Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 88 (2026)

A digest of the work of Rothblum, Vadhan, and Wigderson (2013)

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science, Rehovot, Israel.

June 18, 2026

Abstract

The work of Rothblum, Vadhan, and Wigderson (*STOC*, 2013) is pivotal to the study of interactive proofs of proximity (IPPs). We present the main contents of their work, while clarify a few (conceptual) aspects. Specifically, starting with the definition of IPP systems, our main focus is on the construction of IPP systems for any property in log-space uniform $\mathcal{NC}$ (and beyond). We also present limitations on the power of constant-round IPP systems.

Contents

1 Preface

The work of Rothblum, Vadhan, and Wigderson [15] is pivotal to the study of interactive proofs of proximity (IPPs). Specifically, their construction of IPP systems for any property that is decidable in log-space uniform $\mathcal{NC}$ (and beyond) is one of the two known general positive results about IPP systems; furthermore, the other result (by [13]) builds on the (two-stage) construction schema presented in [15, Sec. 3]. The focus of this memo (see Section 3) is on presenting this (two-stage) construction schema. In addition, in Section 4, we present the negative results of [15]; that is, the limitations of constant-round IPPs (as presented in [15, Sec. 4]).

ISSN 1433-8092

2 Defining IPPs

We assume familiarity with the basic notions of interactive proof systems (see, e.g., [5, Sec. 9.1]) and of property testers (see, e.g., [6, Chap. 1]).

The definition of interactive proofs of proximity is a hybrid of the definitions of interactive proofs and of property testers. One may start from the formulation of interactive proof systems and restrict the verifier's access to the common input (via the mechanism of oracle queries), while relaxing the soundness condition so that only inputs that are far from the property are required to be rejected. Alternatively, one may extend the framework of property testing by turning the tester into a verifier and letting it interact with an all-powerful prover. (We actually prefer the latter perspective, and comment that an NP (or rather MA) analogue of property testers, called MAPs, arises naturally in this case (see [9]).)

Recall that $(A,B)(x)$ denotes the output machine A when interacting with machine B on common input x, and that M^f denotes the behavior of machine M when given oracle access to a function f. For simplicity, we consider properties of Boolean functions; the formulation can be extended to functions with other ranges (potentially to ranges that depend on the domain). In addition, we associate the function's domain with $[n]$, for some $n \in \mathbb{N}$.

Definition 1 (verifier for property Π): *Let $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$ such that Π_n contains Boolean functions over $[n]$. An* interactive verifier of proximity for Π *is an interactive and probabilistic oracle machine, denoted V, that, on input parameters n and ϵ and oracle access to a function $f : [n] \to \{0,1\}$, interacts with a potential prover, and outputs a binary verdict that satisfies the following two conditions.*

1. Completeness: V accepts inputs in Π. *For every $n \in \mathbb{N}$ and $\epsilon > 0$, and for every $f \in \Pi_n$, there exists a prover strategy P that makes the verifier accept* (w.h.p)*; that is,*

$$\Pr[(P, V^f)(n,\epsilon)=1] \geq 2/3.$$

   If $\Pr[(P, V^f)(n,\epsilon)=1] = 1$ always holds, then we say that V has perfect completeness.

2. Soundness: V rejects inputs that are ϵ-far from Π. *For every $n \in \mathbb{N}$ and $\epsilon > 0$, and for every $f : [n] \to \{0,1\}$ such that $\delta_\Pi(f) > \epsilon$, no prover strategy $\widetilde{P}$ can make the verifier accept* (with significant probability)*; that is,*

$$\Pr[(\widetilde{P}, V^f)(n,\epsilon)=0] \geq 2/3,$$

   where $\delta(f,g) \stackrel{\text{def}}{=} |\{i \in [n] : f(i) \neq g(i)\}|/n$ and $\delta_\Pi(f) \stackrel{\text{def}}{=} \min_{g\in\Pi_n}\{\delta(f,g)\}$.

   *Indeed, we say that f is ϵ-*far from Π (resp., ϵ-far from g) *if $\delta_\Pi(f) > \epsilon$ (resp., $\delta(f,g) > \epsilon$).*

We shall often refer to such a verifier as an IPP system, where IPP stands for interactive proof of proximity.

An alternative formulation specifies the (honest) prover strategy that is used in the completeness condition (denoted P). This is called for when wishing to discuss the complexity of such honest prover strategies. Indeed, verifier of proximity that admit a relatively efficient proving strategy (in case $f \in \Pi$) are of natural interest. We stress that the honest prover is also provided with access to the input oracle (i.e., the function f), whereas the arbitrary prover (in the soundness condition) may depend (arbitrarily) on f anyhow.

Complexity measures. The query complexity of V is a function (of n and ϵ) that specifies the number of queries made by V on input parameters n and ϵ, when given oracle access to any function $f : [n] \to \{0,1\}^*$. (One may also define the query complexity of the honest prover; see [1].) Defining communication complexity measures presumes a convention by which the verifier expects a certain communication pattern (i.e., number of messages sent by the prover as well as their length). In that case, we define

- The round complexity of V is a function (of n and ϵ) that specifies the number of messages that V expects to receive (from a potential prover), when interacting on input parameters n and ϵ.[1]
- The communication complexity (of the prover) is a function (of n and ϵ) that specifies the total length of messages that V expects to receive (from a potential prover).

In addition, we may also consider the time complexity of V, which is a function (of n and ϵ) that specifies the number of steps taken by V when interacting with any potential prover. Moreover, one can also consider the time complexity of the honest prover, defined analogously.

Public-coin protocol. An interactive proof (or an IPP) system is said to be of the public-coin type if the verifier's messages are disjoint portions of its random-pad. In other words, at each round, the verifier makes fresh random choices (i.e., tosses a new sequence of coins) and sends their outcome (and nothing else) to the prover. In this case, the verifier is often called Arthur (denoted A), while the prover is called Merlin (denoted M).

(For a constant $r \in \mathbb{N}$, an r-round public-coin system is schematically said to have the form of either $[\mathrm{AM}]^r$ or $[\mathrm{AM}]^r\mathrm{A}$, where 'A' (resp., 'M') represent a verifier (resp., prover) message, and the last 'A' in $[\mathrm{AM}]^r\mathrm{A}$ represents a fictitious message that serves to allow the verifier to make additional random choices before taking its final decision.)[2]

3 IPPs for properties that are decidable in bounded-depth

Throughout most of this section, unless stated differently, by circuits we mean Boolean circuits with fan-in 2 (as used in the definition of the class $\mathcal{NC}$). The main result of [15] is the following

Theorem 2 (IPPs for $\mathcal{NC}$ (see [15, Thm. 1.1])):[3] *Let Π be a set* (equiv., a property) *that is decidable by log-space uniform circuits of depth D and size S, where $D(n) = \mathrm{poly}(\log n)$ and $S(n) = \mathrm{poly}(n)$ corresponds to $\mathcal{NC}$. Then, for every auxiliary parameter $\eta : \mathbb{N} \times [0,1] \to [0,1]$ such that $\eta(n,\epsilon) \leq \epsilon$, the property Π has a public-coin IPP of perfect completeness, with*

- *query complexity $q(n,\epsilon) = O(1/\eta(n,\epsilon))^{1+o(1)}$,*
- *round complexity $r(n) = O(D(n) \cdot \log S(n))$, and*

[1] We warn that the round complexity is sometimes defined (i.e., elsewhere) as the total number of messages exchanged (i.e., both prover and verifier messages).

[2] Using this convention, it is reasonable to refer to MA-systems as 1/2-round systems; these are actually randomized versions of NP-proof systems.

[3] For simplicity, we may assume that $D(n) = \Omega(\log S(n))$ and that $S(n) \geq n$. That is, we assume that the (fan-in 2) circuit is not degenerate.

- *communication complexity* $c(n,\epsilon) = O(\eta(n,\epsilon)^{1-o(1)} \cdot n) \cdot \mathrm{poly}(D(n))$.

Furthermore, the verifier runs in time $(q(n,\epsilon) + c(n,\epsilon))^{1+o(1)}$*, and the honest prover runs in time* $\mathrm{poly}(S)$.

Note that $q(n,\epsilon) \geq 1/\epsilon$ always holds. We stress that $\eta(n,\epsilon) \in (0,\epsilon]$ is a *free parameter* that offers a trade-off between the query complexity and the communication complexity. This trade-off satisfies

$$q(n,\epsilon) \cdot c(n,\epsilon) \leq (1/\eta(n,\epsilon)^{o(1)}) \cdot n \cdot \mathrm{poly}(D(n)) \leq n^{1+o(1)} \cdot \mathrm{poly}(D(n)). \quad (1)$$

(In [15, Thm. 1.1], $\eta(n,\epsilon)$ is set to ϵ, which creates a false impression as if some complexity measures decreases with ϵ, which is (of course) totally unintuitive.)

We comment that the subsequent work of Rothblum and Rothblum [14] replaces the $(1/\eta(n,\epsilon))^{o(1)}$ factors by $\mathrm{poly}(\log S)$ factors. These factors are viewed as the slackness of the result, because reasonable cryptographic assumptions imply that $q(n,\epsilon) \cdot c(n,\epsilon) = \Omega(n)$, for some set in $\mathcal{NC}$ (see [10]).

3.1 Proof of Theorem 2

Theorem 2 is proved by observing that the GKR interactive proof system [8] reduces the claim that its input (i.e., x) is in the set Π to a claim about the value of an encoding of its input x at a specific location, where the x is encoded by a low-degree multi-variate polynomial (over a finite field $\mathcal{F}$, which has $\mathrm{poly}(D(n))$ size). Specifically, the GKR-protocol refers to the encodings of the intermediate results of the computation of a predetermined circuit on the input x, and the claims are about values of these encodings at specified locations in these encodings. The protocol iteratively reduces a claim about the value (at one location) of an encoding of one layer of the circuit to a claim about the value (at one random location) of the encoding of an adjacent layer (closer to the circuit's input). Starting with a claim about the value at the circuit's output-layer (i.e., the value at the first location of the encoding of the output), we end with a claim regarding one location in the encoding of the circuit's input-layer (i.e., an encoding of x).[4]

Hence, for a multi-variate polynomial $P_x : \mathcal{F}^m \to \mathcal{F}$ of low degree, which is specified by the input x, given a location $\ell \in \mathcal{F}^m$ and a value $v \in \mathcal{F}$, we are left with the task of verifying that $P_x(\ell) = v$. In [8], this task is performed by a verifier that reads x and evaluates the polynomial P_x at ℓ by extrapolation, but, seeking a verifier of sublinear query (and time) complexity, we cannot afford reading x (entirely). Hence, we cannot hope to reject x whenever $P_x(\ell) \neq v$, because x may be very close to some x' such that $P_{x'}(\ell) = v$ holds, whereas it may be hard to distinguish x' and x. Fortunately, recalling that the IPP for Π is only required to reject x if x is ϵ-far from Π, we can relax the requirement regarding the residual verification (of $P_x(\ell) = v$) accordingly; that is, require

[4] Equating the contents of the circuit's input-layer with the input itself is inaccurate (but essentially correct). The contents of the circuit's input-layer is a string $X \in \{0,1\}^{S(n)}$ such that $X(i) = x(i)$ if $i \in [n]$ and $X(i) = 0$ otherwise. For any H such that $\{0,1\} \subseteq H \subset \mathcal{F}$, viewing x (resp., X) as a function $x : H^m \to \{0,1\}$ (resp., $X : H^{m'} \to \{0,1\}$), we note that the low-degree extension of X, denoted P_X, can be expressed in terms of the low-degree extension of x, denoted P_x. (In both cases we refer to multi-variate polynomials of individual degree $|H| - 1$.) Specifically, for every $(\ell, \ell') \in \mathcal{F}^m \times \mathcal{F}^{m'-m}$, it holds that $P_X(\ell, \ell') = Q(\ell') \cdot P_x(\ell)$, where Q is an $(m' - m)$-variate polynomial of individual degree $|H| - 1$ that satisfies $Q(0^{m'-m}) = 1$ and $Q(\ell') = 0$ for every $\ell' \in (H^{m'-m}) \setminus \{0\}^{m'-m})$. Hence, the last iteration of the GKR-protocol yields a claim of the form $P_X(\ell, \ell') = v'$, which is presented below as a claim of the form $P_x(\ell) = v$. This presentation is valid provided we set $v \leftarrow v'/Q(\ell')$, which presumes that $Q(\ell') \neq 0$ (as otherwise the verifier accepts if and only if $v' = 0$).

that x be rejected only if x is far from satisfying $P_x(\ell) = v$. We warn that this idea does not work as just stated, but a variant of it (detailed next) will do.

The point is that the GKR-protocol only guarantees that $x \in \Pi$ if and only if (w.h.p., over the execution that yields ℓ and v) it holds that $P_x(\ell) = v$. In order to claim that for every $x' \notin \Pi$ that is close to x it holds that $P_{x'}(\ell) \neq v$, we need a union bound over all the strings that are close to x, whereas the error probability in the GKR-protocol is $\Omega(1/|\mathcal{F}|)$. Nevertheless, we can essentially achieve our goal by using parallel executions of the GKR-protocol. Specifically, using t (parallel) repetitions, we obtain $(\ell_1, v_1), ..., (\ell_t, v_t) \in \mathcal{F}^m \times \mathcal{F}$ such that if $x' \notin \Pi$ then, with probability at least $1 - \exp(-t)$, it holds that $(P_{x'}(\ell_1), ..., P_{x'}(\ell_t)) \neq (v_1, ..., v_t)$. Now, for any η and $t = O(\eta n \cdot \log n)$ (so that $\exp(t) \gg \binom{n}{\eta n}$ holds), w.h.p., for every $x' \notin \Pi$ that is η-close to x it holds that $(P_{x'}(\ell_1), ..., P_{x'}(\ell_t)) \neq (v_1, ..., v_t)$.

Put in other words, for $\eta \leq \epsilon$, we reduced verifying $x \in \Pi$ with proximity parameter ϵ to verifying that $(P_x(\ell_1), ..., P_x(\ell_t)) = (v_1, ..., v_t)$ with proximity parameter η (i.e., we consider $P_{x'}$ for each x' that is η-close to x). Note that we can set $\eta \in (1/n, \epsilon]$ arbitrarily, and obtain a trade-off between $t = O(\eta n \cdot \log n)$ and the proximity parameter η for which we verify the equality $(P_x(\ell_1), ..., P_x(\ell_t)) = (v_1, ..., v_t)$. Intuitively, the query complexity of the resulting IPP will be linearly related to $1/\eta$, whereas its communication complexity will be linearly related to t (which is linearly related to η). Thus, while setting $\eta = \epsilon$ seems natural, it is not really the right thing to do, given that we seek flexibility in the trade-off between the query and communication complexities.[5]

Let us recap. On input $x \in \{0,1\}^n$, which we view as a function $x : [n] \to \{0,1\}$ (or rather as a function $x : H^m \to \{0,1\}$ for a suitable H), the first stage of the IPP for Π is an invocation of a variant of the GKR-protocol.[6] For $t = O(\eta n \cdot \log n)$, this invocation produces a sequence $(\ell_1, v_1), ..., (\ell_t, v_t) \in \mathcal{F}^m \times \mathcal{F}$, where $\mathcal{F}$ is a finite field of size $\text{poly}(D(n))$. The input x implicitly defines a polynomial $P_x : \mathcal{F}^m \to \mathcal{F}$, which is viewed as an encoding of x, such that the following conditions hold

1. If $x \in \Pi$, then $(P_x(\ell_1), ..., P_x(\ell_t)) = (v_1, ..., v_t)$ always holds.
2. If x is ϵ-far from Π, then, with high probability (over the execution of the first stage), for every x' that is $\min(\epsilon, \eta)$-far from x it holds that $(P_{x'}(\ell_1), ..., P_{x'}(\ell_t)) \neq (v_1, ..., v_t)$. That is, x is $\min(\epsilon, \eta)$-far from a property defined in Definition 2.1 (below).

The second stage of the IPP for Π constitutes an IPP for the foregoing property, where the latter IPP is invoked with the proximity parameter η.

Before spelling out the definition of this property, we comment that in [8], the standard setting of $|H| = O(\log n)$ is used; this setting guarantees that $|\mathcal{F}^m| = \text{poly}(|H^m|) = \text{poly}(n)$. However, as pointed out in [14], we may as well use $H = \{0,1\}$ (equiv., $m = \log_2 n$), although in that case $|\mathcal{F}^m|$ is quasi-polynomial in n. In both case, $P_x : \mathcal{F}^m \to \mathcal{F}$ will be an m-variate polynomial of individual

[5] Indeed, the flexibility can be maintained by redefining $\epsilon \leftarrow \eta$ in case we seek η that is smaller than the original ϵ. This is what [15] is actually doing.

[6] Using a recent terminology of [3], one may view the bulk of the GKR-protocol as an interactive protocol that solves a "dichotomous" search problem in the following sense:

- When given a valid instance, the *searcher* (akin a verifier) should output a corresponding valid solution (assuming that the *guide* (akin a prover) behaves properly).
- When given an invalid instance, no matter how the guide behaves, the searcher should either reject or output an invalid solution.

We stress that the searcher cannot necessarily determine by itself whether the output solution is valid or not.

degree $|H|-1$ that extends $x : H^m \to \{0,1\}$. We stress that the fact that $|\mathcal{F}^m|$ is not polynomial in n is not an obstacle, because the honest prover (in both stages) does not need to reconstruct the full description of P_x; it only needs to evaluate P_x in relatively few locations, which it can do by reading the entire x. (As for the verifier, it cannot afford to reconstruct the entire P_x in any case, because it cannot even afford to read the entire x (per being required to have sublinear query complexity).)

Definition 2.1 (Polynomial Evaluation, PVAL): *For a finite field $\mathcal{F}$ that contains $H \supseteq \{0,1\}$, a natural number m, and a function $x : H^m \to \{0,1\}$, we denote by $P_x : \mathcal{F}^m \to \mathcal{F}$ the unique m-variate polynomial of individual degree $|H|-1$ that extends x* (i.e., $P_x(\ell) = x(\ell)$ for every $\ell \in H^m$). *For every t, $\overline{\ell}$ and $\overline{v}$ such that $\overline{\ell} = (\ell_1, ..., \ell_t) \in (\mathcal{F}^m)^t$ and $\overline{v} = (v_1, ..., v_t) \in \mathcal{F}^t$, we define the following property*

$$\mathtt{PVAL}_{\overline{\ell},\overline{v}} \stackrel{\text{def}}{=} \{x \in H^m : (\forall i \in [t])\ P_x(\ell_i) = v_i\}\,. \tag{2}$$

Let $\mathtt{PVAL} = \bigcup_{t,m\in\mathbb{N},H\subset\mathcal{F}} \bigcup_{(\overline{\ell},\overline{v})\in(\mathcal{F}^m)^t\times\mathcal{F}^t} \mathtt{PVAL}_{\overline{\ell},\overline{v}}$.

Indeed, H and $\mathcal{F}$ are implicit in the notation $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$, and they were omitted for sake of simplicity. We can now state formally the effect of the variant of the GKR-protocol that is being used.

Lemma 2.2 (the first stage – reduction to PVAL):[7] *For every log-space uniform family of circuits of depth $D : \mathbb{N} \to \mathbb{N}$ and size $S : \mathbb{N} \to \mathbb{N}$ that decide Π, there exists an $O(D \log S)$-round prover-verifier protocol that satisfies the following conditions. On common input parameters $n \in \mathbb{N}$, $\epsilon, \eta \in [0,1]$, and H, m such that $|H|^m = n$ and $|H| \in [2, \mathrm{poly}(\log n)]$, the parties set $t = O(\min(\eta, \epsilon) \cdot n \cdot \log n)$ and fix a finite field $\mathcal{F} \supset H$ of size $\mathrm{poly}(D(n))$. In addition, the prover gets $x : H^m \to \{0,1\}$ as an auxiliary input. The verifier runs in time $O(t \cdot \mathrm{poly}(D(n)))$, the honest prover runs in time $\mathrm{poly}(S(n))$, and the following two conditions* (akin completeness and soundness) *hold:*

1. ("Completeness"): *If $x \in \Pi$ and the prover behave properly, then both parties always output some $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that $x \in \mathtt{PVAL}_{\overline{\ell},\overline{v}}$.*
2. ("Soundness"): *If x is ϵ-far from Π, then, no matter how the prover behaves, with probability at least 0.9, either the verifier rejects or it outputs $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that x is $\min(\epsilon, \eta)$-far from $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$.*

Furthermore, the system uses $O(D(n) \cdot \log S(n))$ rounds and is of the public-coin type.

Needless to say, the communication complexity is $O(t \cdot \mathrm{poly}(D))$. We stress that the verifier has no access to x; it only gets its length (i.e., n).

Proof Sketch: As hinted above, the case of $\epsilon = \eta = 0$ and $t = 1$ is implicit in [8]: This is effectively, the bulk of the GKR-interactive-proof system, which is followed (in [8]) by having the verifier evaluate $P_x : \mathcal{F}^m \to \mathcal{F}$ (by itself) on a single location (and compare the result to a single value).[8] The general case is proved as outlined above.

[7] Recall that using a recent terminology of [3], one may phrase this lemma as asserting the existence of an IPP-like protocol that solves a "dichotomous" search problem associated with PVAL (cf. Footnote 6). Specifically, the set of valid solutions for $x \in \Pi$ consists of pairs $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that $x \in \mathtt{PVAL}_{\overline{\ell},\overline{v}}$, whereas the set of invalid solutions for x that is ϵ-far from Π consists of pairs $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that x is $\min(\epsilon, \eta)$-far from $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$.

[8] Recall that being 0-far from a set means not being in it. Still, for sake of formality (wrt the definition of t), one may prefer setting $\epsilon = \eta = 1/2n$. In the current setting, we cannot afford to have the verifier evaluate P_x by itself, because this requires reading the entire input x. (We also remind the reader that we blurred the distinction between the contents of the input-layer of the circuit and the input itself (see Footnote 4).)

Specifically, if $x \in \Pi$, then the proper (parallel) execution of t copies of the GKR-protocol yields $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that $x \in \texttt{PVAL}_{\overline{\ell},\overline{v}}$. On the other hand, if x is ϵ-far from Π, then we consider the set of n-bit strings that are $\min(\epsilon, \eta)$-close to x, noting that all these strings are not in Π. Now, for each such x', with probability at least $1 - 2^{-t}$, the t parallel executions of the GRK-protocol (either rejects or) yields $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that $x' \notin \texttt{PVAL}_{\overline{\ell},\overline{v}}$. Using a union bound (on these $\binom{n}{\eta n} < 2^t/10$ strings), with probability at least 0.9, the verifier (either rejects or) outputs $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ such that x is $\min(\epsilon, \eta)$-far from $\texttt{PVAL}_{\overline{\ell},\overline{v}}$. ■

The second stage – an IPP for PVAL. Note that PVAL is actually a massively parameterized property (see, e.g., [6, Sec. 12.7.2]). Hence, in an IPP for PVAL, both parties get the explicit parameter as input, whereas the verifier gets oracle access to the actual input $x \in \{0,1\}^n$. Recall that the (problem) parameters for $\texttt{PVAL}_{\overline{\ell},\overline{v}}$ are $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$ as well as H and $\mathcal{F}$ (which are implicit in the notation), whereas $x \in \{0,1\}^n$ is viewed as $x : H^m \to \{0,1\}$. While we shall always invoke the following IPP with $t = O(\eta n \log n)$, its analysis does not depend on this setting (except for assuming that $m < |\mathcal{F}|/10$).

Lemma 2.3 (an IPP for $\texttt{PVAL}_{\overline{\ell},\overline{v}}$): *There exists an IPP for* PVAL *such that on input the problem parameters $(\overline{\ell}, \overline{v}) \in (\mathcal{F}^m)^t \times \mathcal{F}^t$, a proximity parameter η, and a trade-off parameter $r \in [m]$, for $\eta' = \min(\eta, |H|^{-r})$ and $n = |H|^m$, its complexities are as follows:*

- *query complexity $q(|H|, \eta', r) = T/\eta'$, where $T = O(\log |H|)^{2r}$;*
- *communication complexity $c(|H|, m, t, r) = O(|H|^{m-r} + r \cdot t \cdot |H|) \cdot T^{1/2} \cdot \log_2 |\mathcal{F}|$;*
- *round complexity $r + 1$;*
- *verifier's time complexity $(q(|H|, \eta', r) + c(|H|, m, t, r)) \cdot \mathrm{poly}(\log n)$;*
- *prover's time complexity $\mathrm{poly}(n)$.*

Furthermore, the IPP has perfect completeness and uses public coins.

In case $\eta' = |H|^{-r}$, we get query complexity $\widetilde{O}(|H|)^r$, whereas the communication complexity is

$$\frac{n}{\widetilde{\Omega}(|H|)^r} + O(t \cdot |H|) \cdot O(\log |H|)^r. \tag{3}$$

Recall that in our application $t = O(\eta n \cdot \log n)$, and that it always holds that $|H| < |\mathcal{F}| = \mathrm{poly}(\log n)$. Hence, when invoking Lemma 2.2, we better select η such that $n/|H|^r \approx \eta n \cdot |H|$ (equiv., $\eta \approx |H|^{-r-1}|$). In this case, we get query complexity $\widetilde{O}(|H|)^r$ and communication complexity $n/\widetilde{\Omega}(|H|)^r$.

A few additional comments on the choice of parameters are in place. First, for very small H (e.g., $|H| = 2$), the foregoing IPP has at least linear communication complexity (see Eq. (3)), which gives no improvement over the trivial IPP (in which the prover sends x to the verifier, who checks its authenticity using $O(1/\eta)$ queries). Indeed, recall that [15] uses the setting $|H| = \log n$, whereas [14] uses $|H| = 2$ while reducing the slackness factor (i.e., T) from $O(\log |H|)^{2r}$ to $\mathrm{poly}(\log n)$.

Second, note that $r = m$ is worthless, since in that case the query complexity, which is larger than $1/\eta'$, exceeds $|H|^m = n$. (In fact, the slackness factor (i.e., $T = O(\log |H|)^{2r}$) deems any

$r > m - (m/\log^{1-o(1)}|H|)$ worthless.) On the other hand, the case of $r = \Theta(m)$ yields the most appealing results. In that case, the slackness factor is $n^{\Theta((\log\log|H|)/\log|H|)}$, which calls for picking a large H (as long as $|H| \leq \mathrm{poly}(\log n)$).

Proof Sketch: At this point it is good to notice that the mapping $x \mapsto P_x$ is linear; hence, $\mathtt{PVAL}^{\overline{\ell},\overline{v}}$ is an affine space. On the other hand, the foregoing mapping constitutes a tensor code. Indeed, this tensor code is linear (equiv., it is a tensor of a base code that is linear).

We start with a toy problem, which corresponds to the special case of $m = 2$ (equiv., $n = |H^2|$). The treatment of this toy problem presents many of the ideas that are used in the general case. Furthermore, handling this toy problem yields a non-trivial IPP for Π, in which all verifier's complexities are of the form $\widetilde{O}(n^{3/4})$.

The toy problem. The first idea is having the prover send (to the verifier) the restriction of $x : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$ to the columns that occur in the sequence of locations $\overline{\ell} = ((\ell_1^{\mathtt{row}}, \ell_1^{\mathtt{col}}), ..., (\ell_t^{\mathtt{row}}, \ell_t^{\mathtt{col}})) \in (\mathcal{F}^2)^t$; that is, the prover is asked to send x restricted to the columns $\overline{\ell}^{\mathtt{col}} = (\ell_1^{\mathtt{col}}, ..., \ell_t^{\mathtt{col}})$. In fact, it suffices to send the restriction of these columns to the rows in H; that is, for every $i \in [t]$ and $h \in H$, the honest prover sends $x(h, \ell_i^{\mathtt{col}})$. Hence, the communicated message has length $O(t \cdot |H| \cdot \log|\mathcal{F}|)$.

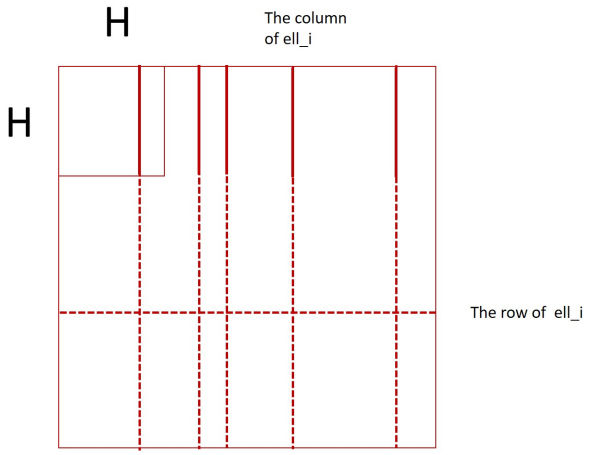

Figure 1: The prover's first message (i.e., x restricted to $H \times C$) and location $(\ell_i^{\mathtt{row}}, \ell_i^{\mathtt{col}})$.

Clearly, the verifier would reject if for any $i \in [t]$ the value of the i^{th} column (i.e., column $\ell_i^{\mathtt{col}}$) does not fit v_i (i.e., the univariate polynomial of degree $|H|-1$ that is supposed to describe $x(\cdot, \ell_i^{\mathtt{col}})$ does not equal v_i (at $\ell_i^{\mathtt{row}}$)). Specifically, for $C \stackrel{\text{def}}{=} \{\ell_i^{\mathtt{col}} : i \in [t]\}$, let $y : H \times C \to \{0,1\}$ denote the message sent by the prover (see Figure 1). Then, we may assume (w.l.o.g) that, for every $i \in [t]$, the degree $|H| - 1$ univariate polynomial that interpolates $(y(h, \ell_i^{\mathtt{col}}))_{h \in H}$ evaluates to v_i at $\ell_i^{\mathtt{row}}$.

Suppose that x is η-far from $\mathtt{PVAL}^{\overline{\ell},\overline{v}}$ and that the prover avoided rejection in the prior message. Obviously, y can be extended (by interpolation (which is unique if and only if $|C| \geq |H|$)) to $x' \in \mathtt{PVAL}^{\overline{\ell},\overline{v}}$, which means that (any such extension) x' must be η-far from x. Furthermore, for each $h \in H$, the h^{th} row of such x' is in $\Pi_h \stackrel{\text{def}}{=} \mathtt{PVAL}^{C,\{y(h,j):j\in C\}}$. Hence, letting δ_h denote the (relative) distance of the h^{th} row of x from Π_h, we have

$$\sum_{h \in H} \delta_h \cdot |H| > \eta \cdot |H|^2$$

because extending the corrected rows that satisfy the Π_h's yields a matrix in $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$ (whereas such a matrix must be η-far from x).[9] Thus, we got

$$\sum_{h\in H}\delta_h > \eta\cdot|H|. \tag{4}$$

By bucketing the rows in H according to their approximate δ_h's (up to a constant factor), we infer that there exists a set $H'\subseteq H$ such that $\delta_h=\Omega(\eta\cdot|H|/(|H'|\cdot\log|H|))$ for every $h\in H'$.[10] Now, the verifier selects uniformly at random $j\in[\log_2|H|]$, hoping that $|H'|\approx 2^j$, selects uniformly a set R of $O(|H|/2^j)$ rows in H (hoping that R contains some row in H'), and selects a random linear combination of these rows, denoted $(c_i)_{i\in R}\in\mathcal{F}^{|R|}$. The hope is that $z\stackrel{\text{def}}{=}\sum_{i\in R}c_i\cdot x(i,\cdot)\in\mathcal{F}^{|H|}$ is $\Omega(\eta\cdot|H|/(2^j\cdot\log|H|))$-far from fitting the corresponding linear combination that is applied to y (i.e., the values $w_j\stackrel{\text{def}}{=}\sum_{i\in R}c_i\cdot y(i,j)$ for $j\in C$).[11]

We now face a 1-dimensional PVAL problem (i.e., is z in $\mathtt{PVAL}_{C,(w_i)_{i\in C}}$ or is it $\Omega(\eta\cdot|H|/(2^j\cdot\log|H|))$-far from $\mathtt{PVAL}_{C,(w_i)_{i\in C}}$), which we solve by a straightforward IPP. Specifically, the verifier sends R and $(c_i)_{i\in R}$ to the prover, asking it to provide the foregoing $z\in\mathcal{F}^{|H|}$. Recall that if x is η-far from $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$ then, with probability $\Omega(1/\log|H|)$ over the choice of j (and the choices of R and $(c_i)_{i\in R}$), the foregoing z is $\Omega(\eta\cdot|H|/(2^j\cdot\log|H|))$-far from $\mathtt{PVAL}_{C,(w_i)_{i\in C}}$. Denoting the prover's answer by $\widetilde{z}:H\to\mathcal{F}$, the verifier rejects if $\widetilde{z}\notin\mathtt{PVAL}_{C,(w_i)_{i\in C}}$. Otherwise, the verifier selects $s=O((2^j\cdot\log|H|)/(\eta\cdot|H|))$ entries uniformly at random in $\widetilde{z}$ (i.e., locations in H), and compares their values (in $\widetilde{z}$) to the linear combinations of the corresponding entries of $\sum_{i\in R}c_i\cdot x(i,\cdot)$. That is, for each selected location $h\in H$, it compares $\widetilde{z}(h)$ to $\sum_{i\in R}c_i\cdot x(i,h)$, by making $|R|$ queries to x. Thus, the verifier makes $s\cdot|R|=O(\log|H|)/\eta$ queries to x, where the equality is due $|R|=O(|H|/2^j)$ and the definition of s (i.e., $s=O((2^j\cdot\log|H|)/(\eta\cdot|H|))$).

Let us recap. The foregoing verifier always accepts $x\in\mathtt{PVAL}_{\overline{\ell},\overline{v}}$, and rejects x that is η-far from $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$ with probability $\Omega(1/\log|H|)$, where this factor is mostly due to guessing the approximate

[9] Note that, for every $j\in C$, the j^{th} column of the corrected rows that satisfy the Π_h's (for $h\in H$) defines a univariate polynomial that is consistent with the v_i's that reside in column j.

[10] Note that if $\delta_h=\Omega(\eta\cdot|H|)$ for some $h\in H$, then the claim follows (with $H'=\{h\}$). Otherwise (i.e., $\delta_h<\eta\cdot|H|/2$ for every $h\in H$), for every $j\in[\log_2|H|]$, let $B_j=\{h\in H:\delta_h\in(\eta\cdot|H|/2^{j+1},\eta\cdot|H|/2^j]\}$. Then,

$$\begin{aligned}\eta\cdot|H| &< \sum_{j\in[\log_2|H|]}\sum_{h\in B_j}\delta_j \;+\; \sum_{h\in H\setminus\bigcup_j B_j}\delta_j\\ &\leq \sum_{j\in[\log_2|H|]}|B_j|\cdot\frac{\eta\cdot|H|}{2^j} \;+|H|\cdot\frac{\eta\cdot|H|}{2|H|}\end{aligned}$$

which implies that there exists j such that $|H_j|=\Omega(2^j/\log|H|)$. Letting $H'=B_j$, note that for every $h\in H'$ it holds that $\delta_h\geq\eta\cdot|H|/2^{j+1}=\Omega(\eta\cdot|H|/(|H'|\cdot\log|H|))$.

[11] Recall that $\Pi_h=\mathtt{PVAL}_{C,\{y(h,j):j\in C\}}$ is an affine space of the form $V+y(h,\cdot)$, where V is a linear space that depends on C only (i.e., V is independe of h). As proved in [15, Clm. 3.11 (full version)], if R contains a row h that is $\Omega(\eta\cdot|H|/(2^j\cdot\log|H|))$-far from Π_h, then (w.h.p.) z is $\Omega(\eta\cdot|H|/(2^j\cdot\log|H|))$-far from the 1-dimensional PVAL instance parameterized by $(C,(w_j)_{j\in C})$ (i.e., z is far from $V+w(\cdot)$). We warn that this claim is highly non-trivial and clearly more refined than the well-known fact that refers to preservation of non-membership (i.e., "if R contains a row h that is not in Π_h, then (w.h.p.) z is not in the 1-dimensional PVAL). Letting x_h denote the h^{th} row of x and recalling that it is at distance δ_h from the affine space Π_h, the standard argument (of fixing the other c_i's) reduces the foregoing claim to an analysis of the distance of $c_h\cdot x_h$ from the affine space $c_h\cdot\Pi_h+s$, for a uniformly distributed $c_h\in\mathcal{F}$, where $s\in\mathcal{F}^{|H|}$ is arbitrary. As proved in [14, Lem. 5.8] (following [15, Lem. 1.6 (full version)]), with high probability over the choice of $r\in\mathcal{F}$, the distance $r\cdot x_h$ from $r\cdot\Pi_h+s$ is at least $\delta_h/2$. The proof is reproduced in the appendix.

size of H'. As for the communication complexity, the prover's messages, denoted y and $\widetilde{z}$, are of length $O(t \cdot |H| \cdot \log |\mathcal{F}|)$ and $O(|H| \log |\mathcal{F}|)$, respectively. Recall that the query complexity is $O(\log |H|)/\eta$.

Recalling that $n = |H|^2$ and $t \approx \eta \cdot |H|^2$, consider the case that $\eta = |H|^{-3/2}$: In this case, both the query and communication complexities of the foregoing IPP are $\widetilde{O}(|H|^{3/2}) = \widetilde{O}(n^{3/4})$. This, *per se*, yields a non-trivial IPP for PVAL. (Furthermore, using some trivial adaptation, this yields a non-trivial IPP for any set in $\mathcal{NC}$.)[12]

Figure 2: The case of $m > 2$; compare to Figure 1.

Solving the real problem. We now turn to the general case (i.e., arbitrary $m \in \mathbb{N}$ and $r \in [m]$). Here we view the input x as a function $x : H \times H^{m-1} \to \{0,1\}$ and view P_x as $P_x : \mathcal{F} \times \mathcal{F}^{m-1} \to \mathcal{F}$ (see Figure 2). Likewise, we view PVAL's parameters (i.e., $\overline{\ell}$ and $\overline{v} = (v_1, ..., v_t) \in \mathcal{F}^t$) as a sequence of evaluation claims regarding positions in $\mathcal{F} \times \mathcal{F}^{m-1}$ (i.e., $\overline{\ell} = (\ell_1, ..., \ell_t) \in (\mathcal{F} \times \mathcal{F}^{m-1})^t$ claiming $P_x(\ell_i) = v_i$ for every $i \in [t]$). We can now apply the same strategy as in the toy problem (but here $\ell_i = (\ell_i^{\mathtt{row}}, \ell_i^{\mathtt{col}}) \in \mathcal{F} \times \mathcal{F}^{m-1}$), except that we don't use the straightforward IPP but rather make a recursive call with dimension $m - 1$. Specifically,

1. The honest prover sends the restriction of x to $H \times C$, where $C \stackrel{\text{def}}{=} \{\ell_i^{\mathtt{col}} : i \in [t]\} \subset \mathcal{F}^{m-1}$; that is, for every $i \in [t]$ and $h \in H$, the honest prover sends $x(h, \ell_i^{\mathtt{col}})$.

   The message actually sent by the prover is denoted $y : H \times C \to \mathcal{F}$. We may assume (w.l.o.g), that for every $i \in [t]$, the degree $|H| - 1$ univariate polynomial that interpolates the values $(y(h, \ell_i^{\mathtt{col}}))_{h \in H}$ (at H) evaluates to v_i at $\ell_i^{\mathtt{row}}$.

   (Suppose that x is η-far from $\mathtt{PVAL}_{\overline{\ell},\overline{v}}$. Then, for every $h \in H$, defining Π_h and δ_h as in the toy problem, Eq. (4) holds, since $\sum_{h \in H} \delta_h \cdot |H|^{m-1} > \eta \cdot |H|^m$. Again, there exists a set $H' \subseteq H$ such that $\delta_h = \Omega(\eta \cdot |H|/(|H'| \cdot \log |H|))$ for every $h \in H'$.)

2. The verifier selects uniformly at random $j \in [\log_2 |H|]$, hoping that $|H'| \approx 2^j$ (when x is far from PVAL), selects an $O(|H|/2^j)$-subset R of H (hoping that R contains some row in H'), and selects a random linear combination of these rows, denoted $(c_i)_{i \in R} \in \mathcal{F}^{|R|}$.

   The hope is that, when x is η-far from PVAL, the resulting random combination (of the rows of x), denoted $z \stackrel{\text{def}}{=} \sum_{i \in R} c_i \cdot x(i, \cdot) \in \mathcal{F}^{|H|^{m-1}}$, is $\Omega(\eta \cdot |H|/(2^j \cdot \log |H|))$-far from fitting the corresponding linear combination that is applied to y (i.e., the values $w_j \stackrel{\text{def}}{=} \sum_{i \in R} c_i \cdot y(i, j)$ for $j \in C$).)

[12] Specifically, recall that Lemma 2.2 is stated for $m = \log_{|H|} n$ and $|H| \leq |\mathcal{F}| = \text{poly}(\log n)$, whereas we wish to use $m = 2$. Hence, we first embed the output of the protocol guaranteed by Lemma 2.2 in a field $\mathcal{F}' = \mathcal{F}^{m/2}$ and use $H' = H^{m/2}$. Next, we view the input x as a function $x : H' \times H' \to \{0,1\}$, and view PVAL's parameters as a sequence of evaluation claims regarding positions in $\mathcal{F}' \times \mathcal{F}'$ (and values in $\mathcal{F}'$). Now, we can apply the foregoing IPP to the input x (and these PVAL parameters).

3. We now face an $(m-1)$-dimensional PVAL problem (i.e., is z in $\mathtt{PVAL}_{C,(w_j)_{j\in C}}$ or is it $\Omega(\eta \cdot |H|/(2^j \cdot \log|H|))$-far from $\mathtt{PVAL}_{C,(w_j)_{j\in C}}$). The plan is to (recursively) apply the very same procedure on this instance, but a couple of comments are in place before doing so:

   (a) The proximity parameter (typically) increases by a factor of $\Omega(|H|/(2^j \cdot \log|H|))$ (i.e., from η to $\Omega(\eta \cdot |H|/(2^j \cdot \log|H|))$). This will be useful when we reach the two-dimensional case, where we apply the straightforward IPP (which samples the input at a rate that is inversely proportional to the proximity parameter).

   (b) We no longer have direct access to $\widetilde{z} : H^{m-1} \to \mathcal{F}$, which is claimed to equal z (since $\widetilde{z}$ is not sent by the prover in this case). Still, as before, each query to z can be emulated by $|R| = |H|/2^j$ queries to x.

   Hence, the product of the size of the potential sample of z (which is inversely proportional to the proximity parameter) and the cost of answering a query to z increases by a factor of $O(\log|H|)$. We also recall that the probability that this "reduction step" maintains the predetermined distances is $\Omega(1/\log|H|)$.

We shall keep recursing $r-1$ times and invoke a protocol akin to the toy problem at the end. We stress that in these $r-1$ invocations of the recursive IPP no queries are made, and that in each of them the prover sends a single message of length $O(t \cdot |H| \cdot \log|\mathcal{F}|)$. For $i \in [r-1]$, the i^{th} recursive call defines a virtual input $x^{(i)} : H^{m-i} \to \mathcal{F}$, corresponding parameters for an $(m-i)$-dimensional PVAL problem, and a proximity parameter $\eta^{(i)}$ for that problem.

The last (i.e., r^{th}) invocation differs from the previous $r-1$ invocations in Step 3: At this point we face an $(m-r)$-dimensional PVAL problem, which we solve as in the toy problem. Specifically, the prover sends a message $\widetilde{z} : |H|^{m-r} \to \mathcal{F}$ that is supposed to equal the virtual input $x^{(r)}$, which is (iteratively) determined in the r invocations, the verifier rejects if this $\widetilde{z}$ does not reside in the corresponding PVAL problem, and otherwise tests that $\widetilde{z}$ equals the virtual input $x^{(r)}$. Each query made to the virtual input $x^{(r)}$ is emulated using queries to the actual input (i.e., x), whereas the number of emulated queries is inversely proportional to the (iteratively) determined proximity parameter (i.e., $\eta^{(r)}$). Recall that the product of the number of emulated queries and the cost of emulating each query (by queries to the original input) equals $O(\log|H|)^r/\eta'$, since in each recursive call the corresponding product is increased by a factor of $O(\log|H|)$.

The resulting system always accepts $x \in \mathtt{PVAL}$, and rejects each x that is η-far from PVAL with probability $\Omega(1/\log|H|)^r$. Employing error-reduction, and letting $\eta' = \min(\eta, |H|^{-r})$, we derive an IPP with the following complexities:

- query complexity T/η', where $T = O(\log|H|)^{2r}$;
- communication complexity $O(|H|^{m-r} + r \cdot t \cdot |H|) \cdot T^{1/2} \cdot \log|\mathcal{F}|$;
- round complexity $r+1$;
- verifier's time complexity $O(|H|^{m-r} + r \cdot t \cdot |H| + (1/\eta')) \cdot T \cdot \text{poly}(\log n)$;
- prover's time complexity $\text{poly}(n)$.

Needless to say, the foregoing description is quite hand-waving. Specifically, the proximity parameter cannot be increased indefinitely, and this explains why $\eta < |H|^{-r}$ (or so) must hold (equiv., η replaced by $\eta' = \min(\eta, |H|^{-r})$). ■

Digest: Dimension reduction. The IPP obtained in (the proof of) Lemma 2.3 proceeds in r rounds such that in the i^{th} round it reduces an $(m-i+1)$-dimensional PVAL problem to an $(m-i)$-dimensional PVAL problem. The input to the former problem, denoted $x^{(i-1)}$, where $x^{(0)}$ equals the original input x, is replaced by a virtual input, denoted $x^{(i)}$, such that each bit in $x^{(i)}$ can be recovered by making q_i queries to $x^{(i-1)}$. Fortunately, the proximity parameter for the latter problem, denoted $\eta^{(i)}$ (where $\eta^{(0)} = \eta'$), increases by a factor that is related to q_i. Unfortunately, the said factor is $\Omega(q_i/\log|H|)$, rather than q_i, and this is one source of the slackness factor. Another source of slackness is that the verifier does not know the q_i's but rather guesses their approximate value (and so in each round it is correct with probability $\Omega(1/\log|H|)$). In any case, the last round (i.e., round r) is different. At this point a straightforward IPP is used: The prover is supposed to send the current virtual input (i.e., $x^{(r)}$) to the verifier, who tests it by using $O(1/\eta^{(r)})$ queries, where each of these queries is emulated by making $\prod_{i\in[r]} q_i$ queries to the actual input (i.e., x). Recalling that

$$\eta^{(r)} = \left(\prod_{i\in[r]} \Omega(q_i/\log|H|)\right) \cdot \eta',$$

this translates to query complexity $O(\log|H|)^r/\eta'$. As for the communication complexity, we observe that at each of the first $r-1$ rounds the prover sent a message of length at most $t\cdot|H|\log_2|\mathcal{F}|$, whereas in the last round it also sends a message of length $|H|^{m-r}\cdot\log_2|\mathcal{F}|$.

Alternative digest: Batch verification. An alternative perspective is that each round ($i\in[r]$) reduces a single $(m-i+1)$-dimensional PVAL to $|H|$ instances of an $(m-i)$-dimensional PVAL, and then applies "batch verification" (a notion explicitly introduced in [13])[13] to these $|H|$ instances. The IPP obtained in (the proof of) Lemma 2.3 capitalizes on the fact that it is dealing with IPPs rather than with standard interactive proofs, and that is PVAL is a linear property. Furthermore, it uses the fact that the $|H|$ instances to be batched are at an average (relative) distances that equals the (relative) distance of the original instance (see Eq. (4)). At this point, we cluster the $|H|$ instances according to the approximate magnitude of their distances and pick a cluster H' that is responsible for an $\Omega(1/\log|H|)$ fraction of the sum of distances. We effectively perform batch verification on these $|H'|$ instances, and this is done by taking a random linear combination of these instances, which yields a single instance that is of distance that is lower-bounded by the distance of each of these $|H'|$ instances. Indeed, we lose one $O(\log|H|)$ factor (per round) due to focusing on the "heaviest" cluster, and an addition $O(\log|H|)$ factor (per round) due to guessing the identity of the cluster.

3.2 Comments

For simplicity, we described the two-stage protocol as referring to the encoding of the input x by a Reed-Muller codeword P_x. However, both stages can be carried out when using tensor codes with some extra features (i.e., linearity and multiplicativity). In particular, both extra features are used in the proof of Lemma 2.2 (cf. [12]), but the proof of Lemma 2.3 holds for any linear tensor code.

[13] We comment that the notion of batch verification is implicit in the sum-check protocol of [11]; an extensive study of this notion was initiated in [13], and [14] provides an almost optimal result (for the case of sets in $\mathcal{NC}$). Note that in all these cases batch verification was studied in the context of standard interactive proof systems, whereas here we discuss batch verification in the context of IPPs.

An application: For any finite field $\mathcal{F}$ and $d \in [\Omega(|\mathcal{F}|), |\mathcal{F}| - \Omega(|\mathcal{F}|)]$, consider the set, denoted $P_{\mathcal{F},d}$, of univariate polynomials of degree d over $\mathcal{F}$. Viewing $n = |\mathcal{F}|$ as varying, and considering a constant value of the proximity parameter, we note that testing $P_{\mathcal{F},d}$ has query complexity $d + 2 = \Omega(|\mathcal{F}|)$. Yet, for any $q(n) \geq n^{o(1)}$, Theorem 2 yields an IPP of query complexity $q(n)$ and communication complexity $n^{1+o(1)}/q(n)$ for $P_{\mathcal{F},d}$. This holds since the foregoing set is in $\mathcal{NC}$.

However, it is weird to work with a Boolean circuit that decides $P_{\mathcal{F},d}$, whereas there is a more natural Arithmetic circuit that decides this algebraic problem.[14] Indeed, Theorem 2 can be easily adapted to Arithmetic circuit; in fact, it suffices to adapt the GKR-protocol to this case.

A generalization to massively parametrized properties. While Definition 1 and Theorem 2 refers to ordinary properties, both extend easily to massively parameterized properties of the form $\Pi = \bigcup_{p \in \{0,1\}^*} \Pi_p$, where $\Pi_p \subseteq \{0,1\}^{|p|}$ (see, e.g., [6, Sec. 12.7.2]).[15] Indeed, Definition 1 can be seen as a special case (i.e., $\Pi_p = \Pi_{1^{|p|}}$). In the case of massively parameterized properties, the parameter p is given explicitly to both parties, whereas (as before) the verifier has oracle access to the actual input $x \in \{0,1\}^{|p|}$. A natural example as the property that is parameterized by CNF formulas and consists of all truth assignments that satisfy the CNF formula (used as a parameter). A generalization of Theorem 2 refers to massively parameterized properties and to the circuit complexity of deciding them (i.e., given p and x, does $x \in \Pi_p$ hold). This generalization also applies to the property that is parameterized by depth D circuits of size S, in which case the decision problem (of circuit evaluation) is the universal problem of sets that are decidable by such circuits.[16]

4 On the limitations of constant-round IPPs

In this section we present lower bounds on the query and communication complexities of constant-round IPPs. These lower bounds demonstrate inherent limitations of constant-round IPP systems.

As usual, we state these bounds for a sufficiently small constant value of the proximity parameter. Hence, these bounds are stated in terms of n only. We let q and p denote the query and (prover) communication complexities of the IPP.

The starting point and main lower bound of [15] is the following result, which refers to AMP (i.e., AM-proofs of proximity, the "property testing version" of AM).

Theorem 3 (lower bound on AMP): *There exists a property Π in log-space uniform $\mathcal{NC}$ such that any AMP system for Π must satisfy $p(n) + q(n) = \Omega(\sqrt{n/\log n})$, where $p(n)$ denotes the number of bits sent by the prover and $q(n)$ denotes the query complexity of the verifier.*

[14] This Arithmetic circuit (over $\mathcal{F}$) multiplies the input by the parity check matrix of the Reed-Solomon code $P_{\mathcal{F},d}$. We then apply an Arithmetic version of Theorem 2, while using the same field $\mathcal{F}$ (where in its proof we may use $H = \{0,1\}$ or any $H \subset \mathcal{F}$ of size $\text{poly}(\log |F|)$). We also comment that we don't need the Arithmetic circuit to check that the $|\mathcal{F}| - (d+1)$ values are all zero; it suffices to check a random location in the Reed-Muller encoding of the output layer.

[15] A more general formulation allows $\Pi_p \subseteq \{0,1\}^{\ell(|p|)}$ for some length function $\ell : \mathbb{N} \to \mathbb{N}$, where typically $\ell(n) \in [n^{\Omega(1)}, n^{O(1)}]$.

[16] In fact, the presentation of the GKR-protocol can be simplified in this case; that is, rather than presenting an IPP for every set that has circuits in a class $\mathcal{C}$, it suffices to present an IPP for the set $\{(C,x) \in \mathcal{C} \times \{0,1\}^* : U(C,x) = 1\}$, where U is a universal machine that evaluates circuits in $\mathcal{C}$ (cf. [7, Sec. 3]).

Proof Sketch: The key observation underlying the proof is that the acceptance probability of such an AMP on input x equals the probability that a DNF of size $s = \exp(p(|x|) + q(|x|))$ that is selected from a specific distribution is satisfied by $x = (x_1, ..., x_n)$. We stress that the distribution of DNFs (in the variables $x_1, ..., x_n$) depends only on $|x|$ and on the verifier's strategy.

Specifically, each DNF in the distribution is associated with a different choice of the verifier's random choice (i.e., message). The DNF associated with the random choice r is a disjunction of 2^{p+q} terms, where each term corresponds to a different choice of a (p-bit long) prover-message and (the q) answers of the oracle x (assuming that the resulting $(p+q)$-bit long value causes the verifier to accept). Each of these terms "checks" that the relevant bits in x match the fixed answer bits associated with the term (where each answer-bit location is determined by r, the prover-message, and the prior answer-bits).[17]

Using a pseudorandom generator with seed-length $k = O((\log s)^2 \cdot \log n)$ that fools DNFs of size s over n variables, the theorem follows (provided that $k < n/2$ or so). Specifically, let $G : \{0,1\}^k \to \{0,1\}^n$ be such a pseudorandom generator (cf. [2]) and consider the set $\Pi = \{G(s) : s \in \{0,1\}^k\}$. Then, an IPP for Π must accept inputs in Π (w.h.p.), while rejecting random n-bit inputs (w.h.p.), since the latter are $\Omega(1)$-far from Π (w.h.p). This yields a contradiction unless the query and communication complexity of this IPP, denoted p and q respectively, satisfy $O((p(n) + q(n))^2 \cdot \log n) > n/2$ (equiv., $p(n) + q(n) = \Omega(\sqrt{n/\log n})$). ■

Obtaining results for constant-round IPPs. For any constant $r \in \mathbb{N}$, using the known transformation of A[MA]^r to AM, one derives a result for r-round public-coin IPPs.[18] Specifically, following the presentation in [5, Apdx F.2.2.1], we observe that an interactive proof of A[MA]^r-type in which the prover sends $p(n)$ bits is converted to an AM-proof in which the prover sends $O(p(n))^r$ bits and the verifier invokes the original decision predicate on $O(p(n))^r$ transcripts.[19] In the context of IPPs, this means that an IPP of $A[MA]^r$-type in which the prover sends $p(n)$ bits is converted to an AMP in which the prover sends $O(p(n))^r$ bits and the verifier makes $O(p(n))^r \cdot q(n)$ queries, where $q(n)$ is the number of queries made in the A[MA]^r-type IPP. Applying Theorem 3, we get

Theorem 4 (lower bound on constant-round public-coin IPPs): *There exists a property Π in log-space uniform $\mathcal{NC}$ such that, for every constant $r \in \mathbb{N}$, any r-round public-coin IPP for Π must satisfy $p(n)^r \cdot q(n) = \Omega(\sqrt{n/\log n})$, where $p(n)$ and $q(n)$ are as in Theorem 3.*

[17] Specifically, the DNF associated with the random choice r has the form

$$\bigvee_{(\beta,a_1,...,a_q)\in\{0,1\}^{p+q}} \bigwedge_{i\in[q]} (x_{Q_i(r,\beta,a_1,...,a_{i-1})} = a_i) \quad \wedge D(r,\beta,a_1,...,a_q),$$

where $Q_i(r,\beta,a_1,...,a_{i-1}) \in [n]$ (resp., $D(r,\beta,a_1,...,a_q) \in \{0,1\}$) is the i^{th} query that the verifier performs (resp., the verifier's final decision) when using the random-pad r, and receiving the prover's message β and the prior oracle answers $a_1,...,a_{i-1}$ (resp., the oracle answers $a_1,...,a_q$). Needless to say, $x_j = a_i$ represents the literal x_j if $a_i = 1$ and $\neg x_j$ otherwise.

[18] Indeed, A[MA]^r = [AM]^{r}A is a r-round system, since the last A-move only generates coins for the verifier's decision.

[19] It is tempting to try to use the more sophisticated strategy presented in [5, Apdx F.2.2.2], which converts an interactive proof of A[MA]^r-type (in which the prover sends $p(n)$ bits) to an AMA-proof in which the prover sends $O(p(n))^r$ bits and the verifier invokes the original decision predicate once. Unfortunately, we do not know whether Theorem 3 extends to IPP of the AMA-type, whereas converting the resulting AMA-type system to an AM-type one would yield the same result as Theorem 4.

Actually, Theorem 4 extends to varying r; in that case we get $(\mathrm{poly}(r(n)) \cdot p(n))^{r(n)} \cdot q(n) = \Omega(\sqrt{n/\log n})$.

As stated in [15], an extension to *general* (constant-round) IPPs is possible too.[20] Specifically, following the presentation in [5, Apdx F.2.1], any r-round IPP with p and q as in Theorem 3 implies an $(r+3)$-round *public-coin* IPP of query complexity $q' = O(q \cdot \log n)$ in which the prover sends messages of total length $p' = O(p \cdot \log n) + O(\rho \cdot \log n)^2$, where ρ denotes the randomness complexity of the original IPP.[21] The randomness complexity of the resulting IPP is $\rho' = O(\rho \cdot \log n)$. (A key observation is that, in the context of standard interactive proofs, the resulting verifier invokes the error-reduced version of the original verifier only once.)

Lastly, we mention that the transformation of (public-coin) interactive proof systems with two-sided error into ones with one-sided error also extends to the IPP setting. Specifically, following the presentation in [4], an r'-round public-coin IPP with p', q' and ρ' as above, implies an $(r'+1)$-round public-coin IPP with one-sided error of query complexity $q'' = O(\rho' \cdot q' \cdot \log^2 n)$ in which the prover sends messages of total length $p'' = O(\rho' \cdot \log n)^2 + \rho' \cdot p' \cdot \mathrm{poly}(\log n)$.[22]

Digest. In all three cases, we adapted the (black-box) transformations that are well-known in the context of standard interactive proof systems to the context of IPP systems. A key parameter that we considered is the number of times that the resulting verifier invokes the original verifier. This number determines the query complexity blow-up factor (when starting with the query complexity of the original verifier), whereas the comminication complexity may grow also due to other issues.

Acknowledgments

I am grateful to Guy Rothblum for important comments on an early draft of this text. I also wish to thank Noga Amit for many discussions regarding [15].

Appendix: Proof of the claim in Footnote 11

Footnote 11 ends with the claim that, for every $x_h, s \in \{0,1\}^{|H|}$ and a affine space Π_h, with high probability over the random choice of $r \in \mathcal{F}$, the distance of $r \cdot x_h$ from $r \cdot \Pi_h + s$ is at least half the distance of x_h from Π_h. This claim is proved below, where Π_h is replaced by $V + w$ (and $|H|$ and x_h are replaced by k and v, respectively). (The following proof follows the proof of [14, Lem. 5.8], which in turn follows the proof of [15, Lem. 1.6 (full version)].)

[20] Actually, since our focus is on the query and communication complexities, while ignoring the computational complexity of the verifier, we may employ a straightforward transformation in which the verifier's messages are replaced by coin-tosses to the corresponding conditional probability spaces. The original messages and queries are computed accordingly; but, indeed, this may involve exponential-time computations.

[21] The $O(\log n)$ factors are due to error reduction, whereas the added $O(\rho \cdot \log n)^2$ term (in p') is due to the prover's message in the random-selection protocol that is used to generate verifier messages. Recall that the random-selection protocol that we use is of the MAM-type and proceeds as follows: First, the prover selects and sends a description of a partition (which is specified using an ρ'-wise independent hash function). Next, the verifier specifies a random cell in this partition, and finally the prover replies with a random element in that cell.

[22] Again, the $O(\log n)$ factors are due to error reduction, whereas the $O(\rho' \cdot \log n)^2$ term (in c'') is due to sending the "shifts" for the verifier's coins. Recall that the parties execute $O(\rho' \log n)$ copies of the original IPP.

Claim: *For a linear space $V \subseteq \mathcal{F}^k$ and $w \in \mathcal{F}^k$, suppose that $v \in \mathcal{F}^k$ is at distance δ from the affine space $V + w$. Then, for every $s \in \mathcal{F}^k$, there exists at most one $r \in \mathcal{F}$ such that the distance of $r \cdot v$ from $V + r \cdot w + s$ is smaller than $\delta/2$.*

Proof: Let $u \stackrel{\text{def}}{=} v - w$, and suppose towards the contradiction that there exists two distinct $r_1, r_2 \in \mathcal{F}$ such that both $r_1 \cdot u - s$ and $r_2 \cdot u - s$ are at distance smaller than $\delta/2$ from V. That is, for both $i \in \{1, 2\}$, there exist a vector $v_i \in V$ and a set $I_i \subseteq [k]$ of size smaller than $\delta k/2$ such that $r_i \cdot u - s$ and v_i differ only on indices in I_i. Then, for every $c', c'' \in \mathcal{F}$, it holds that $c' \cdot u + c'' \cdot s$ differs from some vector in V only on indices in $I = I_1 \cup I_2$ (i.e, consider $\alpha, \beta \in \mathcal{F}$ such that $\alpha \cdot (r_1 \cdot u - s) + \beta \cdot (r_2 \cdot u - s) = c' \cdot u + c'' \cdot s$, and note that it differs from $\alpha \cdot v_1 + \beta \cdot v_2 \in V$ only on I). Using $c' = 1$ and $c'' = 0$, we conclude that $u = v - w$ is $(|I|/k)$-close to V, which contradicts the hypothesis (since $|I| < \delta \cdot k$). ■

Comment: An almost equivalent (and somewhat more appealing) phrasing of conclusion of the foregoing claim asserts that *for every $s \in \mathcal{F}^k$, there exists at most one $r \in \mathcal{F}$ such that the distance of $v - w + r \cdot s$ from V is smaller than $\delta/2$.* (Clearly, if such an r exists, then it is not 0 (and $s \neq 0^k$).) Intuitively, this alternative form asserts that if $s' = r \cdot s$ manages to shift $v - w$ to distance smaller than $\delta/2$ from V, then very other multiple of s' fails to do so.

References

[1] Noga Amir, Oded Goldreich, and Guy Rothblum. Doubly Sub-Linear Interactive Proofs of Proximity. In *16th ITCS*, pages 6:1–6:25, 2025.

[2] Louay Bazzi. Polylogarithmic Independence Can Fool DNF Formulas. *SIAM Journal on Computing*, Vol. 38 (6), pages 2220–2272, 2009. Preliminary version in *48th FOCS*, 2007.

[3] Inbar Ben Yaacov, Oded Goldreich, and Guy Rothblum. Design Methodologies for Interactive Proof Systems. In preparation, 2026.

[4] Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On Completeness and Soundness in Interactive Proof Systems. *Adv. Comput. Res.*, Vol. 5, pages 429–442, 1989. See preliminary version in *28th FOCS*, pages 449–461, 1987.

[5] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[6] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[7] Oded Goldreich. On Doubly-Efficient Interactive Proof Systems. *Found. Trends Theor. Comput. Sci.*, Vol. 13 (3), pages 158–246, 2018.

[8] Shafi Goldwasser, Yael Tauman Kalai, and Guy Rothblum. Delegating Computation: Interactive Proofs for Muggles. *Journal of the ACM*, Vol. 62 (4), pages 27:1–27:64, 2015. Preliminary version in *40th STOC*, 2008.

[9] Tom Gur and Ron Rothblum. Non-Interactive Proofs of Proximity. *Comput. Complex.*, Vol. 27 (1), pages 99–207, 2018. Preliminary version in *ECCC*, TR13-078, 2013.

[10] Yael Tauman Kalai and Ron D. Rothblum. Arguments of Proximity (Extended Abstract). In *35th CRYPTO*, pages 422–442, 2015.

[11] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.

[12] Or Meir. IP = PSPACE Using Error-Correcting Codes. *SIAM Journal on Computing*, Vol. 42 (1), pages 380–403, 2013.

[13] Omer Reingold, Guy Rothblum, and Ron Rothblum. Constant-Round Interactive Proofs for Delegating Computation. *SIAM Journal on Computing*, Vol. 50 (3), 2021. Preliminary version in *48th STOC*, 2016.

[14] Guy Rothblum and Ron Rothblum. Batch Verification and Proofs of Proximity with Polylog Overhead. In *18th TCC* (Part II), LNCS (Vol. 12551), Springer, pages 108–138, 2020.

[15] Guy Rothblum, Salil Vadhan, and Avi Wigderson. Interactive Proofs of Proximity: Delegating Computation in Sublinear Time. In *45th ACM Symposium on the Theory of Computing*, pages 793–802, 2013. Full version available from Guy Rothblum's website.[23]

[23] See `https://guyrothblum.wordpress.com/wp-content/uploads/2014/11/rvw13.pdf`

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il