_A[\vec{\mathtt{Y}},\vec{\mathtt{U}},\vec{\mathtt{Z}}]_{\leq 1,0}$.

Proof. The term on each side of Eq. (4) (resp. Eq. (5)) is a multilinear polynomial in $R_A[\vec{\mathtt{Z}}]_{\leq 1,0}$ (resp. $R_A[\vec{\mathtt{Z}}]_{0,\leq 1}$), so by the uniqueness of LMLEs (resp. RMLEs), we are done if their evaluation at every $z \in \{0,1\}^{s_i}$ coincides. The latter follows from the definitions of $\widehat{\mathtt{add}}_{\mathtt{L}}^{(i+1)}, \widehat{\mathtt{mult}}_{\mathtt{L}}^{(i+1)}$ (resp. $\widehat{\mathtt{add}}_{\mathtt{R}}^{(i+1)}, \widehat{\mathtt{mult}}_{\mathtt{R}}^{(i+1)}$). ∎

Remark 1. An interesting detail about this construction, in which $A \not\subset Z(R)$, is that it is necessary for wiring predicates to be MLEs, rather than simply LMLEs or RMLEs. Otherwise, we would not obtain toast polynomials (in the $\mathtt{X}_i$ variables for Equation (4), in $\mathtt{Y}_i$ variables for Equation (5)) throughout the execution of the sumcheck protocol and we would be unable to apply Lemma 8 to determine soundness. Whereas for the standard addition and multiplication gates (since $\{0,1\} \subseteq Z(R) \subseteq C_R(A)$) we always obtain MLEs, if we use more complex wiring predicates which enable multiplication by hard-coded constants, those constants have to belong to $C_R(A)$.

5.2 2-to-1 reduction

Polynomial	$\widehat{\mathtt{add}_\mathtt{L}}^{(i+1)}, \widehat{\mathtt{mult}_\mathtt{L}}^{(i+1)}$	$\widehat{\mathtt{add}_\mathtt{R}}^{(i+1)}, \widehat{\mathtt{mult}_\mathtt{R}}^{(i+1)}$	$\hat{V}_L^{(i+1)}$	$\hat{V}_R^{(i+1)}$	$\hat{V}_L^{(i)}$	$\hat{V}_R^{(i)}$
(L/R)MLE	MLE	MLE	LMLE	RMLE	LMLE	RMLE
Ring	$R_A[\vec{\mathtt{X}}, \vec{\mathtt{W}}, \vec{\mathtt{Z}}]_{\leq 1,0}$	$R_A[\vec{\mathtt{Y}}, \vec{\mathtt{U}}, \vec{\mathtt{Z}}]_{\leq 1,0}$	$R_A[\vec{\mathtt{X}}]_{\leq 1,0}$	$R_A[\vec{\mathtt{Y}}]_{0,\leq 1}$	$R_A[\vec{\mathtt{Z}}]_{\leq 1,0}$	$R_A[\vec{\mathtt{Z}}]_{0,\leq 1}$

Table 1. Polynomials involved in layer consistency equations. Note that MLEs such as $\widehat{\mathtt{add}_\mathtt{L}}^{(i+1)}$ could be considered as either polynomials in $R_A[\vec{\mathtt{X}}, \vec{\mathtt{W}}, \vec{\mathtt{Z}}]_{\leq 1,0}$ or $R_A[\vec{\mathtt{X}}, \vec{\mathtt{W}}, \vec{\mathtt{Z}}]_{0,\leq 1}$.

Our protocol starts with a simple layer consistency equation, which relates the output layer with layer 1 in the circuit according to the following equation:

$$\hat{V}_L^{(0)}(\gamma) = \sum_{x,y \in \{0,1\}^{s_1}} \Big(\widehat{\mathtt{mult}_\mathtt{L}}^{(1)}(\gamma, x, y) \cdot \big(\hat{V}_L^{(1)}(x) \cdot \hat{V}_R^{(1)}(y)\big) + \\ + \widehat{\mathtt{add}_\mathtt{L}}^{(1)}(\gamma, x, y) \cdot \big(\hat{V}_L^{(1)}(x) + \hat{V}_R^{(1)}(y)\big)\Big). \tag{6}$$

At the conclusion of the sumcheck protocol which is run to verify Equation (6), $\mathcal{V}$ encounters one of the usual obstacles in this kind of constructions. Namely, $\mathcal{V}$ needs to evaluate the following expression:

$$\widehat{\mathtt{mult}_\mathtt{L}}^{(1)}(\gamma, \vec{\mathtt{X}}, \vec{\mathtt{W}}) \cdot (\hat{V}_L^{(1)}(\vec{\mathtt{X}}) \cdot \hat{V}_R^{(1)}(\vec{\mathtt{Y}})) + \widehat{\mathtt{add}_\mathtt{L}}^{(1)}(\gamma, \vec{\mathtt{X}}, \vec{\mathtt{W}}) \cdot (\hat{V}_L^{(1)}(\vec{\mathtt{X}}) + \hat{V}_R^{(1)}(\vec{\mathtt{Y}}))$$

by replacing $\vec{\mathtt{X}}, \vec{\mathtt{Y}}, \vec{\mathtt{W}}$ with respective random values $\chi^{(0)}, \psi^{(0)}, \omega^{(0)} \in A^{s_1}$. In our protocol, we assume that $\mathcal{V}$ has access to oracles that return the required evaluations[9] $\widehat{\mathtt{mult}_\mathtt{L}}^{(i+1)}(\gamma, \chi^{(i)}, \omega^{(i)})$, $\widehat{\mathtt{add}_\mathtt{L}}^{(i+1)}(\gamma, \chi^{(i)}, \omega^{(i)})$ (for $i = 0, \dots, D-1$). However, $\mathcal{V}$ cannot compute neither $\hat{V}_L^{(1)}(\chi^{(0)})$ nor $\hat{V}_R^{(1)}(\psi^{(0)})$ on their own, so $\mathcal{P}$ will provide those values. These new values claimed by the Prover have to satisfy the two following layer consistency equations:

$$\hat{V}_L^{(1)}(\chi^{(0)}) = \sum_{x,y \in \{0,1\}^{s_2}} \Big(\widehat{\mathtt{add}_\mathtt{L}}^{(2)}(\chi^{(0)}, x, y) \cdot (\hat{V}_L^{(2)}(x) + \hat{V}_R^{(2)}(y)) + \\ + \widehat{\mathtt{mult}_\mathtt{L}}^{(2)}(\chi^{(0)}, x, y) \cdot (\hat{V}_L^{(2)}(x) \cdot \hat{V}_R^{(2)}(y))\Big). \tag{7}$$

[9] For circuits with enough structure, the oracles can be either computed or verified by $\mathcal{V}$ in $\mathsf{poly}(s_i, s_{i+1})$ time, as shown in prior works.

$$\hat{V}_R^{(1)}(\psi^{(0)}) = \sum_{x,y\in\{0,1\}^{s_2}} \Big((\hat{V}_L^{(2)}(x) + \hat{V}_R^{(2)}(y)) \cdot \widehat{\texttt{add}_\texttt{R}}^{(2)}(\psi^{(0)},x,y) + \\ + (\hat{V}_L^{(2)}(x)\cdot \hat{V}_R^{(2)}(y)) \cdot \widehat{\texttt{mult}_\texttt{R}}^{(2)}(\psi^{(0)},x,y)\Big). \tag{8}$$

In order to avoid an exponential blow-up in the depth of the circuit, we perform a reduction from the two claimed values $\hat{V}_L^{(1)}(\chi^{(0)}), \hat{V}_R^{(1)}(\psi^{(0)})$ to a single one. We do so by sampling random values $\alpha^{(1)}, \beta^{(1)} \in A$ and combining Equations (7) and (8) as described in the next, more general equation:

$$\alpha^{(i)}\hat{V}_L^{(i)}(\chi^{(i-1)}) + \beta^{(i)}\hat{V}_R^{(i)}(\psi^{(i-1)}) = \sum_{x,y\in\{0,1\}^{s_{i+1}}} \\ \Big(\quad \alpha^{(i)} \cdot \widehat{\texttt{mult}_\texttt{L}}^{(i+1)}(\chi^{(i-1)},x,y)\cdot(\hat{V}_L^{(i+1)}(x)\cdot\hat{V}_R^{(i+1)}(y)) + \\ + \beta^{(i)}\cdot(\hat{V}_L^{(i+1)}(x)\cdot\hat{V}_R^{(i+1)}(y))\cdot\widehat{\texttt{mult}_\texttt{R}}^{(i+1)}(\psi^{(i-1)},x,y) + \\ + \alpha^{(i)}\cdot\widehat{\texttt{add}_\texttt{L}}^{(i+1)}(\chi^{(i-1)},x,y)\cdot(\hat{V}_L^{(i+1)}(x)+\hat{V}_R^{(i+1)}(y)) + \\ + \beta^{(i)}\cdot(\hat{V}_L^{(i+1)}(x)+\hat{V}_R^{(i+1)}(y))\cdot\widehat{\texttt{add}_\texttt{R}}^{(i+1)}(\psi^{(i-1)},x,y) \quad \Big) \tag{9}$$

Lemma 11. *Let $\alpha^{(i)}, \beta^{(i)} \in A$ be chosen uniformly at random. Except with probability $2/|A|$, Equation (9) holds only if Equations (7) and (8) hold.*

Proof. Let L_1, R_1 denote the left (resp. right) hand side of Equation (7). Let L_2, R_2 denote the left (resp. right) hand side of Equation (8). Assume the verification of Equation (9) passes (i.e. $\alpha(L_1 - R_1) + \beta(L_2 - R_2) = 0$) but any of $L_i \neq R_i$. By Lemma 8, that would mean that (α, β) is a root of the bivariate polynomial $p(\texttt{T}_1, \texttt{T}_2) = \texttt{T}_1(L_1 - R_1) + \texttt{T}_2(L_2 - R_2)$, which only happens with probability $2/|A|$. ■

As we are about to see in Section 5.3, this 2-to-1 reduction will lead to a sum-check protocol that runs in $4m$ rounds, sends a total of $14m$ ring elements and has a soundness error of $8m|A|^{-1}$, where m is the amount of variables in each vector $\vec{\texttt{X}}, \vec{\texttt{Y}}, \vec{\texttt{W}}, \vec{\texttt{U}}$. In Appendix B we provide a different approach using a 4-to-2 reduction, for which the underlying sum-check protocol only requires $3m$ rounds and sending $9m$ ring elements, with a soundness error of $6m|A|^{-1}$. The downside of the alternative is that *two* parallel sumcheck protocols need to be executed for (almost) every layer of the circuit, resulting in a communication complexity of $18m$ ring elements per circuit layer.

5.3 Sum-check for non-commutative layer consistency

In this subsection, we generalize the sum-check protocol to work for the layer consistency equations of our doubly-efficient IP. Specifically, we provide the protocol for Equation (9)[10]. This description is simple enough to analyze its soundness error, communication and round complexity. We postpone the specific algorithm run by the prover and its complexity analysis to Section 5.4. Remember that A is an R.D. commutative set that contains $H = \{0, 1\}$.

[10] The simpler protocol for Equation (6) can be found in Appendix B.2

Sum-check protocol for Equation (9): Let $\vec{x} = (x_1, \dots, x_m)$, $\vec{y} = (y_1, \dots, y_m)$. We provide a sum-check protocol for $\sum_{\vec{x},\vec{y} \in H^m} f(\vec{x}, \vec{y}, \vec{x}, \vec{y}) = \beta$, where $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ is the multi-variate sandwich polynomial given by Equation (9). If any of the checks throughout the protocol are not satisfied, $\mathcal{V}$ rejects.

1. In the first round, for $b \in \{0, 1\}$, $\mathcal{P}$ computes the univariate toast polynomials $g_{1,b} \in R_A[\mathtt{U}_1]_{0,\leq 1}$ given by:
$$g_{1,b}(\mathtt{U}_1) = \sum_{\substack{x_2,\dots,x_m \in H \\ \vec{y} \in H^m}} f(\mathtt{U}_1, x_2, \dots, x_m, \vec{y}, b, x_2, \dots, x_m, \vec{y}),$$
and sends them to $\mathcal{V}$. Then $\mathcal{V}$ checks whether $g_{1,0}, g_{1,1} \in R_A[\mathtt{U}_1]_{0,\leq 1}$ and $\sum_{b \in H} g_{1,b}(b) = \beta$. If that is the case, $\mathcal{V}$ chooses a random element $r_1 \in A$ and sends it to $\mathcal{P}$.
2. For rounds $2 \leq i \leq m$, define $\vec{x}_{(i,m]} = (x_{i+1}, \dots, x_m)$ and $\vec{x}_{[1,i)} = (x_1, \dots, x_{i-1})$. $\mathcal{P}$ sends the univariate toast polynomials $g_{i,0}, g_{i,1} \in R_A[\mathtt{U}_i]_{0,\leq 1}$ given by:
$$g_{i,b}(\mathtt{U}_i) = \sum_{\substack{\vec{x}_{[1,i)} \in H^{i-1}, \vec{x}_{(i,m]} \in H^{m-i} \\ \vec{y} \in H^m}} f(r_1, \dots, r_{i-1}, \mathtt{U}_i, \vec{x}_{(i,m]}, \vec{y}, \vec{x}_{[1,i)}, b, \vec{x}_{(i,m]}, \vec{y}),$$
$\mathcal{V}$ checks whether $g_{i,b} \in R_A[\mathtt{U}_i]_{0,\leq 1}$ and $\sum_{b \in H} g_{i,b}(b) - g_{i-1,b}(r_{i-1}) = 0$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
3. For rounds $m+1 \leq i \leq 2m$, $\mathcal{P}$, define $j = i - m$, $\vec{r}_{[1,i)} = (r_1, \dots, r_{i-1})$, $\vec{y}_{(j,m]} = (y_{j+1}, \dots, y_m)$ and $\vec{y}_{[1,j)} = (y_1, \dots, y_{j-1})$. $\mathcal{P}$ sends the univariate toast polynomials $g_{i,0}, g_{i,1} \in R_A[\mathtt{W}_j]_{\leq 1,0}$ given by:
$$g_{i,b}(\mathtt{W}_j) = \sum_{\substack{\vec{y}_{[1,j)} \in H^{j-1}, \vec{y}_{(j,m]} \in H^{m-j} \\ \vec{x} \in H^m}} f(\vec{r}_{[1,i)}, \mathtt{W}_j, \vec{y}_{(j,m]}, \vec{x}, \vec{y}_{[1,j)}, b, \vec{y}_{(j,m]}),$$
$\mathcal{V}$ checks whether $g_{i,0}, g_{i,1} \in R_A[\mathtt{W}_j]_{\leq 1,0}$ and $\sum_{b \in H} g_{i,b}(b) - g_{i-1,b}(r_{i-1}) = 0$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
4. For round $i = 2m+1$, define $\vec{r}_{[1,2m+1)} = (r_1, \dots, r_{2m})$ and $\vec{x}_{(1,m]} = (x_2, \dots, x_m)$. $\mathcal{P}$ sends the toast polynomial $g_{2m+1} \in R_A[\mathtt{X}_1]_{\leq 2,0}$ given by:
$$g_{2m+1}(\mathtt{X}_1) = \sum_{\vec{x}_{(1,m]} \in H^{m-1}, \vec{y} \in H^m} f(\vec{r}_{[1,2m+1)}, \mathtt{X}_1, \vec{x}_{(1,m]}, \vec{y}),$$
$\mathcal{V}$ checks whether $g_{2m+1} \in R_A[\mathtt{X}_1]_{\leq 2,0}$ and $\sum_{b \in H} g_{2m+1}(b) - g_{2m,b}(r_{2m}) = 0$. If so, $\mathcal{V}$ chooses a random element $r_{2m+1} \in A$ and sends it to $\mathcal{P}$.
5. For rounds $2m+2 \leq i \leq 3m$, define $j = i - 2m$, $\vec{r}_{[1,i)} = (r_1, \dots, r_{i-1})$ and $\vec{x}_{(j,m]} = (x_{j+1}, \dots, x_m)$. $\mathcal{P}$ sends the toast polynomial $g_i \in R_A[\mathtt{X}_j]_{\leq 2,0}$ given by:
$$g_i(\mathtt{X}_j) = \sum_{\vec{x}_{(j,m]} \in H^{m-j}, \vec{y} \in H^m} f(\vec{r}_{[1,i)}, \mathtt{X}_j, \vec{x}_{(j,m]}, \vec{y})$$
$\mathcal{V}$ checks whether $g_i \in R_A[\mathtt{X}_j]_{\leq 2,0}$ and $\sum_{b \in H} g_i(b) = g_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.

6. For rounds $3m+1 \leq i \leq 4m$, define $j = i-3m$, $\vec{r}_{[1,i)} = (r_1, \ldots, r_{i-1})$ and $\vec{y}_{(j,m]} = (y_{j+1}, \ldots, y_m)$. $\mathcal{P}$ sends the toast polynomial $g_i \in R_A[\mathtt{Y}_j]_{0,\leq 2}$ given by:

$$g_i(\mathtt{Y}_j) = \sum_{\vec{y}_{(j,m]} \in H^{m-j}} f(\vec{r}_{[1,i)}, \mathtt{Y}_j, \vec{y}_{(j,m]}),$$

$\mathcal{V}$ checks whether $g_i \in R_A[\mathtt{Y}_j]_{0,\leq 2}$ and $\sum_{b \in H} g_i(b) = g_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
7. After the $4m$-th round, $\mathcal{V}$ checks whether $g_{4m}(r_{4m}) = f(r_1, \ldots, r_{4m})$ by querying[11] its oracles at $(r_1, \ldots, r_{4m})$.

Theorem 4. *Let A be a commutative R.D. set such that $\{0,1\} \subseteq A$. Let $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ be the multi-variate sandwich polynomial given by Equation (9). The sum-check protocol is a public coin interactive proof with soundness error $\leq 8m \cdot |A|^{-1}$. The communication complexity is $14m$ elements in R.*

Proof. Completeness and communication complexity follow from inspection of the protocol, and hence we will concentrate on the soundness claim.

Let $\tilde{\mathcal{P}}$ denote an arbitrary malicious prover, trying to convince the verifier of a false claim $\sum_{\vec{x},\vec{y} \in H^m} f(\vec{x}, \vec{y}, \vec{x}, \vec{y}) = \tilde{\beta}$, where $\tilde{\beta} \neq \beta$. During each of the $4m$ rounds, $\tilde{\mathcal{P}}$ has to send toast polynomials, specifically $\tilde{g}_{i,0}, \tilde{g}_{i,1} \in R_A[\mathtt{U}_i]_{0,\leq 1}$ for $i \in [m]$, $\tilde{g}_{i,0}, \tilde{g}_{i,1} \in R_A[\mathtt{W}_{i-m}]_{\leq 1,0}$ for $i \in [m+1, 2m]$, $\tilde{g}_i \in R_A[\mathtt{X}_{i-2m}]_{\leq 2,0}$ for $i \in [2m+1, 3m]$ and $\tilde{g}_i \in R_A[\mathtt{Y}_{i-3m}]_{0,\leq 2}$ for $i \in [3m+1, 4m]$. Notice that the honest polynomials are indeed toasts, since the random challenges $r_1, \ldots, r_{i-1}$ are in A and can hence be "pushed to the middle" with the rest of the coefficients of each monomial. The verifier can easily check whether the polynomials received from $\tilde{\mathcal{P}}$ are also toasts of the right degree.

Let V denote the event where $\tilde{\mathcal{P}}$ succeeds cheating $\mathcal{V}$. For $i \in [1, 2m]$, let E_i denote the event that $\sum_{b \in H} \tilde{g}_{i,b} = \sum_{b \in H} g_{i,b}$ and for $i \in [2m+1, 4m]$, let E_i denote the event that $\tilde{g}_i = g_i$. Notice that $\Pr[V|E_1] = 0$, since $\mathcal{V}$ checks whether $\tilde{\beta}$ is equal to $\sum_{b \in H} \tilde{g}_{1,b}(b) = \sum_{b \in H} g_{1,b}(b) = \beta$.

Following reverse induction, we will prove that for $i = 4m, \ldots, 1$, $\Pr[V] \leq (4m - i + 1) \cdot 2 \cdot |A|^{-1} + \Pr[V|E_i \wedge \ldots \wedge E_{4m}]$. Once we prove the case $i = 1$ we will be done, since then $\Pr[V] \leq 4m \cdot 2 \cdot |A|^{-1} + \Pr[V|E_1 \wedge \ldots \wedge E_{4m}] \leq 4m \cdot 2 \cdot |A|^{-1} + \Pr[V|E_1] \leq 8m \cdot |A|^{-1}$.

For $i = 4m$, we have that $\Pr[V] \leq \Pr[V|\overline{E_{4m}}] + \Pr[V|E_{4m}] \leq 2/|A| + \Pr[V|E_{4m}]$. The inequality $\Pr[V|\overline{E_{4m}}] \leq 2/|A|$ follows from Lemma 8 as we explain next. Define $G_{4m}(\mathtt{Y}_m) = \tilde{g}_{4m}(\mathtt{Y}_m) - g_{4m}(\mathtt{Y}_m) \in R_A[\mathtt{Y}_m]_{0,\leq 2}$. Since we are in the case $\overline{E_{4m}}$, then $\tilde{g}_{4m} \neq g_{4m}$ and $G_{4m}(\mathtt{Y}_m)$ is *not* the zero polynomial. As the Verifier checks whether $\tilde{g}_{4m}(r_{4m}) = f(r_1, \ldots, r_{4m})$ and we know that $g_{4m}(r_{4m}) = f(r_1, \ldots, r_{4m})$, passing the check implies that $G_{4m}(r_{4m}) = 0$.

As an intermediate step towards proving that the statment is true for the $(i-1)$-th case, let us show that for $i = 4m, \ldots, 2$, we have $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{4m}] \leq 2 \cdot |A|^{-1}$. First, for $i = 4m, \ldots, 3m+1$, define the toast polynomial

$$G_{i-1}(\mathtt{Y}_{i-3m-1}) = \tilde{g}_{i-1}(\mathtt{Y}_{i-3m-1}) - g_{i-1}(\mathtt{Y}_{i-3m-1}) \in R_A[\mathtt{Y}_{i-3m-1}]_{0,\leq 2},$$

which is non-zero in the event $(V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{4m})$. Since $\mathcal{P}$ successfully passes the check $\sum_{b \in H} g_i(b) = \tilde{g}_{i-1}(r_{i-1})$ and we know that $\sum_{b \in H} g_i(b) = g_{i-1}(r_{i-1})$, by applying Lemma 8 we

[11] As usual in the GKR protocol, some values are actually provided by $\mathcal{P}$, unless the input layer has been reached. This step is more detailed in Figure 4, but we need to provide a simpler description here for self-containment.

conclude that $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{4m}] \leq |A|^{-1} \leq 2 \cdot |A|^{-1}$. For $i = 3m, \ldots, 2m+2$ we apply the same reasoning of the previous paragraph, this time for the toast polynomial

$$G_{i-1}(\mathtt{X}_{i-2m-1}) = \tilde{g}_{i-1}(\mathtt{X}_{i-2m-1}) - g_{i-1}(\mathtt{X}_{i-2m-1}) \in R_A[\mathtt{X}_{i-2m-1}]_{\leq 2,0}.$$

For $i = 2m+1, \ldots, m+2$ we apply Lemma 8 to the toast polynomial:

$$G_{i-1}(\mathtt{W}_{i-m-1}) = \sum_{b \in H} \tilde{g}_{i-1,b}(\mathtt{W}_{i-m-1}) - g_{i-1,b}(\mathtt{W}_{i-m-1}) \in R_A[\mathtt{W}_{i-m-1}]_{\leq 1,0}.$$

Finally, for $i = m+1, \ldots, 2$, we apply Lemma 8 to the toast polynomial:

$$G_{i-1}(\mathtt{U}_{i-1}) = \sum_{b \in H} \tilde{g}_{i-1,b}(\mathtt{U}_{i-1}) - g_{i-1,b}(\mathtt{U}_{i-1}) \in R_A[\mathtt{U}_{i-1}]_{0,\leq 1}.$$

Assume the induction hypothesis is true for i. Using our recently proved fact that $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{4m}] \leq 2 \cdot |A|^{-1}$, we have the following:

$$\begin{aligned}
\Pr[V] &\leq (4m-i+1) \cdot 2 \cdot |A|^{-1} + \Pr[V|E_i \wedge \ldots \wedge E_{4m}] \\
&\leq (4m-i+1) \cdot 2 \cdot |A|^{-1} + \Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{4m}] + \Pr[V|E_{i-1} \wedge E_i \wedge \ldots \wedge E_{4m}] \\
&\leq (4m-i+1) \cdot 2 \cdot |A|^{-1} + 2 \cdot |A|^{-1} + \Pr[V|E_{i-1} \wedge E_i \wedge \ldots \wedge E_{4m}]
\end{aligned}$$

Hence, $\Pr[V] \leq (4m-i+2) \cdot 2 \cdot |A|^{-1} + \Pr[V|E_{i-1} \wedge \ldots \wedge E_{4m}]$ and the statment is true for the $(i-1)$-th case. This finishes our reverse induction and concludes our proof by reaching the case $i = 1$. ■

5.4 Linear time Prover for Equation (9)

Multi-linear extensions were key for [CMT12] to improve the complexity of the Prover in [GKR15] from $\mathsf{poly}(|C|)$ to $O(|C|\log(|C|))$. In this section, we will show how to improve upon this to a complexity of $O(|C|)$, in a style similar to Libra [XZZ+19]. In order to achieve this, we will show how the Prover can execute the sum-check algorithm of Section 5.3 for Equation (9). Recall that $\mathcal{P}$ has to sum the evaluations of $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ in the hypercube H^{4m}, where $H = \{0,1\}$. Our following algorithms assume that $\mathcal{P}$ has an initial lookup table (LUT) $\boldsymbol{T}_f$ with these evaluations, as well as the same kind of lookup tables for its constituent (L/R)MLEs $\boldsymbol{T}_{\widehat{\mathtt{mult}_\mathtt{L}}}, \boldsymbol{T}_{\widehat{\mathtt{add}_\mathtt{L}}}, \boldsymbol{T}_{\hat{V}_L}, \boldsymbol{T}_{\widehat{\mathtt{mult}_\mathtt{R}}}, \boldsymbol{T}_{\widehat{\mathtt{add}_\mathtt{R}}}, \boldsymbol{T}_{\hat{V}_R}$. We also assume that $\mathcal{P}$ has received and stored the 2-to-1 reduction challenges $\alpha, \beta \in A$. For a simpler write-up, we write the different algorithms as if $\mathcal{P}$ already knew the challenge vector $\vec{r} = (r_1, \ldots, r_{4m}) \in A^{4m}$, even though they will receive the different r_i values as the progress through the execution of the sum-check protocol.

In constrast with [XZZ+19], we make the Prover provide the Verifier with explicit polynomials, rather than with their evaluations at (up to) three different points. We do this for the sake of generality[12], since interpolation requires exceptional rather than regular-difference sets, and the target ring (e.g. $\mathbb{Z}$) might not contain a commutative exceptional set of size three.

In order to avoid filling the following pages with its detailed description, we only provide a high level view of the algorithm in the main body. The different subroutines within it are detailed in Appendix A.

[12] It would be easy to modify our algorithms to work by providing polynomial evaluations instead. In fact, the set $\{0, 1, \gamma\}$ is commutative and exceptional as long as γ and $\gamma - 1$ are invertible. If 2 is not a zero divisor, we can always pick $\gamma = 2$. Otherwise we may still find such γ easily (as e.g. in $\mathbb{F}_{2^d}, \mathsf{GR}(2^k, d), \mathcal{M}_{n\times n}(\mathbb{Z}/2^k\mathbb{Z})$) or resort to ring extensions (e.g. embed $\mathbb{Z}/2^k\mathbb{Z}$ in $\mathsf{GR}(2^k, d)$ or $\mathbb{Z}$ in $\mathbb{R}$).

```
Algorithm Linear_P_Consistency($\boldsymbol{T}_f, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{L}}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{L}}}, \boldsymbol{T}_{\hat{V}_L}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{R}}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{R}}}, \boldsymbol{T}_{\hat{V}_R}, \vec{\chi}, \vec{\psi}, \alpha, \beta, \vec{r}$)

All figures referenced within this algorithm are in Appendix A.
Input: $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ from Equation (9) and its initial lookup table $\boldsymbol{T}_f$. Initial lookup tables of its constituent polynomials $\boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{L}}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{L}}}, \boldsymbol{T}_{\hat{V}_L}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{R}}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{R}}}, \boldsymbol{T}_{\hat{V}_R}$. Challenge vector $\vec{r} = (r_1, \ldots, r_{4m}) \in A^{4m}$ and 2-to-1 reduction challenges $\alpha, \beta \in A$.
Output: Sum-check messages for the layer consistency Equation (9).

1. Run Sumcheck_{U, W}($f, \boldsymbol{T}_f, \vec{r}_{[1,2m]}$) as described in Figure 6.
2. Run Setup_X($\widehat{\mathtt{mult}}_{\mathtt{L}}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{L}}}, \widehat{\mathtt{add}}_{\mathtt{L}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{L}}}, \widehat{\mathtt{mult}}_{\mathtt{R}}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{R}}}, \widehat{\mathtt{add}}_{\mathtt{R}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{R}}}, \vec{\chi}, \vec{\psi}, \vec{r}_{[1,2m]}$) as described in Figure 9 in order
   to obtain $\{\boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{L}}}(x), \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{L}}}(x), \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{R}}}(y), \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{R}}}(y)\}$.
3. $\mathcal{F}_{\hat{V}_L} \leftarrow$ Function_Evaluations($\hat{V}_L, \boldsymbol{T}_{\hat{V}_L}, r_{2m+1}, \ldots, r_{3m}$).
4. Run Sumcheck_X_Left($\widehat{\mathtt{mult}}_{\mathtt{L}}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{L}}}, \widehat{\mathtt{add}}_{\mathtt{L}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{L}}}, \mathcal{F}_{\hat{V}_L}, \hat{V}_R, \boldsymbol{T}_{\hat{V}_R}, \vec{r}_{[2m+1,3m]}$) as described in Figure 10 in order
   to obtain $\{g^{\mathtt{mult}_{\mathtt{L}}}_{2m+i}(\mathtt{X}_i), g^{\mathtt{add}_{\mathtt{L}}}_{2m+i}(\mathtt{X}_i)\}_{i=1}^{m}$.
5. Run Sumcheck_X_Right($\widehat{\mathtt{mult}}_{\mathtt{R}}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{R}}}, \widehat{\mathtt{add}}_{\mathtt{R}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{R}}}, \mathcal{F}_{\hat{V}_L}, \hat{V}_R, \boldsymbol{T}_{\hat{V}_R}, \vec{r}_{[2m+1,3m]}$) as described in Figure 11 in order
   to obtain $\{g^{\mathtt{mult}_{\mathtt{R}}}_{2m+i}(\mathtt{X}_i), g^{\mathtt{add}_{\mathtt{R}}}_{2m+i}(\mathtt{X}_i)\}_{i=1}^{m}$.
6. For $i \in [m]$, compute:

   $$g_{2m+i}(\mathtt{X}_i) = \alpha \cdot \left(g^{\mathtt{mult}_{\mathtt{L}}}_{2m+i}(\mathtt{X}_i) + g^{\mathtt{add}_{\mathtt{L}}}_{2m+i}(\mathtt{X}_i)\right) + \beta \cdot \left(g^{\mathtt{mult}_{\mathtt{R}}}_{2m+i}(\mathtt{X}_i) + g^{\mathtt{add}_{\mathtt{R}}}_{2m+i}(\mathtt{X}_i)\right).$$

7. Run Setup_Y($\widehat{\mathtt{mult}}_{\mathtt{L}}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{L}}}, \widehat{\mathtt{add}}_{\mathtt{L}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{L}}}, \vec{r}_{[2m+1,3m]}$) as described in Figure 12.
8. In order to obtain $\{g_{3m+i}(\mathtt{Y}_i)\}_{i=1}^{m}$, run Sumcheck_Y($\widehat{\mathtt{mult}}_{\mathtt{R}}, \boldsymbol{T}_{\widehat{\mathtt{mult}}_{\mathtt{R}}}, \widehat{\mathtt{add}}_{\mathtt{R}}, \boldsymbol{T}_{\widehat{\mathtt{add}}_{\mathtt{R}}}, \hat{V}_R, \boldsymbol{T}_{\hat{V}_R}, \hat{V}_L(\vec{r}_{[2m+1,3m]}), \alpha \cdot$
   $\widehat{\mathtt{mult}}_{\mathtt{L}}(\vec{\chi}, \vec{r}_{[m+1,3m]}), \alpha \cdot \widehat{\mathtt{add}}_{\mathtt{L}}(\vec{\chi}, \vec{r}_{[m+1,3m]}), \vec{r}_{[3m+1,4m]}, \beta$) as described in Figure 13.
```

Fig. 3. Linear time prover for the sum-check protocol in Section 5.3.

$\vec{\mathtt{U}}, \vec{\mathtt{W}}$ variables (Step 1). This is the easiest phase, since we can directly reason about $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ from Equation (9) and its LUT $\boldsymbol{T}_f$. `Sumcheck_{U, W}` (Figure 6) provides with the polynomials for these first $2m$ messages. All terms in the sums of Figure 6 can be found, in turn, in the lookup table $\mathcal{F}$ produced by `Function_Evaluations_{U, W}` (Figure 5).

The algorithm in Figure 5 follows from the simple observation that, because of linearity, any RMLE $f(\vec{r}, \mathtt{U}_i, \vec{t}) \in R_A[\mathtt{U}_i]_{0, \leq 1}$ satisfies that $f(\vec{r}, \mathtt{U}_i, \vec{t}) = (f(\vec{r}, 1, \vec{t}) - f(\vec{r}, 0, \vec{t})) \cdot \mathtt{U}_i + f(\vec{r}, 0, \vec{t})$. Notice that, whereas the initial lookup table $\boldsymbol{T}_f$ contains all the 2^{4m} evaluations of f from Equation (9) in $H = \{0, 1\}$, modifications only occur on the first $2m$ indices, which are the ones related to variables $\mathtt{U}$ and $\mathtt{W}$.

$\vec{\mathtt{X}}$ variables (Steps 2-6). In this phase, rather than reasoning about $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ from Equation (9), we look at its constituent polynomials ($\widehat{\mathtt{mult}}_{\mathtt{L}}$, $\widehat{\mathtt{add}}_{\mathtt{L}}$, $\hat{V}_L$, $\widehat{\mathtt{mult}}_{\mathtt{R}}$, $\widehat{\mathtt{add}}_{\mathtt{R}}$, $\hat{V}_R$) separately. Dealing with non-commutative rings is the main reason for the different algorithms in these steps, compared with the simpler description in Libra [XZZ+19]. Whereas in Libra expressions of the form $\sum_{\vec{x}, \vec{y} \in H^m} \mathtt{wp}(\vec{g}, \vec{x}, \vec{y}) f_2(\vec{x}) f_3(\vec{y})$ are rewritten as $\sum_{\vec{x} \in H^m} f_2(\vec{x}) \cdot h_{\vec{g}}(\vec{x})$, where $h_{\vec{g}}(\vec{x}) = \sum_{\vec{y} \in H^m} \mathtt{wp}(\vec{g}, \vec{x}, \vec{y}) f_3(\vec{y})$, we cannot assume this to be possible in our setting, since $f_2(\vec{x}) \in R$ might not commute with $\mathtt{wp}(\vec{g}, \vec{x}, \vec{y})$.

Instead, we start by using the algorithm `SetupX` (Figure 9), which substitutes the $\vec{\mathtt{Z}}, \vec{\mathtt{U}}, \vec{\mathtt{W}}$ variables in the LUTs of $\widehat{\mathtt{mult}}_{\mathtt{L}}$, $\widehat{\mathtt{add}}_{\mathtt{L}}$, $\widehat{\mathtt{mult}}_{\mathtt{R}}$ and $\widehat{\mathtt{add}}_{\mathtt{R}}$ with their corresponding challenges. Next, applying `Function_Evaluations` (Figure 7) to the (updated) LUTs, we can produce LMLEs in the $\vec{\mathtt{X}}$ variables for $\hat{V}_L$ (in Step 3) and $\widehat{\mathtt{mult}}_{\mathtt{L}}, \widehat{\mathtt{add}}_{\mathtt{L}}$ (which happens within `Sumcheck_X_Left`). Given two multi-linear polynomials $f(\mathtt{X}), g(\mathtt{X})$, we know that we can compute the sum-check protocol on their product $f(\mathtt{X}) \cdot g(\mathtt{X})$ in linear time [Tha13]. That is what we do in algorithms `Sumcheck_X_Left` (Figure 10) and `Sumcheck_X_Right` (Figure 11), where we compute, in linear time and without reordering its terms, the sum-check messages for $\sum_{\vec{x},\vec{y}\in H^m} \mathtt{wp}(\vec{g},\vec{x},\vec{y}) f_2(\vec{x}) f_3(\vec{y})$ corresponding to the $\vec{\mathtt{X}}$ variables. The key observation for the latter two algorithms is that, for $i \in [m]$, $\vec{x} \in H^{m-i}$ and $\mathtt{wp} \in \{\widehat{\mathtt{add}}_{\mathtt{L}}, \widehat{\mathtt{mult}}_{\mathtt{L}}, \widehat{\mathtt{add}}_{\mathtt{R}}, \widehat{\mathtt{mult}}_{\mathtt{R}}\}$, the set

$$\mathcal{N}^i_{\vec{x}} = \{\vec{y} \in H^m : \exists \vec{z} \in H^m, (x_1,\ldots,x_i) \in H^i \ s.t.\ \mathtt{wp}(\vec{z},(x_1,\ldots,x_i),\vec{x},\vec{y}) \neq 0\},$$

is s.t. $\sum_{\vec{x}\in H^{m-i}} |\mathcal{N}^i_{\vec{x}}| \in O(2^{m-i})$. We exploit the sparseness of our wiring predicates to keep an $O(2^m)$-time prover without reordering the terms of Eq. (9).

$\vec{\mathtt{Y}}$ variables (Steps 7-8). `Setup_Y` (Figure 12) substitutes the $\vec{\mathtt{X}}$ variables with $\vec{r}_{[2m+1,3m]} \in A^m$ in the LUTs of $\widehat{\mathtt{mult}}_{\mathtt{L}}, \widehat{\mathtt{add}}_{\mathtt{L}}$, so that $\mathcal{P}$ obtains values $\widehat{\mathtt{add}}_{\mathtt{L}}(\vec{g},\vec{r}_{[m+1,2m]},\vec{r}_{[2m+1,3m]})$ and $\widehat{\mathtt{mult}}_{\mathtt{L}}(\vec{g},\vec{r}_{[m+1,2m]},\vec{r}_{[2m+1,3m]})$. Applying `Function_Evaluations` (Figure 7) to the LUT of $\hat{V}_R$ and the LUTs of $\widehat{\mathtt{mult}}_{\mathtt{R}}$ and $\widehat{\mathtt{add}}_{\mathtt{R}}$ (which were previously updated in `SetupX`, Figure 9), we can produce the different RMLEs in the $\vec{\mathtt{Y}}$ variables that are required for the execution of `Sumcheck_Y` (Figure 13): For $i \in [m]$, $\vec{y} \in H^{m-i}$, polynomials $\hat{V}_R(\vec{r}_{[3m+1,3m+i-1]}, \mathtt{Y}_i, \vec{y})$, $\widehat{\mathtt{add}}_{\mathtt{R}}(\vec{r}_{[3m+1,3m+i-1]}, \mathtt{Y}_i, \vec{y})$ and $\widehat{\mathtt{mult}}_{\mathtt{R}}(\vec{r}_{[3m+1,3m+i-1]}, \mathtt{Y}_i, \vec{y})$.

5.5 Putting everything together

Figure 4 contains the description of our doubly-efficient interactive proof in the setting where A is a commutative, R.D. set. The following theorem captures its complexity and soundness, which follow from the analysis of its building blocks throughout this section. Remember that the $\{0,1\} \subset A$ condition is not a strict requirement, but a writing simplification. We could instead work with polynomials in $R_{A\cup\{0,1\}}[\vec{\mathtt{U}},\vec{\mathtt{W}},\vec{\mathtt{X}},\vec{\mathtt{Y}}]_{\leq 2,\leq 2}$ rather than $R_A[\vec{\mathtt{U}},\vec{\mathtt{W}},\vec{\mathtt{X}},\vec{\mathtt{Y}}]_{\leq 2,\leq 2}$, since if A is a commutative set, so is $A \cup \{0,1\}$.

Theorem 5. *Let R be a ring and $A \subset R$ a commutative, regular difference set such that $\{0,1\} \subset A$ Let $C : R^n \to R^k$ be a depth-D layered arithmetic circuit. There is an interactive proof for C with soundness error $O(D \log|C|/|A|)$. Its round complexity is $O(D\log|C|)$ and it communicates $O(D\log|C|)$ elements in R. In terms of operations in R, the prover complexity is $O(|C|)$ and the verifier complexity is $O(n + k + D\log|C| + T)$, where T is the optimal time to evaluate every wiring predicate $(\widehat{\mathtt{add}}^{(i)}_L, \widehat{\mathtt{mult}}^{(i)}_L, \widehat{\mathtt{add}}^{(i)}_R, \widehat{\mathtt{mult}}^{(i)}_R)$. For log-space uniform circuits, $T = \mathsf{poly}\log(|C|)$ and hence the IP is succinct.*

Proof. Completeness follows from the completeness of the sumcheck protocol and the complexity metrics follow from the analysis of previous building blocks.

Regarding soundness, assume that $C(\mathtt{inp}) \neq \mathtt{out}$. This means that the prover must have sent an incorrect message at some point, either within one of the D sum-check executions or otherwise some value(s) among $\{\hat{V}^{(i+1)}_L(\chi^{(i)}), \hat{V}^{(i+1)}_R(\psi^{(i)})\}_{i=0}^{D-1}$. For a given layer i, the probability that the former goes unnoticed is at most $8s_i \cdot |A|^{-1}$, whereas the probability that the latter is undetected is at most $2 \cdot |A|^{-1}$ (see Lemma 11). Thus, if we denote by E_i the event that any of these two things happen at layer i, we have that $\Pr[E_i] \in O(s_i/|A|)$. Applying the union bound, we obtain that

$$\Pr[C(\mathtt{inp}) \neq \mathtt{out} \wedge \mathcal{V} \text{ outputs } 1] = \Pr[\cup_{i=1}^D E_i] \leq \sum_{i=1}^D \Pr[E_i] \leq \sum_{i=1}^D O(s_i/|A|) \leq O(D\log(|C|) \cdot |A|^{-1})$$

∎

Doubly-efficient interactive proof over a non-commutative ring

Let R be a ring and A a commutative regular difference set such that $\{0,1\} \subset A \subset R$. Let $C : R^n \to R^k$ be a layered arithmetic circuit over R with depth D. Without loss of generality, we assume that n and k are powers of 2.
Input: Circuit input `inp` and claimed output `out`.
Output: Accept or reject.

- Compute $\hat{V}_L^{(0)}(\mathtt{X})$ as the LMLE of `out`. $\mathcal{V}$ chooses a random $\gamma \in A^{s_0}$ and sends it to $\mathcal{P}$. Both parties compute $\hat{V}_L^{(0)}(\gamma)$.
- Run a sum-check protocol on Equation (6) as described in[a] Appendix B.2. Let $\chi^{(0)}, \psi^{(0)}, \omega^{(0)}$ denote the challenge vectors corresponding to the $\vec{\mathtt{X}}$, $\vec{\mathtt{Y}}$ and $\vec{\mathtt{W}}$ variables within that execution. $\mathcal{P}$ sends $\hat{V}_L^{(1)}(\chi^{(0)})$ and $\hat{V}_R^{(1)}(\psi^{(0)})$ to $\mathcal{V}$.
- $\mathcal{V}$ queries their oracles for $\widehat{\mathtt{mult}_\mathtt{L}}^{(1)}(\gamma, \chi^{(0)}, \omega^{(0)})$ and $\widehat{\mathtt{add}_\mathtt{L}}^{(1)}(\gamma, \chi^{(0)}, \omega^{(0)})$, so as to check that $\widehat{\mathtt{add}_\mathtt{L}}^{(1)}(\gamma, \chi^{(0)}, \omega^{(0)}) \cdot (\hat{V}_L^{(1)}(\chi^{(0)}) + \hat{V}_R^{(1)}(\psi^{(0)})) + \widehat{\mathtt{mult}_\mathtt{L}}^{(1)}(\gamma, \chi^{(0)}, \omega^{(0)}) \cdot (\hat{V}_L^{(1)}(\chi^{(0)}) \cdot \hat{V}_R^{(1)}(\psi^{(0)}))$ equals the last message of the sumcheck execution.
- For circuit layers $i = 1, \ldots, D-1$:
  - $\mathcal{V}$ samples $\alpha^{(i)}, \beta^{(i)} \in A$ and sends them to $\mathcal{P}$. They run a sumcheck protocol on Equation (9) as described in Figure 3. Let $\chi^{(i)}, \psi^{(i)}$ denote the challenge vectors corresponding to the $\vec{\mathtt{X}}$, and $\vec{\mathtt{Y}}$ variables within that execution. At the end of the protocol, $\mathcal{P}$ sends $\hat{V}_L^{(i+1)}(\chi^{(i)})$ and $\hat{V}_R^{(i+1)}(\psi^{(i)})$ to $\mathcal{V}$, so that $\mathcal{V}$ can check the validity of the last message in the sumcheck execution. If the check passes, they proceed to the $(i+1)$-th layer, otherwise, $\mathcal{V}$ outputs reject and aborts.
- At the input layer D, $\mathcal{V}$ has received two claims $\hat{V}_L^{(D)}(\chi^{(D-1)})$ and $\hat{V}_R^{(D)}(\psi^{(D-1)})$. $\mathcal{V}$ queries the evaluation oracles of $\hat{V}_L^{(D)}$ and $\hat{V}_R^{(D)}$ at $\chi^{(D-1)}$ and $\psi^{(D-1)}$ respectively, and checks that they equal the sumcheck claims. If they do, $\mathcal{V}$ outputs accept, otherwise, $\mathcal{V}$ outputs reject.

[a] The most detailed version of $\mathcal{P}$'s algorithm is not provided in there. It is, anyway, a simpler version of the one for Equation (9) in Section 5.4.

Fig. 4. Doubly-efficient IP over a ring containing a commutative, regular difference set.

References

ACD$^+$19. Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via galois rings. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 471–501. Springer, Heidelberg, December 2019.

AIK10. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP 2010, Part I*, volume 6198 of *LNCS*, pages 152–163. Springer, Heidelberg, July 2010.

BCFK21. Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Heidelberg, May 2021.

BCS21. Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 742–773, Virtual Event, August 2021. Springer, Heidelberg.

BFL91. László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991.

BS10. Eduardo Bayro-Corrochano and Gerik Scheuermann. *Geometric algebra computing: in engineering and computer science*. Springer Science & Business Media, 2010.

CCKP19. Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. Verifiable computing for approximate computation. Cryptology ePrint Archive, Report 2019/762, 2019. https://eprint.iacr.org/2019/762.

CDI$^+$13. Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 185–202. Springer, Heidelberg, August 2013.

CFIK03. Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 596–613. Springer, Heidelberg, May 2003.

CMT12. Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In Shafi Goldwasser, editor, *ITCS 2012*, pages 90–112. ACM, January 2012.

DLS20. Anders P. K. Dalskov, Eysa Lee, and Eduardo Soria-Vazquez. Circuit amortization friendly encodingsand their application to statistically secure multiparty computation. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 213–243. Springer, Heidelberg, December 2020.

ES21. Daniel Escudero and Eduardo Soria-Vazquez. Efficient information-theoretic multi-party computation over non-commutative rings. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 335–364, Virtual Event, August 2021. Springer, Heidelberg.

Fre79. Rūsiņš Freivalds. Fast probabilistic algorithms. In *International Symposium on Mathematical Foundations of Computer Science*, pages 57–69. Springer, 1979.

GKR15. Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

GNS21. Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: Snarks for ring arithmetic. Cryptology ePrint Archive, Report 2021/322, 2021. https://eprint.iacr.org/2021/322.

GNSW07. Herman Geuvers, Milad Niqui, Bas Spitters, and Freek Wiedijk. Constructive analysis, types and exact real numbers. *Mathematical Structures in Computer Science*, 17(1):3–36, 2007.

HL75. Thomas D Howell and Jean-Claude Lafon. The complexity of the quaternion product. Technical report, Cornell University, 1975.

HR18. Justin Holmgren and Ron Rothblum. Delegating computations with (almost) minimal time and space overhead. In Mikkel Thorup, editor, *59th FOCS*, pages 124–135. IEEE Computer Society Press, October 2018.

HY11. Pavel Hrubeš and Amir Yehudayoff. Arithmetic complexity in ring extensions. *Theory of Computing*, 7(1):119–129, 2011.

IK04. Yuval Ishai and Eyal Kushilevitz. On the hardness of information-theoretic multiparty computation. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 439–455. Springer, Heidelberg, May 2004.

JMRO22. J. Jin, E. McMurtry, B. Rubinstein, and O. Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1547–1547, Los Alamitos, CA, USA, may 2022. IEEE Computer Society.

LFKN92. Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.

LXZ21. Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2968–2985. ACM Press, November 2021.

Mei13. Or Meir. Ip= pspace using error-correcting codes. *SIAM Journal on Computing*, 42(1):380–403, 2013.

Mir12. Ilya Mironov. On significance of the least significant bits for differential privacy. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 650–661. ACM Press, October 2012.

QBC13. Guillaume Quintin, Morgan Barbier, and Christophe Chabot. On generalized reed–solomon codes over commutative and noncommutative rings. *IEEE transactions on information theory*, 59(9):5882–5897, 2013.

RAS08. Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action MACH a spatio-temporal maximum average correlation height filter for action recognition. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

SS10. René Schott and G. Stacey Staples. Reductions in computational complexity using clifford algebras. *Advances in applied Clifford algebras*, 20(1):121–140, 2010.

Tha13. Justin R Thaler. *Practical verified computation with streaming interactive proofs*. PhD thesis, 2013.

XZZ$^+$19. Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Heidelberg, August 2019.

ZLW$^+$21. Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 159–177, 2021.

A Appendix to Section 5.4: Linear time Prover for Equation (9)

Here we provide the protocols for Section 5.4 that could not fit in the main body.

Algorithm $\texttt{Function\_Evaluations\_}\{\texttt{U},\texttt{W}\}(f, \boldsymbol{T}_f, r_1, \ldots, r_{2m})$

Input: $f \in R_A[\vec{\texttt{U}}, \vec{\texttt{W}}, \vec{\texttt{X}}, \vec{\texttt{Y}}]_{\leq 2, \leq 2}$ from Equation (9), initial lookup table $\boldsymbol{T}_f$, random challenges $\vec{r} = (r_1, \ldots, r_{2m}) \in A^{2m}$.
Output: Polynomials $f(r_1, \ldots, r_{i-1}, \texttt{U}_i, \vec{t}_{i,b}) \in R_A[\texttt{U}_i]_{0,\leq 1}$ for $i \in [1, m]$ and $\vec{t}_{i,b} \in H^{4m-i}$. Polynomials $f(r_1, \ldots, r_{i-1}, \texttt{W}_{i-m}, \vec{t}_{i,b}) \in R_A[\texttt{W}_{i-m}]_{\leq 1, 0}$ for $i \in [m+1, 2m]$ and $\vec{t}_{i,b} \in H^{4m-i}$.

– For $i \in [1, 2m]$ let $\vec{0}$ be the length-$(i-1)$ zero vector and $r_{[1,i)} = (r_1, \ldots, r_{i-1})$.
  - For $i \in [1, m]$: For $b \in H$, and for every $\vec{y} \in H^m$, $\vec{x}_{(i,m]} \in H^{m-i}$, $\vec{x}_{[1,i)} \in H^{i-1}$, define $\vec{t}_{i,b} = (\vec{x}_{(i,m]}, \vec{y}, \vec{x}_{[1,i)}, b, \vec{x}_{(i,m]}, \vec{y}) \in H^{4m-i}$ and do:

$$\begin{aligned} f(r_{[1,i)}, \texttt{U}_i, \vec{t}_{i,b}) &\leftarrow \boldsymbol{T}_f[\vec{0}, 1, \vec{t}_{i,b}] \cdot \texttt{U}_i + \boldsymbol{T}_f[\vec{0}, 0, \vec{t}_{i,b}] \cdot (1 - \texttt{U}_i) \\ \boldsymbol{T}_f[\vec{0}, 0, \vec{t}_{i,b}] &\leftarrow \boldsymbol{T}_f[\vec{0}, 1, \vec{t}_{i,b}] \cdot r_i + \boldsymbol{T}_f[\vec{0}, 0, \vec{t}_{i,b}] \cdot (1 - r_i) \end{aligned} \tag{10}$$

  - For $i \in [m+1, 2m]$: For $b \in H$, and for every $\vec{x} \in H^m$, $\vec{y}_{(i-m,m]} \in H^{2m-i}$, $\vec{y}_{[1,i-m)} \in H^{i-m-1}$, define $\vec{t}_{i,b} = (\vec{y}_{(i-m,m]}, \vec{x}, \vec{y}_{[1,i-m)}, b, \vec{y}_{(i-m,m]}) \in H^{4m-i}$ and do:

$$\begin{aligned} f(r_{[1,i)}, \texttt{W}_{i-m}, \vec{t}_{i,b}) &\leftarrow \texttt{W}_{i-m} \cdot \boldsymbol{T}_f[\vec{0}, 1, \vec{t}_{i,b}] + (1 - \texttt{W}_{i-m}) \cdot \boldsymbol{T}_f[\vec{0}, 0, \vec{t}_{i,b}] \\ \boldsymbol{T}_f[\vec{0}, 0, \vec{t}_{i,b}] &\leftarrow r_i \cdot \boldsymbol{T}_f[\vec{0}, 1, \vec{t}_{i,b}] + (1 - r_i) \cdot \boldsymbol{T}_f[\vec{0}, 0, \vec{t}_{i,b}] \end{aligned} \tag{11}$$

– Let $\mathcal{F}$ contain all polynomials in $R_A[\texttt{U}_i]_{0,\leq 1}$ (resp. $R_A[\texttt{W}_i]_{\leq 1,0}$) defined at Equation (10) (resp. Equation (11)) throughout the execution.

Fig. 5. Evaluations of toast multi-linear polynomials prior to sum-check.

B Doubly-efficient IP over non-commutative rings: Better round complexity

We introduce a variant of the protocol from Section 5 which has a better round complexity ($3m$ rounds per layer of the circuit, where m is the width of the layer) at the cost of a more expensive communication complexity ($18m$ ring elements). The main idea behind this improvement is explained in Appendix B.1 and the high level description of this variant appears Figure 14. The necessary sum-check protocols appear in Appendix B.2 and B.3. $\mathcal{P}$'s most detailed algorithms are not provided in those two appendices, but they are a simpler version of the one used in Section 5.4.

B.1 4-to-2 reduction

The GKR protocol starts with a simple layer consistency equation, which relates the output layer with layer 1 in the circuit according to the following equation:

$$\hat{V}_L^{(0)}(\gamma) = \sum_{x,y \in \{0,1\}^{s_1}} \Big(\widehat{\texttt{mult}_\texttt{L}}^{(1)}(\gamma, x, y) \cdot \big(\hat{V}_L^{(1)}(x) \cdot \hat{V}_R^{(1)}(y)\big) + \\ + \widehat{\texttt{add}_\texttt{L}}^{(1)}(\gamma, x, y) \cdot \big(\hat{V}_L^{(1)}(x) + \hat{V}_R^{(1)}(y)\big)\Big). \tag{14}$$

```
Algorithm Sumcheck_{U,W}$(f, \boldsymbol{T}_f, r_1, \ldots, r_{2m})$

Input: $f \in R_A[\vec{\mathtt{U}}, \vec{\mathtt{W}}, \vec{\mathtt{X}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 2}$ from Equation (9), initial lookup table $\boldsymbol{T}_f$, random challenges $\vec{r} = (r_1, \ldots, r_{2m}) \in A^{2m}$.
Output: First $2m$ sumcheck messages for $f$.

  – $\mathcal{F} \leftarrow$ Function_Evaluations_{U,W}$(f, \boldsymbol{T}_f, r_1, \ldots, r_{2m})$
  – For $i \in [1, m]$ and $b \in H$, define $\vec{t}_{i,b} = (\vec{x}_{(i,m]}, \vec{y}, \vec{x}_{[1,i)}, b, \vec{x}_{(i,m]}, \vec{y}) \in H^{4m-i}$ for every $\vec{y} \in H^m$, $\vec{x}_{(i,m]} \in H^{m-i}$, $\vec{x}_{[1,i)} \in H^{i-1}$. Compute and send:
        $$g_{i,b}(\mathtt{U}_i) = \sum_{\vec{t}_{i,b} \in H^{4m-i}} f(r_1, \ldots, r_{i-1}, \mathtt{U}_i, \vec{t}_{i,b}).$$
  – For $i \in [m+1, 2m]$ and $b \in H$, define $\vec{t}_{i,b} = (\vec{y}_{(i-m,m]}, \vec{x}, \vec{y}_{[1,i-m)}, b, \vec{y}_{(i-m,m]}) \in H^{4m-i}$ for every $\vec{x} \in H^m$, $\vec{y}_{(i-m,m]} \in H^{2m-i}$, $\vec{y}_{[1,i-m)} \in H^{i-m-1}$. Compute and send:
        $$g_{i,b}(\mathtt{W}_{i-m}) = \sum_{\vec{t}_{i,b} \in H^{4m-i}} f(r_1, \ldots, r_{i-1}, \mathtt{W}_{i-m}, \vec{t}_{i,b}).$$
  – Return $\{g_{i,b}(\mathtt{U}_i)\}_{b \in H, i \in [1,m]}$, $\{g_{i,b}(\mathtt{W}_{i-m})\}_{b \in H, i \in [m+1,2m]}$.
```

Fig. 6. Sum-check polynomials for the block of $\vec{\mathtt{U}}, \vec{\mathtt{W}}$ variables.

```
Algorithm Function_Evaluations$(f, \boldsymbol{T}_f, r_1, \ldots, r_m)$

Input: Lookup table $\boldsymbol{T}_f$ corresponding to either $f \in R_A[\mathtt{X}_1, \ldots, \mathtt{X}_m]_{\leq 1, 0}$ or $f \in R_A[\mathtt{Y}_1, \ldots, \mathtt{Y}_m]_{0, \leq 1}$, random challenges $\vec{r} = (r_1, \ldots, r_m) \in A^m$.
Output: Polynomials $f(r_1, \ldots, r_{i-1}, \mathtt{X}_i, \vec{b}) \in R_A[\mathtt{X}_i]$ (or $f(r_1, \ldots, r_{i-1}, \mathtt{Y}_i, \vec{b}) \in R_A[\mathtt{Y}_i]$) for $i \in [1, m]$ and $\vec{b} \in \{0,1\}^{m-i}$.

  – For $i = 1, \ldots, m$ let $\vec{0}$ be the length-$(i-1)$ zero vector and do:
      • For $\vec{b} = (b_{m-i}, \ldots, b_1) \in H^{m-i}$ do:
          ∗ If $f \in R_A[\mathtt{Y}_1, \ldots, \mathtt{Y}_m]_{0, \leq 1}$, define:
              $$\begin{aligned} f(r_1, \ldots, r_{i-1}, \mathtt{Y}_i, \vec{b}) &\leftarrow \big(\boldsymbol{T}_f[\vec{0}, 1, \vec{b}] - \boldsymbol{T}_f[\vec{0}, 0, \vec{b}]\big) \cdot \mathtt{Y}_i + \boldsymbol{T}_f[\vec{0}, 0, \vec{b}] \\ \boldsymbol{T}_f[\vec{0}, 0, \vec{b}] &\leftarrow \big(\boldsymbol{T}_f[\vec{0}, 1, \vec{b}] - \boldsymbol{T}_f[\vec{0}, 0, \vec{b}]\big) \cdot r_i + \boldsymbol{T}_f[\vec{0}, 0, \vec{b}] \end{aligned} \quad (12)$$
          ∗ Else (i.e. if $f \in R_A[\mathtt{X}_1, \ldots, \mathtt{X}_m]_{\leq 1, 0}$), define:
              $$\begin{aligned} f(r_1, \ldots, r_{i-1}, \mathtt{X}_i, \vec{b}) &\leftarrow \mathtt{X}_i \cdot \big(\boldsymbol{T}_f[\vec{0}, 1, \vec{b}] - \boldsymbol{T}_f[\vec{0}, 0, \vec{b}]\big) + \boldsymbol{T}_f[\vec{0}, 0, \vec{b}] \\ \boldsymbol{T}_f[\vec{0}, 0, \vec{b}] &\leftarrow r_i \cdot \big(\boldsymbol{T}_f[\vec{0}, 1, \vec{b}] - \boldsymbol{T}_f[\vec{0}, 0, \vec{b}]\big) + \boldsymbol{T}_f[\vec{0}, 0, \vec{b}] \end{aligned} \quad (13)$$
  – Let $\mathcal{F}$ contain all polynomials in $R_A[\mathtt{X}_i]_{\leq 1, 0}$ (resp. $R_A[\mathtt{Y}_i]_{0, \leq 1}$) defined at Equation (13) (resp. Equation (12)) throughout the execution.
```

Fig. 7. Evaluations of toast multi-linear polynomials for sum-check.

```
Algorithm Precompute$(g_1, \ldots, g_\ell)$

Input: Random challenge $\vec{g} = (g_1, \ldots, g_\ell) \in A^\ell$.
Output: Lookup table $\{\boldsymbol{T}_{\vec{g}}[\vec{b}]\}_{\vec{b} \in \{0,1\}^\ell}$ containing the evaluations of $I(\vec{g}, \vec{b}) = \prod_{i=1}^{m} \big(g_i \cdot b_i + (1 - g_i) \cdot (1 - b_i)\big)$.

  – Set $\boldsymbol{T}_{\vec{g}}[\vec{0}] \leftarrow (1 - g_1)$ and $\boldsymbol{T}_{\vec{g}}[0, \ldots, 0, 1] \leftarrow g_1$.
  – For $i = 1, \ldots, \ell - 1$, do:
      • For $(b_i, \ldots, b_1) \in \{0,1\}^i$, do:
          ∗ $\boldsymbol{T}_{\vec{g}}[\vec{0}, 0, b_i, \ldots, b_1] \leftarrow \boldsymbol{T}_{\vec{g}}[\vec{0}, b_i, \ldots, b_1] \cdot (1 - g_{i+1})$.
          ∗ $\boldsymbol{T}_{\vec{g}}[\vec{0}, 1, b_i, \ldots, b_1] \leftarrow \boldsymbol{T}_{\vec{g}}[\vec{0}, b_i, \ldots, b_1] \cdot g_{i+1}$.
```

Fig. 8. Computing LUT for identity polynomial evaluated at a challenge.

Algorithm $\texttt{Setup\_X}(f_1, \boldsymbol{T}_{f_1}, f_2, \boldsymbol{T}_{f_2}, f_3, \boldsymbol{T}_{f_3}, f_4, \boldsymbol{T}_{f_4}, \vec{\chi}, \vec{\psi}, \vec{r})$

Input: Multi-linear $f_1(z,x,y), f_2(z,x,y) \in R_A[\vec{\texttt{X}}, \vec{\texttt{W}}, \vec{\texttt{Z}}]_{\le 1,0}$, $f_3(z,x,y), f_4(z,x,y) \in R_A[\vec{\texttt{Y}}, \vec{\texttt{U}}, \vec{\texttt{Z}}]_{\le 1,0}$ and their initial look-up tables. Random challenges[a] $\vec{\chi}, \vec{\psi} \in A^m$, $\vec{r} \in A^{2m}$.
Output: Look-up tables $\boldsymbol{T}_{f_i}$ for the block of $\vec{\texttt{X}}$ variables.

- $\boldsymbol{T}_{\vec{\chi}}[\vec{z}] \leftarrow \texttt{Precompute}(\vec{\chi})$.
- $\boldsymbol{T}_{\vec{\psi}}[\vec{z}] \leftarrow \texttt{Precompute}(\vec{\psi})$.
- $\boldsymbol{T}_{\vec{r}_{[1,m]}}[\vec{x}] \leftarrow \texttt{Precompute}(\vec{r}_{[1,m]})$.
- $\boldsymbol{T}_{\vec{r}_{[m+1,2m]}}[\vec{y}] \leftarrow \texttt{Precompute}(\vec{r}_{[m+1,2m]})$.
- $\forall \vec{x}, \vec{y} \in \{0,1\}^m$, set $\boldsymbol{T}_{f_1}[\vec{x}] = \boldsymbol{T}_{f_2}[\vec{x}] = \boldsymbol{T}_{f_3}[\vec{y}] = \boldsymbol{T}_{f_4}[\vec{y}] = 0$.
- For $i = 1, 2$ and for every $(\vec{z}, \vec{x}, \vec{y}) \in H^{3m}$ such that $f_i(\vec{z}, \vec{x}, \vec{y}) \neq 0$, do:
$$\boldsymbol{T}_{f_i}[\vec{x}] \leftarrow \boldsymbol{T}_{f_i}[\vec{x}] + \boldsymbol{T}_{\vec{\chi}}[\vec{z}] \cdot \boldsymbol{T}_{\vec{r}_{[m+1,2m]}}[\vec{y}] \cdot f_i(\vec{z}, \vec{x}, \vec{y}).$$
- For $i = 3, 4$ and for every $(\vec{z}, \vec{x}, \vec{y}) \in H^{3m}$ such that $f_i(\vec{z}, \vec{x}, \vec{y}) \neq 0$, do:
$$\boldsymbol{T}_{f_i}[\vec{y}] \leftarrow \boldsymbol{T}_{f_i}[\vec{y}] + f_i(\vec{z}, \vec{x}, \vec{y}) \cdot \boldsymbol{T}_{\vec{\psi}}[\vec{z}] \cdot \boldsymbol{T}_{\vec{r}_{[1,m]}}[\vec{x}].$$
- Return $\boldsymbol{T}_{f_1}[x], \boldsymbol{T}_{f_2}[x], \boldsymbol{T}_{f_3}[y], \boldsymbol{T}_{f_4}[y]$.

[a] For a shorter write-up, we describe this algorithm as if $\vec{z}, \vec{x}, \vec{y} \in \{0,1\}^m$. In practice, since $\vec{z}$ comes from a different layer, it might have different length.

Fig. 9. Substituting $\vec{\texttt{Z}}, \vec{\texttt{U}}, \vec{\texttt{W}}$ in LUTs with their corresponding challenges.

Algorithm $\texttt{Sumcheck\_X\_Left}(\widehat{\texttt{mult}}_{\texttt{L}}, \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{L}}}(\vec{x}), \widehat{\texttt{add}}_{\texttt{L}}, \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{L}}}(\vec{x}), \mathcal{F}_{\hat{V}_L}, \hat{V}_R, \boldsymbol{T}_{\hat{V}_R}(\vec{y}), \vec{r}_{[2m+1,3m]})$

Input: Parse $\vec{\texttt{X}} = (\texttt{X}_1, \ldots, \texttt{X}_m)$ and $\vec{\texttt{Y}} = (\texttt{Y}_1, \ldots, \texttt{Y}_m)$. Toast polynomials $\widehat{\texttt{mult}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,2m]}, \vec{\texttt{X}}), \widehat{\texttt{add}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,2m]}, \vec{\texttt{X}}), \hat{V}_L(\vec{\texttt{X}}) \in R_A[\vec{\texttt{X}}]_{\le 1,0}$ and $\hat{V}_R(\vec{\texttt{Y}}) \in R_A[\vec{\texttt{Y}}]_{0,\le 1}$, given by their lookup tables $\boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{L}}}(x), \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{L}}}(x), \boldsymbol{T}_{\hat{V}_R}(y)$ containing all evaluations at H^m. Random challenges $\vec{r}_{[2m+1,3m]} \in A^m$ and table $\mathcal{F}_{\hat{V}_L} \leftarrow \texttt{Function\_Evaluations}(\hat{V}_L, \boldsymbol{T}_{\hat{V}_L}, r_{2m+1}, \ldots, r_{3m})$.
Output: $2m$ partial sumcheck messages, half for $\sum_{\vec{x}, \vec{y} \in H^m} g_{2m+i}^{\texttt{mult}_{\texttt{L}}}(\vec{x}, \vec{y})$ and half for $\sum_{\vec{x}, \vec{y} \in H^m} g_{2m+i}^{\texttt{add}_{\texttt{L}}}(\vec{x}, \vec{y})$. Each message is a polynomial in $R_A[\texttt{X}_i]_{\le 2,0}$.

- $\mathcal{F}_{\widehat{\texttt{mult}}_{\texttt{L}}} \leftarrow \texttt{Function\_Evaluations}(\widehat{\texttt{mult}}_{\texttt{L}}, \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{L}}}, r_{2m+1}, \ldots, r_{3m})$.
- $\mathcal{F}_{\widehat{\texttt{add}}_{\texttt{L}}} \leftarrow \texttt{Function\_Evaluations}(\widehat{\texttt{add}}_{\texttt{L}}, \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{L}}}, r_{2m+1}, \ldots, r_{3m})$.
- Compute $Y = \sum_{\vec{y} \in H^m} \hat{V}_R(\vec{y})$ from $\boldsymbol{T}_{\hat{V}_R}(\vec{y})$ and store it for the next steps.
- For $i \in [m]$, compute as follows:
$$g_{2m+i}^{\texttt{mult}_{\texttt{L}}}(\texttt{X}_i) = \sum_{\vec{x} \in H^{m-i}} \widehat{\texttt{mult}}_{\texttt{L}}(\vec{r}_{[2m+1,2m+i-1]}, \texttt{X}_i, \vec{x}) \cdot \hat{V}_L(\vec{r}_{[2m+1,2m+i-1]}, \texttt{X}_i, \vec{x}) \cdot Y.$$
$$g_{2m+i}^{\texttt{add}_{\texttt{L}}}(\texttt{X}_i) = \sum_{\vec{x} \in H^{m-i}} \widehat{\texttt{add}}_{\texttt{L}}(\vec{r}_{[2m+1,2m+i-1]}, \texttt{X}_i, \vec{x}) \cdot \Big(\hat{V}_L(\vec{r}_{[2m+1,2m+i-1]}, \texttt{X}_i, \vec{x}) + Y\Big).$$
- Return each polynomial $\{g_{2m+1}^{\texttt{mult}_{\texttt{L}}}(\texttt{X}_1), g_{2m+1}^{\texttt{add}_{\texttt{L}}}(\texttt{X}_1), \ldots, g_{3m}^{\texttt{mult}_{\texttt{L}}}(\texttt{X}_m), g_{3m}^{\texttt{add}_{\texttt{L}}}(\texttt{X}_m)\}$.

Fig. 10. Sumcheck polynomials for $\vec{\texttt{X}}$ variables and "$\hat{V}_L^{(i)}$-side" of Equation (9).

Algorithm $\texttt{Sumcheck\_X\_Right}(\widehat{\texttt{mult}}_{\texttt{R}}, \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{R}}}(\vec{y}), \widehat{\texttt{add}}_{\texttt{R}}, \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{R}}}(\vec{y}), \mathcal{F}_{\hat{V}_L}, \hat{V}_R, \boldsymbol{T}_{\hat{V}_R}(\vec{y}), \vec{r}_{[2m+1,3m]})$

Input: Parse $\vec{\texttt{X}} = (\texttt{X}_1, \ldots, \texttt{X}_m)$ and $\vec{\texttt{Y}} = (\texttt{Y}_1, \ldots, \texttt{Y}_m)$. Table $\mathcal{F}_{\hat{V}_L}$ with all evaluations of $\hat{V}_L(\vec{\texttt{X}}) \in R_A[\vec{\texttt{X}}]_{\leq 1,0}$ in H^m. Toast polynomials $\widehat{\texttt{mult}}_{\texttt{R}}(\vec{\psi}, \vec{r}_{[1,m]}, \vec{\texttt{Y}}), \widehat{\texttt{add}}_{\texttt{R}}(\vec{\psi}, \vec{r}_{[1,m]}, \vec{\texttt{Y}}), \hat{V}_R(\vec{\texttt{Y}}) \in R_A[\vec{\texttt{Y}}]_{0,\leq 1}$ given by their lookup tables $\boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{R}}}(\vec{y}), \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{R}}}(\vec{y}), \boldsymbol{T}_{\hat{V}_R}(\vec{y})$, containing all evaluations at H^m. Random challenges $\vec{r}_{[2m+1,3m]} \in A^m$ and table $\mathcal{F}_{\hat{V}_L} \leftarrow \texttt{Function\_Evaluations}(\hat{V}_L, \boldsymbol{T}_{\hat{V}_L}, r_{2m+1}, \ldots, r_{3m})$.
Output: $2m$ partial sumcheck messages, half for $\sum_{\vec{x},\vec{y} \in H^m} g^{\texttt{mult}_{\texttt{R}}}_{2m+i}(\vec{x}, \vec{y})$ and half for $\sum_{\vec{x},\vec{y} \in H^m} g^{\texttt{add}_{\texttt{R}}}_{2m+i}(\vec{x}, \vec{y})$. Each message is a polynomial in $R_A[\texttt{X}_i]_{\leq 1,0}$.

- For $i \in [m]$, compute $g^{\texttt{mult}_{\texttt{R}}}_{2m+i} \in R_A[\texttt{X}_i]_{\leq 1,0}$ as follows. Notice $\sum_{\vec{y} \in H^m} \hat{V}_R(\vec{y}) \cdot \widehat{\texttt{mult}}_{\texttt{R}}(\vec{y})$ can be computed once and in time $O(2^m)$ from $\boldsymbol{T}_{\hat{V}_R}(\vec{y})$ and $\boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{R}}}(\vec{y})$ for all the steps.

$$g^{\texttt{mult}_{\texttt{R}}}_{2m+i}(\texttt{X}_i) = \sum_{\vec{x} \in H^{m-i}} \hat{V}_L(\vec{r}_{[2m+1,2m+i-1]}, \texttt{X}_i, \vec{x}) \cdot \Big(\sum_{\vec{y} \in H^m} \hat{V}_R(\vec{y}) \cdot \widehat{\texttt{mult}}_{\texttt{R}}(\vec{y}) \Big).$$

- For $i \in [m]$, compute $g^{\texttt{add}_{\texttt{R}}}_{2m+i} \in R_A[\texttt{X}_i]_{\leq 1,0}$ as follows. First, compute $\sum_{\vec{y} \in H^m} \widehat{\texttt{add}}_{\texttt{R}}(\vec{y})$. Next, compute $\sum_{\vec{y} \in H^m} \hat{V}_R(\vec{y}) \cdot \widehat{\texttt{add}}_{\texttt{R}}(\vec{y})$. Given those, the next expression can be computed in $O(2^{m-i})$ time.

$$g^{\texttt{add}_{\texttt{R}}}_{2m+i}(\texttt{X}_i) = \sum_{\vec{x} \in H^{m-i}} \sum_{\vec{y} \in H^m} \Big(\hat{V}_L(\vec{r}_{[2m+1,2m+i-1]}, \texttt{X}_i, \vec{x}) + \hat{V}_R(\vec{y}) \Big) \cdot \widehat{\texttt{add}}_{\texttt{R}}(\vec{y}).$$

- Return each polynomial $\{g^{\texttt{mult}_{\texttt{R}}}_{2m+1}(\texttt{X}_1), g^{\texttt{add}_{\texttt{R}}}_{2m+1}(\texttt{X}_1), \ldots, g^{\texttt{mult}_{\texttt{R}}}_{3m}(\texttt{X}_m), g^{\texttt{add}_{\texttt{R}}}_{3m}(\texttt{X}_m)\}$.

Fig. 11. Sumcheck polynomials for $\vec{\texttt{X}}$ variables and "$\hat{V}^{(i)}_R$-side" of Equation (9).

Algorithm $\texttt{Setup\_Y}(\widehat{\texttt{mult}}_{\texttt{L}}, \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{L}}}(\vec{x}), \widehat{\texttt{add}}_{\texttt{L}}, \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{L}}}(\vec{x}), \vec{r}_{[2m+1,3m]})$

Input: $\widehat{\texttt{mult}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,2m]}, \vec{\texttt{X}}),\ \widehat{\texttt{add}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,2m]}, \vec{\texttt{X}}) \in R_A[\vec{\texttt{X}}]_{\leq 1,0}$ and their look-up tables after $\texttt{Setup\_X}$ (Figure 9). Random challenge $\vec{r}_{[2m+1,3m]} \in A^m$.
Output: Values $\widehat{\texttt{mult}}_{\texttt{L}}(\vec{g}, \vec{r}_{[m+1,2m]}, \vec{r}_{[2m+1,3m]})$ and $\widehat{\texttt{add}}_{\texttt{L}}(\vec{g}, \vec{r}_{[m+1,2m]}, \vec{r}_{[2m+1,3m]})$.

- $\boldsymbol{T}_{\vec{r}_{[2m+1,3m]}}[\vec{x}] \leftarrow \texttt{Precompute}(\vec{r}_{[2m+1,3m]})$.
- Compute:

$$\widehat{\texttt{add}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,2m]}, \vec{r}_{[2m+1,3m]}) = \sum_{\vec{x} \in H} \boldsymbol{T}_{\vec{r}_{[2m+1,3m]}}[\vec{x}] \cdot \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{L}}}[\vec{x}].$$

$$\widehat{\texttt{mult}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,2m]}, \vec{r}_{[2m+1,3m]}) = \sum_{\vec{x} \in H} \boldsymbol{T}_{\vec{r}_{[2m+1,3m]}}[\vec{x}] \cdot \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{L}}}[\vec{x}].$$

- Return $\widehat{\texttt{mult}}_{\texttt{L}}(\vec{g}, \vec{r}_{[m+1,2m]}, \vec{r}_{[2m+1,3m]})$ and $\widehat{\texttt{add}}_{\texttt{L}}(\vec{g}, \vec{r}_{[m+1,2m]}, \vec{r}_{[2m+1,3m]})$.

Fig. 12. Substituting $\vec{\texttt{X}}$ with $\vec{r}_{[2m+1,3m]} \in A^m$ in the LUTs of $\widehat{\texttt{mult}}_{\texttt{L}}, \widehat{\texttt{add}}_{\texttt{L}}$.

Algorithm `Sumcheck_Y` $(\widehat{\texttt{mult}}_{\texttt{R}}, \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{R}}}(y), \widehat{\texttt{add}}_{\texttt{R}}, \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{R}}}(y), \hat{V}_R, \boldsymbol{T}_{\hat{V}_R}(y), \hat{V}_L(\vec{r}_{[2m+1,3m]}), m_L, a_L, \vec{s}, \beta)$

Input: Toast polynomials $\widehat{\texttt{mult}}_{\texttt{R}}(\vec{\psi}, \vec{r}_{[1,m]}, \vec{\texttt{Y}}), \widehat{\texttt{add}}_{\texttt{R}}(\vec{\psi}, \vec{r}_{[1,m]}, \vec{\texttt{Y}}), \hat{V}_R(\vec{\texttt{Y}}) \in R_A[\vec{\texttt{Y}}]_{0,\leq 1}$ given by their lookup tables $\boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{R}}}(\vec{y}), \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{R}}}(\vec{y})$ (after the execution of `Setup_X` (Figure 9)) and $\boldsymbol{T}_{\hat{V}_R}(\vec{y})$. Values $m_L = \alpha \cdot \widehat{\texttt{mult}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,3m]})$ and $a_L = \alpha \cdot \widehat{\texttt{add}}_{\texttt{L}}(\vec{\chi}, \vec{r}_{[m+1,3m]})$. Random challenges $\vec{s} = \vec{r}_{[3m+1,4m]} \in A^m$.
Output: Last m sumcheck messages for Equation (9). Each message is a polynomial $g_{3m+i}(\texttt{Y}_i) \in R_A[\texttt{Y}_i]_{\leq 2,0}$.

– $\mathcal{F}_{\hat{V}_R} \leftarrow$ `Function_Evaluations`$(\hat{V}_R, \boldsymbol{T}_{\hat{V}_R}, s_1, \ldots, s_m)$.
– $\mathcal{F}_{\widehat{\texttt{add}}_{\texttt{R}}} \leftarrow$ `Function_Evaluations`$(\widehat{\texttt{add}}_{\texttt{R}}, \boldsymbol{T}_{\widehat{\texttt{add}}_{\texttt{R}}}, s_1, \ldots, s_m)$.
– $\mathcal{F}_{\widehat{\texttt{mult}}_{\texttt{R}}} \leftarrow$ `Function_Evaluations`$(\widehat{\texttt{mult}}_{\texttt{R}}, \boldsymbol{T}_{\widehat{\texttt{mult}}_{\texttt{R}}}, s_1, \ldots, s_m)$.
– For $i \in [m]$, compute $g_{3m+i}(\texttt{Y}_i) = (g^L_{3m+i}(\texttt{Y}_i) + g^R_{3m+i}(\texttt{Y}_i)) \in R_A[\texttt{Y}_i]_{0,\leq 2}$ as follows. Notice that given $\mathcal{F}_{\hat{V}_R}, \mathcal{F}_{\widehat{\texttt{add}}_{\texttt{R}}}, \mathcal{F}_{\widehat{\texttt{mult}}_{\texttt{R}}}$, computation takes $O(2^{m-i})$ time:

$$g^L_{3m+i}(\texttt{Y}_i) = a_L \cdot \hat{V}_L(\vec{r}_{[2m+1,3m]}) + \sum_{\vec{y} \in H^{m-i}} \Big(a_L \cdot \hat{V}_R(\vec{s}_{[1,i-1]}, \texttt{Y}_i, \vec{y}) + m_L \cdot \hat{V}_L(\vec{r}_{[2m+1,3m]}) \cdot \hat{V}_R(\vec{s}_{[1,i-1]}, \texttt{Y}_i, \vec{y}) \Big).$$

$$g^R_{3m+i}(\texttt{Y}_i) = \beta \cdot \Big(\sum_{\vec{y} \in H^{m-i}} \big(\hat{V}_L(\vec{r}_{[2m+1,3m]}) + \hat{V}_R(\vec{s}_{[1,i-1]}, \texttt{Y}_i, \vec{y})\big) \cdot \widehat{\texttt{add}}_{\texttt{R}}(\vec{s}_{[1,i-1]}, \texttt{Y}_i, \vec{y}) + \hat{V}_L(\vec{r}_{[2m+1,3m]}) \cdot \hat{V}_R(\vec{s}_{[1,i-1]}, \texttt{Y}_i, \vec{y}) \cdot \widehat{\texttt{mult}}_{\texttt{R}}(\vec{s}_{[1,i-1]}, \texttt{Y}_i, \vec{y}) \Big).$$

– Return each polynomial $\{g_{3m+1}(\texttt{Y}_1), \ldots, g_{4m}(\texttt{Y}_m)\}$.

Fig. 13. Sumcheck polynomials for the block of $\vec{\texttt{Y}}$ variables

At the conclusion of the sumcheck protocol which is run to verify Equation (14), $\mathcal{V}$ encounters one of the usual obstacles in the GKR protocol. Namely, $\mathcal{V}$ needs to evaluate the following expression:

$$\widehat{\texttt{mult}}^{(1)}_{\texttt{L}}(\gamma, \vec{\texttt{X}}, \vec{\texttt{W}}) \cdot (\hat{V}^{(1)}_L(\vec{\texttt{X}}) \cdot \hat{V}^{(1)}_R(\vec{\texttt{Y}})) + \widehat{\texttt{add}}^{(1)}_{\texttt{L}}(\gamma, \vec{\texttt{X}}, \vec{\texttt{W}}) \cdot (\hat{V}^{(1)}_L(\vec{\texttt{X}}) + \hat{V}^{(1)}_R(\vec{\texttt{Y}}))$$

by replacing $\vec{\texttt{X}}, \vec{\texttt{Y}}, \vec{\texttt{W}}$ with respective random values $\chi^{(0)}_L, \psi^{(0)}_L, \omega^{(0)} \in A^{s_1}$. In our protocol, we assume that $\mathcal{V}$ has access to oracles that return the required evaluations[13] of $\widehat{\texttt{mult}}^{(i+1)}_{\texttt{L}}(\gamma, \vec{\texttt{X}}, \vec{\texttt{W}}), \widehat{\texttt{add}}^{i+1}_{\texttt{L}}(\gamma, \vec{\texttt{X}}, \vec{\texttt{W}})$ (for $i = 0, \ldots, D-1$). However, $\mathcal{V}$ cannot compute neither $\hat{V}^{(1)}_L(\chi^{(0)}_L)$ nor $\hat{V}^{(1)}_R(\psi^{(0)}_L)$ on their own, so $\mathcal{P}$ will provide those values. These new values claimed by the Prover have to satisfy the two following layer consistency equations:

$$\hat{V}^{(1)}_L(\chi^{(0)}_L) = \sum_{x,y \in \{0,1\}^{s_2}} \Big(\widehat{\texttt{add}}^{(2)}_{\texttt{L}}(\chi^{(0)}_L, x, y) \cdot (\hat{V}^{(2)}_L(x) + \hat{V}^{(2)}_R(y)) + \widehat{\texttt{mult}}^{(2)}_{\texttt{L}}(\chi^{(0)}_L, x, y) \cdot (\hat{V}^{(2)}_L(x) \cdot \hat{V}^{(2)}_R(y)) \Big). \tag{15}$$

$$\hat{V}^{(1)}_R(\psi^{(0)}_L) = \sum_{x,y \in \{0,1\}^{s_2}} \Big((\hat{V}^{(2)}_L(x) + \hat{V}^{(2)}_R(y)) \cdot \widehat{\texttt{add}}^{(2)}_{\texttt{R}}(\psi^{(0)}_L, x, y) + (\hat{V}^{(2)}_L(x) \cdot \hat{V}^{(2)}_R(y)) \cdot \widehat{\texttt{mult}}^{(2)}_{\texttt{R}}(\psi^{(0)}_L, x, y) \Big). \tag{16}$$

[13] For circuits with enough structure, the oracles can be either computed or verified by $\mathcal{V}$ in $\mathsf{poly}(s_i, s_{i+1})$ time, as shown in prior works.

Usually, at this point and in order to avoid an exponential blow-up in the depth of the circuit, a reduction from the two claimed values $\hat{V}_L^{(1)}(\chi_L^{(0)}), \hat{V}_R^{(1)}(\psi_L^{(0)})$ to a single one is performed, as we did in Section 5.2. In the alternative approach we present in this section, we proceed with the individual verification of Equations (15) and (16). As we are about to show, this will not lead up to computing $O(2^D)$ executions of the sumcheck protocol.

Let $\chi_L^{(1)}, \psi_L^{(1)}, \omega^{(1)} \in A^{s_2}$ (resp. $\chi_R^{(1)}, \psi_R^{(1)}, \upsilon^{(1)} \in A^{s_2}$) be the random challenges resulting from the execution of the sumcheck protocol for Eq. (15) (resp. Eq. (16)). Similar to what happened at the conclusion of the sumcheck protocol used to verify Eq. (14), $\mathcal{V}$ ends up needing to compute (besides the values they can obtain by querying the oracles), the values of $\hat{V}_L^{(2)}(\chi_L^{(1)}), \hat{V}_R^{(2)}(\psi_L^{(1)})$ and $\hat{V}_L^{(2)}(\chi_R^{(1)}), \hat{V}_R^{(2)}(\psi_R^{(1)})$. For $\mathcal{V}$ to meet their efficiency requirements, they need to resort to $\mathcal{P}$, but this time the four values can be reduced to two as described in Equations (17) and (18) instantiated for $i = 2$. Values $\alpha_L^{(i)}, \alpha_R^{(i)}, \beta_L^{(i)}, \beta_R^{(i)} \in A$ are sampled by $\mathcal{V}$ and sent to $\mathcal{P}$.

$$\begin{aligned}\alpha_L^{(i)} \cdot \hat{V}_L^{(i)}(\chi_L^{(i-1)}) + \beta_L^{(i)} \cdot \hat{V}_L^{(i)}(\chi_R^{(i-1)}) = \sum_{x,y \in \{0,1\}^{s_{i+1}}} \\ \Big(\quad \alpha_L^{(i)} \cdot \widehat{\mathtt{mult_L}}^{(i+1)}(\chi_L^{(i-1)}, x, y) \cdot \big(\hat{V}_L^{(i+1)}(x) \cdot \hat{V}_R^{(i+1)}(y)\big) + \\ + \beta_L^{(i)} \cdot \widehat{\mathtt{mult_L}}^{(i+1)}(\chi_R^{(i-1)}, x, y) \cdot \big(\hat{V}_L^{(i+1)}(x) \cdot \hat{V}_R^{(i+1)}(y)\big) + \\ + \alpha_L^{(i)} \cdot \widehat{\mathtt{add_L}}^{(i+1)}(\chi_L^{(i-1)}, x, y) \cdot \big(\hat{V}_L^{(i+1)}(x) + \hat{V}_R^{(i+1)}(y)\big) + \\ + \beta_L^{(i)} \cdot \widehat{\mathtt{add_L}}^{(i+1)}(\chi_R^{(i-1)}, x, y) \cdot \big(\hat{V}_L^{(i+1)}(x) + \hat{V}_R^{(i+1)}(y)\big) \quad \Big).\end{aligned} \tag{17}$$

$$\begin{aligned}\alpha_R^{(i)} \cdot \hat{V}_R^{(i)}(\psi_L^{(i-1)}) + \beta_R^{(i)} \cdot \hat{V}_R^{(i)}(\psi_R^{(i-1)}) = \sum_{x,y \in \{0,1\}^{s_{i+1}}} \\ \Big(\quad \alpha_R^{(i)} \cdot \big(\hat{V}_L^{(i+1)}(x) \cdot \hat{V}_R^{(i+1)}(y)\big) \cdot \widehat{\mathtt{mult_R}}^{(i+1)}(\psi_L^{(i-1)}, x, y) + \\ + \beta_R^{(i)} \cdot \big(\hat{V}_L^{(i+1)}(x) \cdot \hat{V}_R^{(i+1)}(y)\big) \cdot \widehat{\mathtt{mult_R}}^{(i+1)}(\psi_R^{(i-1)}, x, y) + \\ + \alpha_R^{(i)} \cdot \big(\hat{V}_L^{(i+1)}(x) + \hat{V}_R^{(i+1)}(y)\big) \cdot \widehat{\mathtt{add_R}}^{(i+1)}(\psi_L^{(i-1)}, x, y) + \\ + \beta_R^{(i)} \cdot \big(\hat{V}_L^{(i+1)}(x) + \hat{V}_R^{(i+1)}(y)\big) \cdot \widehat{\mathtt{add_R}}^{(i+1)}(\psi_R^{(i-1)}, x, y) \quad \Big).\end{aligned} \tag{18}$$

The advantage of this approach is that, as Eq. (17) (resp. Eq. (18)) only mixes polynomials in $R_A[\vec{\mathtt{X}}, \vec{\mathtt{W}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 1}$ (resp. $R_A[\vec{\mathtt{X}}, \vec{\mathtt{U}}, \vec{\mathtt{Y}}]_{\leq 1, \leq 2}$) with each other, the sumcheck protocol will now check $3m$-variate polynomials, rather than $4m$-variate polynomials as in Section 5. This directly translates into the claimed reduction on round complexity.

B.2 Sum-check protocol for Equations (6), (15) and (17)

For $j \in \{1, 2\}$, let $\vec{t_j} = (t_{j,1}, \ldots, t_{j,m})$. We provide a sum-check protocol for $\sum_{\vec{t_1}, \vec{t_2} \in H^m} f(\vec{t_1}, \vec{t_2}, \vec{t_2}) = \beta$, where $f \in R_A[\vec{\mathtt{X}}, \vec{\mathtt{W}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 1}$ is the multi-variate sandwich polynomial given by Equation (6).

If any of the checks throughout the protocol are not satisfied, $\mathcal{V}$ rejects.

Doubly-efficient IP over a non-commutative ring, using 4-to-2 reduction

Let R be a ring and A a commutative regular difference set such that $\{0,1\} \subset A \subset R$. Let $C : R^n \to R^k$ be a layered arithmetic circuit over R with depth D. W.l.o.g. we assume that n and k are powers of 2.
Input: Circuit input `inp` and claimed output `out`.
Output: Accept or reject.

- Compute $\hat{V}_L^{(0)}(\texttt{X})$ as the LMLE of `out`. $\mathcal{V}$ chooses a random $\gamma \in A^{s_0}$ and sends it to $\mathcal{P}$. Both parties compute $\hat{V}_L^{(0)}(\gamma)$.
- Run a sumcheck protocol on Equation (6) as described in Appendix B.2. Let $\chi_L^{(0)}, \psi_L^{(0)}, \omega^{(0)}$ denote the challenge vectors corresponding to the $\vec{\texttt{X}}$, $\vec{\texttt{Y}}$ and $\vec{\texttt{W}}$ variables within that execution. $\mathcal{P}$ sends $\hat{V}_L^{(1)}(\chi_L^{(0)})$ and $\hat{V}_R^{(1)}(\psi_L^{(0)})$ to $\mathcal{V}$.
- $\mathcal{V}$ queries their oracles for $\widehat{\texttt{mult}_\texttt{L}}^{(1)}(\gamma, \chi_L^{(0)}, \omega^{(0)})$ and $\widehat{\texttt{add}_\texttt{L}}^{(1)}(\gamma, \chi_L^{(0)}, \omega^{(0)})$, so as to check that $\widehat{\texttt{add}_\texttt{L}}^{(1)}(\gamma, \chi_L^{(0)}, \omega^{(0)}) \cdot (\hat{V}_L^{(1)}(\chi_L^{(0)}) + \hat{V}_R^{(1)}(\psi_L^{(0)})) + \widehat{\texttt{mult}_\texttt{L}}^{(1)}(\gamma, \chi_L^{(0)}, \omega^{(0)}) \cdot (\hat{V}_L^{(1)}(\chi_L^{(0)}) \cdot \hat{V}_R^{(1)}(\psi_L^{(0)}))$ equals the last message of the sumcheck execution.
- Run a sumcheck protocol on Equation (15) (resp. Equation (16)) as described in Appendix B.2 (resp. Appendix B.3). Let $\chi_L^{(1)}, \psi_L^{(1)}, \omega^{(1)}$ (resp. $\chi_R^{(1)}, \psi_R^{(1)}, \upsilon^{(1)}$) denote the challenge vectors corresponding to the $\vec{\texttt{X}}$, $\vec{\texttt{Y}}$ and $\vec{\texttt{W}}$ (resp. $\vec{\texttt{X}}$, $\vec{\texttt{Y}}$ and $\vec{\texttt{U}}$) variables within that execution. $\mathcal{P}$ sends $\hat{V}_L^{(2)}(\chi_L^{(1)})$ and $\hat{V}_R^{(2)}(\psi_L^{(1)})$ (resp. $\hat{V}_L^{(2)}(\chi_R^{(1)})$ and $\hat{V}_R^{(2)}(\psi_R^{(1)})$) to $\mathcal{V}$.
- $\mathcal{V}$ queries their oracles for $\widehat{\texttt{mult}_\texttt{L}}^{(2)}(\chi_L^{(0)}, \chi_L^{(1)}, \omega^{(1)})$ and $\widehat{\texttt{add}_\texttt{L}}^{(2)}(\chi_L^{(0)}, \chi_L^{(1)}, \omega^{(1)})$, so as to check that $\widehat{\texttt{add}_\texttt{L}}^{(2)}(\chi_L^{(0)}, \chi_L^{(1)}, \omega^{(1)}) \cdot (\hat{V}_L^{(2)}(\chi_L^{(1)}) + \hat{V}_R^{(2)}(\psi_L^{(1)})) + \widehat{\texttt{mult}_\texttt{L}}^{(2)}(\chi_L^{(0)}, \chi_L^{(1)}, \omega^{(1)}) \cdot (\hat{V}_L^{(2)}(\chi_L^{(1)}) \cdot \hat{V}_R^{(2)}(\psi_L^{(1)}))$ equals the last message of the sumcheck protocol for Equation (15).
- $\mathcal{V}$ queries their oracles for $\widehat{\texttt{mult}_\texttt{R}}^{(2)}(\psi_L^{(0)}, \chi_R^{(1)}, \upsilon^{(1)})$ and $\widehat{\texttt{add}_\texttt{R}}^{(2)}(\psi_L^{(0)}, \chi_R^{(1)}, \upsilon^{(1)})$, so as to check that $(\hat{V}_L^{(2)}(\chi_R^{(1)}) + \hat{V}_R^{(2)}(\psi_R^{(1)})) \cdot \widehat{\texttt{add}_\texttt{R}}^{(2)}(\psi_L^{(0)}, \chi_R^{(1)}, \upsilon^{(1)}) + (\hat{V}_L^{(2)}(\chi_R^{(1)}) \cdot \hat{V}_R^{(2)}(\psi_R^{(1)})) \cdot \widehat{\texttt{mult}_\texttt{R}}^{(2)}(\psi_L^{(0)}, \chi_R^{(1)}, \upsilon^{(1)})$ equals the last message of the sumcheck protocol for Equation (16).
- For circuit layers $i = 2, \ldots, D-1$:
  - $\mathcal{V}$ samples $\alpha_L^{(i)}, \alpha_R^{(i)}, \beta_L^{(i)}, \beta_R^{(i)} \in A$ and sends them to $\mathcal{P}$. They run a sumcheck protocol on Equation (17) (resp. Equation (18)) as described in Appendix B.2 (resp. Appendix B.3). Let $\chi_L^{(i)}, \psi_L^{(i)} \in A^{s_{i+1}}$ (resp. $\chi_R^{(1)}, \psi_R^{(1)} \in A^{s_{i+1}}$) denote the challenge vectors corresponding to the $\vec{\texttt{X}}$, and $\vec{\texttt{Y}}$ variables within that execution. At the end of the protocol, $\mathcal{P}$ sends $\hat{V}_L^{(i+1)}(\chi_L^{(i)})$ and $\hat{V}_R^{(i+1)}(\psi_L^{(i)})$ (resp. $\hat{V}_L^{(i+1)}(\chi_R^{(i)})$ and $\hat{V}_R^{(i+1)}(\psi_R^{(i)})$) to $\mathcal{V}$, so that $\mathcal{V}$ can check the validity of the last message in the sumcheck execution. If the check passes for both Equation (17) and Equation (18), they proceed to the $(i+1)$-th layer, otherwise, $\mathcal{V}$ outputs reject and aborts.
- At the input layer D, $\mathcal{V}$ has received four claims: $\hat{V}_L^{(D)}(\chi_L^{(D-1)})$, $\hat{V}_L^{(D)}(\chi_R^{(D-1)})$, $\hat{V}_R^{(D)}(\psi_L^{(D-1)})$ and $\hat{V}_R^{(D)}(\psi_R^{(D-1)})$. $\mathcal{V}$ queries the evaluation oracle of $\hat{V}_L^{(D)}$ (resp. $\hat{V}_R^{(D)}$) at $\chi_L^{(D-1)}$ and $\chi_R^{(D-1)}$ (resp. $\psi_L^{(D-1)}$ and $\psi_R^{(D-1)}$) and checks that they equal the sum-check claims. If they do, $\mathcal{V}$ outputs accept, otherwise, $\mathcal{V}$ outputs reject.

Fig. 14. Doubly-efficient IP over a non-commutative ring, using 4-to-2 reduction.

1. In the first round, $\mathcal{P}$ computes the univariate toast polynomial $g_1 \in R_A[\mathtt{X}_1]_{\leq 2,0}$ given by:

$$g_1(\mathtt{X}_1) = \sum_{\substack{t_{1,2},\ldots,t_{1,m} \in H \\ \vec{t_2} \in H^m}} f(\mathtt{X}_1, t_{1,2}, \ldots, t_{1,m}, \vec{t_2}, \vec{t_2}),$$

and sends $g_1(\mathtt{X}_1)$ to $\mathcal{V}$. Then $\mathcal{V}$ checks whether $g_1 \in R_A[\mathtt{X}_1]_{\leq 2,0}$ and $\sum_{b \in H} g_1(b) = \beta$. If that is the case, $\mathcal{V}$ chooses a random element $r_1 \in A$ and sends it to $\mathcal{P}$.
2. For rounds $2 \leq i \leq m$, $\mathcal{P}$ sends the univariate toast polynomial $g_i \in R_A[\mathtt{X}_i]_{\leq 2,0}$ given by:

$$g_i(\mathtt{X}_i) = \sum_{\substack{t_{1,i+1},\ldots,t_{1,m} \in H \\ \vec{t_2} \in H^m}} f(r_1, \ldots, r_{i-1}, \mathtt{X}_i, t_{1,i+1}, \ldots, t_{1,m}, \vec{t_2}, \vec{t_2}),$$

$\mathcal{V}$ checks whether $g_i \in R_A[\mathtt{X}_i]_{\leq 2,0}$ and $\sum_{b \in H} g_i(b) = g_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
3. In the m+1-th round, $\mathcal{P}$ computes the following two univariate toast polynomials $g_{m+1,0}, g_{m+1,1} \in R_A[\mathtt{W}_1]_{\leq 1,0}$:

$$g_{m+1,b}(\mathtt{W}_1) = \sum_{t_{2,2},\ldots,t_{2,m} \in H} f(r_1, \ldots, r_m, \mathtt{W}_1, t_{2,2}, \ldots, t_{2,m}, b, t_{2,2}, \ldots, t_{2,m}),$$

and sends them to $\mathcal{V}$. Then $\mathcal{V}$ checks whether $g_{m+1,0}, g_{m+1,1} \in R_A[\mathtt{W}_1]_{\leq 1,0}$ and $\sum_{b \in H} g_{m+1,b}(b) = g_m(r_m)$. If that is the case, $\mathcal{V}$ chooses a random element $r_{m+1} \in A$ and sends it to $\mathcal{P}$.
4. For rounds $m+2 \leq i \leq 2m$, $\mathcal{P}$, define $\vec{r}_{(i)} = (r_1, \ldots, r_{i-1})$, $\vec{t}_{(i)} = (t_{2,i-m+1}, \ldots, t_{2,m})$. $\mathcal{P}$ sends the univariate toast polynomials $g_{i,0}, g_{i,1} \in R_A[\mathtt{W}_{i-m}]_{\leq 1,0}$ given by:

$$g_{i,b}(\mathtt{W}_{i-m}) = \sum_{\substack{t_{2,1},\ldots,t_{2,i-m-1} \in H, \\ \vec{t}_{(i)} \in H^{2m-i}}} f(\vec{r}_{(i)}, \mathtt{W}_{i-m}, \vec{t}_{(i)}, t_{2,1}, \ldots, t_{2,i-m-1}, b, \vec{t}_{(i)}),$$

$\mathcal{V}$ checks whether $g_{i,0}, g_{i,1} \in R_A[\mathtt{W}_{i-m}]_{\leq 1,0}$ and $\sum_{b \in H} g_{i,b}(b) - g_{i-1,b}(r_{i-1}) = 0$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
5. For round $i = 2m+1$, define $\vec{r}_{(2m+1)} = (r_1, \ldots, r_{2m})$ and $\vec{t}_{(2m+1)} = (t_{2,2}, \ldots, t_{2,m})$. $\mathcal{P}$ sends the toast polynomial $g_{2m+1} \in R_A[\mathtt{Y}_{i-2m}]_{0,\leq 1}$ given by:

$$g_{2m+1}(\mathtt{Y}_1) = \sum_{\vec{t}_{(2m+1)} \in H^{m-1}} f(\vec{r}_{(i)}, \mathtt{Y}_1, \vec{t}_{(2m+1)}),$$

$\mathcal{V}$ checks whether $g_{2m+1} \in R_A[\mathtt{Y}_1]_{0,\leq 1}$ and $\sum_{b \in H} g_{2m+1}(b) - g_{2m,b}(r_{2m}) = 0$. If so, $\mathcal{V}$ chooses a random element $r_{2m+1} \in A$ and sends it to $\mathcal{P}$.
6. For rounds $2m+2 \leq i \leq 3m$, define $\vec{r}_{(i)} = (r_1, \ldots, r_{i-1})$ and $\vec{t}_{(i)} = (t_{2,i-2m+1}, \ldots, t_{2,m})$. $\mathcal{P}$ sends the toast polynomial $g_i \in R_A[\mathtt{Y}_{i-2m}]_{0,\leq 1}$ given by:

$$g_i(\mathtt{Y}_{i-2m}) = \sum_{\vec{t}_{(i)} \in H^{3m-i}} f(\vec{r}_{(i)}, \mathtt{Y}_{i-2m}, \vec{t}_{(i)}),$$

$\mathcal{V}$ checks whether $g_i \in R_A[\mathtt{Y}_{i-2m}]_{0,\leq 1}$ and $\sum_{b \in H} g_i(b) = g_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.

7. After the $3m$-th round, $\mathcal{V}$ checks whether $g_{3m}(r_{3m}) = f(r_1, \ldots, r_{3m})$ by querying the oracle at the point $(r_1, \ldots, r_{3m})$.

The following theorem provides the communication complexity and soundness analysis of the sum-check protocol we just presented. A lower soundness error of $\leq 4m \cdot |A|^{-1}$ could be achieved by "splitting" the $\vec{\mathtt{X}}$ variables into two as we did for $\vec{\mathtt{Y}}$, at the cost of increasing the communication and round complexity. Otherwise, hopefully, the same error can be obtained by doing a tighter analysis in our proof.

Theorem 6. *Let A be a commutative R.D. set such that $\{0,1\} \subseteq A$. Let $f \in R_A[\vec{\mathtt{X}}, \vec{\mathtt{W}}, \vec{\mathtt{Y}}]_{\leq 2, \leq 1}$ be a sandwich multivariate polynomial as described in Equation (6). The sum-check protocol is a public coin interactive proof with soundness error $\leq 6m \cdot |A|^{-1}$. The communication complexity is $9m$ elements in R.*

Proof. Completeness and communication complexity follow from inspection of the protocol, and hence we will concentrate on the soundness claim.

Let $\tilde{\mathcal{P}}$ denote an arbitrary malicious prover, trying to convince the verifier of a false claim $\sum_{\vec{t}_1, \vec{t}_2 \in H^m} f(\vec{t}_1, \vec{t}_2, \vec{t}_2) = \tilde{\beta}$, where $\tilde{\beta} \neq \beta$. During each of the $3m$ rounds, $\tilde{\mathcal{P}}$ has to send toast polynomials, specifically $\tilde{g}_i \in R_A[\mathtt{X}_i]_{\leq 2, 0}$ for $i \in [m]$, $\tilde{g}_{i,0}, \tilde{g}_{i,1} \in R_A[\mathtt{W}_{i-m}]_{\leq 1, 0}$ for $i \in [m+1, 2m]$ and $\tilde{g}_i \in R_A[\mathtt{Y}_{i-2m}]_{0, \leq 1}$ for $i \in [2m+1, 3m]$. Notice that the honest polynomials are indeed toasts, since the random challenges $r_1, \ldots, r_{i-1}$ are in A and can hence be "pushed to the middle" with the rest of the coefficients of each monomial. The verifier can easily check whether the polynomials received from $\tilde{\mathcal{P}}$ are also toasts of the right degree.

Let V denote the event where $\tilde{\mathcal{P}}$ succeeds cheating $\mathcal{V}$. For $i \in [1, m] \cup [2m+1, 3m]$, let E_i denote the event that $\tilde{g}_i = g_i$ and for $i \in [m+1, 2m]$, let E_i denote the event that $\sum_{b \in H} \tilde{g}_{i,b} = \sum_{b \in H} g_{i,b}$. Notice that $\Pr[V | E_1] = 0$, since $\mathcal{V}$ checks whether $\tilde{\beta}$ is equal to $\sum_{b \in H} \tilde{g}_1(b) = \sum_{b \in H} g_1(b) = \beta$.

Following reverse induction, we will prove that for $i = 3m, \ldots, 1$, $\Pr[V] \leq (3m - i + 1) \cdot 2 \cdot |A|^{-1} + \Pr[V | E_i \wedge \ldots \wedge E_{3m}]$. Once we prove the case $i = 1$ we will be done, since then $\Pr[V] \leq 3m \cdot 2 \cdot |A|^{-1} + \Pr[V | E_1 \wedge \ldots \wedge E_{3m}] \leq 3m \cdot 2 \cdot |A|^{-1} + \Pr[V | E_1] \leq 6m \cdot |A|^{-1}$.

For $i = 3m$, we have that $\Pr[V] \leq \Pr[V | \overline{E_{3m}}] + \Pr[V | E_{3m}] \leq |A|^{-1} + \Pr[V | E_{3m}]$. The inequality $\Pr[V | \overline{E_{3m}}] \leq |A|^{-1}$ follows from Lemma 8 as we explain next. Define $G_{3m}(\mathtt{Y}_m) = \tilde{g}_{3m}(\mathtt{Y}_m) - g_{3m}(\mathtt{Y}_m) \in R_A[\mathtt{Y}_m]_{0, \leq 1}$. Since we are in the case $\overline{E_{3m}}$, then $\tilde{g}_{3m} \neq g_{3m}$ and $G_{3m}(\mathtt{Y}_m)$ is *not* the zero polynomial. As the Verifier checks whether $\tilde{g}_{3m}(r_{3m}) = f(r_1, \ldots, r_{3m})$ and we know that $g_{3m}(r_{3m}) = f(r_1, \ldots, r_{3m})$, passing the check implies that $G_{3m}(r_{3m}) = 0$.

As an intermediate step towards proving that the statment is true for the $(i-1)$-th case, let us show that for $i = 3m, \ldots, 2$, we have $\Pr[V | \overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] \leq 2 \cdot |A|^{-1}$. First, for $i = 3m, \ldots, 2m+2$, define the toast polynomial

$$G_{i-1}(\mathtt{Y}_{i-2m-1}) = \tilde{g}_{i-1}(\mathtt{Y}_{i-2m-1}) - g_{i-1}(\mathtt{Y}_{i-2m-1}) \in R_A[\mathtt{Y}_{i-2m-1}]_{0, \leq 1},$$

which is non-zero in the event $(V | \overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m})$. Since $\mathcal{P}$ successfully passes the check $\sum_{b \in H} g_i(b) = \tilde{g}_{i-1}(r_{i-1})$ and we know that $\sum_{b \in H} g_i(b) = g_{i-1}(r_{i-1})$, by applying Lemma 8 we conclude that $\Pr[V | \overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] \leq |A|^{-1} \leq 2 \cdot |A|^{-1}$.

For $i = 2m+1, \ldots, m+2$ we apply the same reasoning of the previous paragraph, this time for the toast polynomial

$$G_{i-1}(\mathtt{W}_{i-m-1}) = \sum_{b \in H} \tilde{g}_{i-1,b}(\mathtt{W}_{i-m-1}) - g_{i-1,b}(\mathtt{W}_{i-m-1}) \in R_A[\mathtt{W}_{i-m-1}]_{\leq 1, 0}.$$

Finally, for $i = m+1, \ldots, 2$, the relevant toast polynomial for the application of Lemma 8 is $G_{i-1}(\mathtt{X}_{i-1}) = \tilde{g}_{i-1}(\mathtt{X}_{i-1}) - g_{i-1}(\mathtt{X}_{i-1}) \in R_A[\mathtt{X}_{i-1}]_{\leq 2,0}$.

Assume the induction hypothesis is true for i. Using our recently proved fact that $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] \leq 2 \cdot |A|^{-1}$, we have the following:

$$\begin{aligned}\Pr[V] &\leq (3m-i+1)\cdot 2\cdot |A|^{-1} + \Pr[V|E_i \wedge \ldots \wedge E_{3m}]\\ &\leq (3m-i+1)\cdot 2\cdot |A|^{-1} + \Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] + \Pr[V|E_{i-1} \wedge E_i \wedge \ldots \wedge E_{3m}]\\ &\leq (3m-i+1)\cdot 2\cdot |A|^{-1} + 2\cdot |A|^{-1} + \Pr[V|E_{i-1} \wedge E_i \wedge \ldots \wedge E_{3m}]\end{aligned}$$

Hence, $\Pr[V] \leq (3m-i+2)\cdot 2\cdot |A|^{-1} + \Pr[V|E_{i-1} \wedge \ldots \wedge E_{3m}]$ and the statement is true for the $(i-1)$-th case. This finishes our reverse induction and concludes our proof by reaching the case $i = 1$. ■

B.3 Sum-check protocol for Equations (16) and (18)

For completeness, we provide the sum-check protocol missing in Section 5.3. In more details, for $j \in \{1,2\}$, let $\vec{t_j} = (t_{j,1}, \ldots, t_{j,m})$. We provide a sum-check protocol for $\sum_{\vec{t_1},\vec{t_2} \in H^m} f(\vec{t_1}, \vec{t_2}, \vec{t_1}) = \beta$, where $f \in R_A[\vec{\mathtt{X}}, \vec{\mathtt{Y}}, \vec{\mathtt{W}}]_{\leq 1, \leq 2}$ is the multi-variate sandwich polynomial corresponding to V_R and given by Equation (5).

If any of the checks throughout the protocol are not satisfied, $\mathcal{V}$ rejects.

1. In the first round, $\mathcal{P}$ computes the two univariate toast polynomial $g_{1,0}, g_{1,1} \in R_A[\mathtt{X}_1]_{\leq 1,0}$ given by:
$$g_{1,b}(\mathtt{X}_1) = \sum_{\substack{t_{1,2},\ldots,t_{1,m} \in H\\ \vec{t_2} \in H^m}} f(\mathtt{X}_1, t_{1,2}, \ldots, t_{1,m}, \vec{t_2}, b, t_{1,2}, \ldots, t_{1,m}),$$
and sends them to $\mathcal{V}$. Then $\mathcal{V}$ checks whether $g_{1,0}, g_{1,1} \in R_A[\mathtt{X}_1]_{\leq 1,0}$ and $\sum_{b\in H} g_{1,b}(b) = \beta$. If that is the case, $\mathcal{V}$ chooses a random element $r_1 \in A$ and sends it to $\mathcal{P}$.
2. For rounds $2 \leq i \leq m$, define $\vec{t_{(i)}} = (t_{1,i+1}, \ldots, t_{1,m})$. $\mathcal{P}$ sends the univariate toast polynomials $g_{i,0}, g_{i,1} \in R_A[\mathtt{X}_i]_{\leq 1,0}$ given by:
$$g_{i,b}(\mathtt{X}_i) = \sum_{\substack{t_{1,1},\ldots,t_{1,i-1} \in H\\ \vec{t_{(i)}} \in H^{m-i}, \vec{t_2} \in H^m}} f(r_1, \ldots, r_{i-1}, \mathtt{X}_i, \vec{t_{(i)}}, \vec{t_2}, t_{1,1}, \ldots, t_{1,i-1}, b, \vec{t_{(i)}}),$$
$\mathcal{V}$ checks whether $g_{i,0}, g_{i,1} \in R_A[\mathtt{X}_i]_{\leq 1,0}$ and $\sum_{b\in H} g_{i,b}(b) - g_{i-1,b}(r_{i-1}) = 0$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
3. In the $m+1$-th round, $\mathcal{P}$ computes the univariate toast polynomial $g_{m+1} \in R_A[\mathtt{Y}_1]_{0,\leq 2}$:
$$g_{m+1}(\mathtt{Y}_1) = \sum_{\substack{t_{2,2},\ldots,t_{2,m} \in H\\ \vec{t_1} \in H^m}} f(r_1, \ldots, r_m, \mathtt{Y}_1, t_{2,2}, \ldots, t_{2,m}, \vec{t_1}),$$
and sends them to $\mathcal{V}$. Then $\mathcal{V}$ checks whether $g_{m+1} \in R_A[\mathtt{Y}_1]_{0,\leq 2}$ and $\sum_{b\in H} g_{m+1}(b) - g_{m,b}(r_m) = 0$. If that is the case, $\mathcal{V}$ chooses a random element $r_{m+1} \in A$ and sends it to $\mathcal{P}$.

4. For rounds $m+2 \le i \le 2m$, $\mathcal{P}$, $\mathcal{P}$ sends the univariate toast polynomials $g_i \in R_A[\mathtt{Y}_{i-m}]_{0,\le 2}$ given by:

$$g_i(\mathtt{Y}_{i-m}) = \sum_{\substack{t_{2,i-m+1},\ldots,t_{2,m}\in H,\\ \vec{t_1}\in H^m}} f(r_1,\ldots,r_{i-1},\mathtt{Y}_{i-m},t_{2,i-m+1},\ldots,t_{2,m},\vec{t_1}),$$

   $\mathcal{V}$ checks whether $g_i \in R_A[\mathtt{Y}_{i-m}]_{0,\le 2}$ and $\sum_{b\in H} g_i(b) = g_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
5. For rounds $2m+1 \le i \le 3m$, define $\vec{r}_{(i)} = (r_1,\ldots,r_{i-1})$ and $\vec{t}_{(i)} = (t_{2,i-2m+1},\ldots,t_{2,m})$. $\mathcal{P}$ sends the toast polynomial $g_i \in R_A[\mathtt{W}_{i-2m}]_{0,\le 1}$ given by:

$$g_i(\mathtt{W}_{i-2m}) = \sum_{t_{1,i-2m+1},\ldots,t_{1,m}\in H} f(r_1,\ldots,r_{i-1},\mathtt{W}_{i-2m},t_{1,i-2m+1},\ldots,t_{1,m}),$$

   $\mathcal{V}$ checks whether $g_i \in R_A[\mathtt{W}_{i-2m}]_{0,\le 1}$ and $\sum_{b\in H} g_i(b) = g_{i-1}(r_{i-1})$. If that is the case, $\mathcal{V}$ chooses a random element $r_i \in A$ and sends it to $\mathcal{P}$.
6. After the $3m$-th round, $\mathcal{V}$ checks whether $g_{3m}(r_{3m}) = f(r_1,\ldots,r_{3m})$ by querying the oracle at the point $(r_1,\ldots,r_{3m})$.

The following theorem provides the communication complexity and soundness analysis of the sum-check protocol we just presented. A lower soundness error of $\le 4m\cdot|A|^{-1}$ could be achieved by "splitting" the $\vec{\mathtt{Y}}$ variables into two as we did for $\vec{\mathtt{X}}$, at the cost of increasing the communication and round complexity. Otherwise, hopefully, the same error can be obtained by doing a tighter analysis in our proof.

Theorem 7. *Let A be a commutative R.D. set such that $\{0,1\}\subseteq A$. Let $f \in R_A[\vec{\mathtt{X}},\vec{\mathtt{Y}},\vec{\mathtt{W}}]_{\le 1,\le 2}$ be a sandwich multivariate polynomial as described in Equation (5). The sum-check protocol is a public coin interactive proof with soundness error $\le 6m\cdot|A|^{-1}$. The communication complexity is $9m$ elements in R.*

Proof. Completeness and communication complexity follow from inspection of the protocol, and hence we will concentrate on the soundness claim.

Let $\tilde{\mathcal{P}}$ denote an arbitrary malicious prover, trying to convince the verifier of a false claim $\sum_{\vec{t_1},\vec{t_2}\in H^m} f(\vec{t_1},\vec{t_2},\vec{t_1}) = \tilde{\beta}$, where $\tilde{\beta}\ne\beta$. During each of the $3m$ rounds, $\tilde{\mathcal{P}}$ has to send toast polynomials, specifically $\tilde{g}_{i,0},\tilde{g}_{i,1} \in R_A[\mathtt{X}_i]_{\le 1,0}$ for $i\in[m]$, $\tilde{g}_i \in R_A[\mathtt{Y}_{i-m}]_{0,\le 2}$ for $i\in[m+1,2m]$ and $\tilde{g}_i \in R_A[\mathtt{W}_{i-2m}]_{0,\le 1}$ for $i\in[2m+1,3m]$. Notice that the honest polynomials are indeed toasts, since the random challenges $r_1,\ldots,r_{i-1}$ are in A and can hence be "pushed to the middle" with the rest of the coefficients of each monomial. The verifier can easily check whether the polynomials received from $\tilde{\mathcal{P}}$ are also toasts of the right degree.

Let V denote the event where $\tilde{\mathcal{P}}$ succeeds cheating $\mathcal{V}$. For $i\in[1,m]$, let E_i denote the event that $\sum_{b\in H}\tilde{g}_{i,b} = \sum_{b\in H} g_{i,b}$ and for $i\in[m+1,3m]$, let E_i denote the event that $\tilde{g}_i = g_i$. Notice that $\Pr[V|E_1]=0$, since $\mathcal{V}$ checks whether $\tilde{\beta}$ is equal to $\sum_{b\in H}\tilde{g}_{1,b}(b) = \sum_{b\in H} g_{1,b}(b) = \beta$.

Following reverse induction, we will prove that for $i=3m,\ldots,1$, $\Pr[V] \le (3m-i+1)\cdot 2\cdot|A|^{-1} + \Pr[V|E_i\wedge\ldots\wedge E_{3m}]$. Once we prove the case $i=1$ we will be done, since then $\Pr[V]\le 3m\cdot 2\cdot|A|^{-1} + \Pr[V|E_1\wedge\ldots\wedge E_{3m}] \le 3m\cdot 2\cdot|A|^{-1} + \Pr[V|E_1] \le 6m\cdot|A|^{-1}$.

For $i=3m$, we have that $\Pr[V]\le\Pr[V|\overline{E_{3m}}]+\Pr[V|E_{3m}]\le|A|^{-1}+\Pr[V|E_{3m}]$. The inequality $\Pr[V|\overline{E_{3m}}]\le|A|^{-1}$ follows from Lemma 8 as we explain next. Define $G_{3m}(\mathtt{W}_m) = \tilde{g}_{3m}(\mathtt{W}_m) -$

$g_{3m}(\mathtt{W}_m) \in R_A[\mathtt{W}_m]_{0,\leq 1}$. Since we are in the case $\overline{E_{3m}}$, then $\tilde{g}_{3m} \neq g_{3m}$ and $G_{3m}(\mathtt{W}_m)$ is *not* the zero polynomial. As the Verifier checks whether $\tilde{g}_{3m}(r_{3m}) = f(r_1, \ldots, r_{3m})$ and we know that $g_{3m}(r_{3m}) = f(r_1, \ldots, r_{3m})$, passing the check implies that $G_{3m}(r_{3m}) = 0$.

As an intermediate step towards proving that the statment is true for the $(i-1)$-th case, let us show that for $i = 3m, \ldots, 2$, we have $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] \leq 2 \cdot |A|^{-1}$. First, for $i = 3m, \ldots, 2m+2$, define the toast polynomial

$$G_{i-1}(\mathtt{W}_{i-2m-1}) = \tilde{g}_{i-1}(\mathtt{W}_{i-2m-1}) - g_{i-1}(\mathtt{W}_{i-2m-1}) \in R_A[\mathtt{W}_{i-2m-1}]_{0,\leq 1},$$

which is non-zero in the event $(V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m})$. Since $\mathcal{P}$ successfully passes the check $\sum_{b\in H} g_i(b) = \tilde{g}_{i-1}(r_{i-1})$ and we know that $\sum_{b\in H} g_i(b) = g_{i-1}(r_{i-1})$, by applying Lemma 8 we conclude that $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] \leq |A|^{-1} \leq 2 \cdot |A|^{-1}$.

For $i = 2m+1, \ldots, m+2$ we apply the same reasoning of the previous paragraph, this time for the toast polynomial

$$G_{i-1}(\mathtt{Y}_{i-m-1}) = \tilde{g}_{i-1}(\mathtt{Y}_{i-m-1}) - g_{i-1}(\mathtt{Y}_{i-m-1}) \in R_A[\mathtt{Y}_{i-m-1}]_{0,\leq 2}.$$

Finally, for $i = m+1, \ldots, 2$, the relevant toast polynomial for the application of Lemma 8 is $G_{i-1}(\mathtt{X}_{i-1}) = \sum_{b\in H} \tilde{g}_{i-1,b}(\mathtt{X}_{i-1}) - g_{i-1,b}(\mathtt{X}_{i-1}) \in R_A[\mathtt{X}_{i-1}]_{\leq 1,0}$.

Assume the induction hypothesis is true for i. Using our recently proved fact that $\Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] \leq 2 \cdot |A|^{-1}$, we have the following:

$$\begin{aligned}\Pr[V] &\leq (3m-i+1) \cdot 2 \cdot |A|^{-1} + \Pr[V|E_i \wedge \ldots \wedge E_{3m}] \\ &\leq (3m-i+1) \cdot 2 \cdot |A|^{-1} + \Pr[V|\overline{E_{i-1}} \wedge E_i \wedge \ldots \wedge E_{3m}] + \Pr[V|E_{i-1} \wedge E_i \wedge \ldots \wedge E_{3m}] \\ &\leq (3m-i+1) \cdot 2 \cdot |A|^{-1} + 2 \cdot |A|^{-1} + \Pr[V|E_{i-1} \wedge E_i \wedge \ldots \wedge E_{3m}]\end{aligned}$$

Hence, $\Pr[V] \leq (3m-i+2) \cdot 2 \cdot |A|^{-1} + \Pr[V|E_{i-1} \wedge \ldots \wedge E_{3m}]$ and the statement is true for the $(i-1)$-th case. This finishes our reverse induction and concludes our proof by reaching the case $i = 1$. ■