2013 IEEE 54th Annual Symposium on Foundations of Computer Science

Constant rate PCPs for circuit-SAT with sublinear query complexity

Eli Ben-Sasson, Yohay Kaplan
CS department, Technion and CSAIL, MIT
Haifa, Israel and Cambridge, USA
eli@cs.technion.ac.il, yohayk@cs.technion.ac.il

Swastik Kopparty
Dept of Mathematics and Department of CS
Rutgers University
New Brunswick, USA
swastik.kopparty@rutgers.edu

Or Meir
Institute for Advanced Study Princeton, USA
ormeir@ias.edu

Henning Stichtenoth
MDBF Sabanci University
Istanbul, Turkey

***Abstract*—The PCP theorem (Arora et. al., J. ACM 45(1,3)) says that every NP-proof can be encoded to another proof, namely, a probabilistically checkable proof (PCP), which can be tested by a verifier that queries only a small part of the PCP. A natural question is how large is the blow-up incurred by this encoding, i.e., how long is the PCP compared to the original NP-proof. The state-of-the-art work of Ben-Sasson and Sudan (SICOMP 38(2)) and Dinur (J. ACM 54(3)) shows that one can encode proofs of length n by PCPs of quasi-linear length that can be verified using a constant number of queries. In this work, we show that if the query complexity is relaxed to polynomial, then one can construct PCPs of linear length for $circuit - SAT$, and PCPs of length O(tlog t) for any language in NTIME(t). Our PCPs have perfect completeness and constant soundness. This is the first constant-rate PCP construction that achieves constant soundness with nontrivial query complexity.**

Our proof replaces the low-degree polynomials in algebraic PCP constructions with tensors of transitive algebraic geometry (AG) codes. We show that the automorphisms of an AG code can be used to simulate the role of affine transformations which are crucial in earlier high-rate algebraic PCP constructions. Using this observation we conclude that any asymptotically good family of transitive AG codes over a constant-sized alphabet leads to a family of constant-rate PCPs with polynomially small query complexity. Such codes are constructed for the first time for every message length.

***Keywords*-computational complexity; probabilistically checkable proofs; AG codes;**

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258.

This material is based upon work supported by the National Science Foundation under Grant No. NSF CCF-1253886.

Partially supported by NSF grant CCF-0832797.

This work was partially supported by Tubitak, proj. no. 111T234

A full version of this paper is available at http://eccc.hpi-web.de/report/2013/085/

I. Introduction

The PCP theorem [AS98], [ALM+98] is one of the major achievements of complexity theory. A PCP is a proof system that allows checking the validity of a claim by querying only a small part of the proof. The PCP theorem says that every NP-claim has a PCP of polynomial length that can be verified using a constant number of queries. The theorem has found many applications, most notably in establishing lower bounds for approximation algorithms for constraint satisfaction problems (cf. the surveys [Aro02], [GO05] and references therein).

It is natural to ask how long should the PCPs be compared to the corresponding NP-proofs. To make the discussion a bit more formal, let L be a language in NP, and recall that there is a polynomial-time algorithm V that verifies the membership of a string x in L when given an additional NP-witness. Let $t : \mathbb{N} \to \mathbb{N}$ be such that $t(n)$ is an upper bound on the running time of V on inputs of length n. The original PCP theorem says that there exists a PCP verifier that verifies claims of the form $x \in L$ by making $O(1)$ queries to proofs of length $\mathrm{poly}\,(t(n))$ where $n = |x|$. Following works have improved this state of affairs [PS94], [HS00], [GS06], [BSVW03], [BGH+06] culminating in [BS08], [Din07] which construct PCP verifiers that make $O(1)$ queries to proofs of length $t(n) \cdot \mathrm{poly}\log t(n)$. It is an interesting open question whether this can be further improved.

While the aforementioned works have focused mostly on PCPs that use a constant number of queries, it is also interesting and natural to consider PCPs that make a larger number of queries. In fact, even constructing a PCP that uses any sub-linear number of queries is interesting and non-trivial. In particular, such PCPs have applications to succinct verification of computer programs, as first suggested in [BFLS91] and improved

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.42

upon in [BGH+05], [Mie09], [BCGT13a], [BCGT13b]. Thus, it is natural to ask whether we can get PCPs with better length if we allow ourselves to use more queries, say, $O(n^{\varepsilon})$ queries. Indeed, recent constructions of locally decodable codes [KSY11], and locally testable codes [Vid12], show that by allowing $O(n^{\varepsilon})$ queries, it is possible to achieve very high rates, arbitrarily close to 1.

In this work, we show that for the special case of $circuit - SAT$, there exists a (non-uniform) PCP verifier that verifies the satisfiability of a circuit of size n by making n^{ε} queries to a proof of length $O_{\varepsilon}(n)$ for all $\varepsilon > 0$. Using the efficient reduction of NP to $circuit - SAT$ of [PF79], this implies the existence of a PCP verifier that makes $O(t^{\varepsilon})$ queries to a proof of length $O_{\varepsilon}(t \log t)$ for every language $L \in \mathsf{NTIME}(t)$.

Remark 1.1: As noted above, our PCPs are non-uniform. The reason is that our construction relies on a family of algebraic-geometry codes which we do not know how to construct in polynomial time.

A. Our techniques

In order to explain the new ideas that we employ to get linear PCPs, let us first recall how the state-of-the-art PCP of [BS08], [Din07] was constructed, and what caused the poly-logarithmic blow-up in the length there. Very roughly, that construction applies the following steps to an instance φ of $circuit - SAT$ of size n: (1) (*Graph problem*) Reducing φ to a constraint satisfaction problem (CSP) over a "well-structured" graph G of size m_1. In this context, a graph is said to be "well-structured" if we can identify its vertices with the vectors of some vector space, such that the neighbors of a vertex v can be obtained by applying a collection of affine transformations to v (where the affine transformations are the same for all vertices). This reduction uses routing [Lei92], which loses a logarithmic factor in the length of the PCPs, i.e., $m_1 \geq n \log n$.

(2) (*Arithmetization*) Reducing the foregoing constraint satisfaction problem to an algebraic CSP (ACSP). As part of this reduction, binary strings of length m_1 are represented by evaluations of polynomials of degree m_1 over a field $\mathbb{F}$ which is of size greater than m_1. In particular, if we measure the length of the latter encoding in bits rather than in elements of the field, their length will become $m_2 \geq m_1 \log m_1$.

(3) (*Zero testing*) An ACSP instance obtained via arithmetization is specified by a low-degree polynomial Q. The instance is defined to be satisfiable if and only if there exists a low-degree polynomial P such that when Q is "composed" with P then the resulting polynomial, denoted R, is one that vanishes on a large predefined set of points $H \subset \mathbb{F}$. Roughly speaking, P is supposed to be the polynomial interpolating a boolean assignment that satisfies the graph CSP G, hence $\deg(P) \approx m_1$, and Q "checks" the algebraic analog of each and every constraint of the graph CSP.

To verify that Q is satisfiable, one needs to solve the "zero testing" problem which asks whether R indeed vanishes on every point in H. In [BS08] this reduction is done by an algebraic characterization of polynomials vanishing on H. Other works in the PCP literature (e.g., [AS98], [ALM+98]) have solved the "zero testing" problem using the sum-check protocol of [LFKN92].

(4) (*Low-degree testing*) The zero-testing procedures mentioned above only work under the promise that P and R are close to low-degree polynomials[1]. Thus, in order to verify that the ACSP instance is satisfiable, we need to be able to verify that a function is close to a low-degree polynomial. This is done in [BS08] by constructing a PCP of proximity [BGH+06], [DR06] for testing that a polynomial is of low degree. This step uses $\tilde{O}(\sqrt{n})$ queries and loses only a constant factor in the length of the PCP.

(5) (*Composition*) Reducing the query complexity to a constant by using more composition with PCPs of proximity and gap-amplification. This step loses a poly-logarithmic factor in the length of the PCP: Denoting the final PCP length by m we have $m = m_2 \cdot \mathrm{poly} \log m_2 = n \cdot \mathrm{poly} \log n$.

Thus, in order to construct a PCP of linear length for $circuit - SAT$, we need to find ways to deal with the losses in the Steps 1, 2, and 5 above, while supporting the functionality of steps 3 and 4

For Step 1, we observe that since we are going to construct a PCP with a large query complexity, we can afford to use "well-structured" graphs G' of significantly larger degree (specifically, n^{ε}), in which case the routing loses only a constant factor, i.e. $m'_1 = O_{\varepsilon}(n)$. In contrast, the work of [BS08] uses graphs of constant degree, for which the routing must lose a logarithmic factor. However, in order to support the following steps, we must generalize the definition of "well-structured graph" to settings of AG-codes. Basically, our generalization replaces the affine transformations with automorphisms of the corresponding AG-code.

For Step 2, we reduce the size of the finite field used in the algebraic CSP to a constant. This is done by replacing the Reed-Solomon (RS) code with a transitive AG code that has an alphabet of constant size, which results in codewords of bit-length $m'_2 = O(m'_1) = O(n)$. This replacement is non-trivial, however, and also causes

[1] Actually, those procedures also rely on the promise that some additional auxiliary functions are close to low-degree polynomials.

complications in Steps 3 and 4 which are discussed in the next subsection.

For Step 3, we present two ways for solving the zero testing for AG codes: the first solution follows the ideas of [BS08] by generalizing the "Combinatorial Nullstellensatz" of [Alo99] to tensor products of AG-codes yielding a new "AG Combinatorial Nullstellensatz".

The second method is based on the sum-check protocol of [LFKN92], and in particular, uses the generalization of the sum-check protocol to general error-correcting codes of [Mei10]. The PCP constructed this way has the advantage that it requires a less sophisticated algebraic machinery, and in particular, it does not require the AG combinatorial Nullstellensatz. However, this PCP is less randomness efficient.

In order to emulate Step 4, we need to solve the analog of low-degree testing for AG codes with query complexity n^{ε}. To this end, we use *tensor products* of the AG codes rather than the AG codes themselves. We then use the fact that tensor codes are locally testable[2] as an analog of low-degree testing.

We do not have an analogue of Step 5 in our PCP construction. We currently do not know a way of composing our PCP to reduce the query complexity below n^{ε} while keeping the rate constant.

B. AG arithmetization

We briefly discuss a number of issues that arise from the use of AG codes in PCP constructions, as this is the first time such codes are used in the context of PCPs.

Informal description of AG codes: Codewords of an AG code C are best thought of as (rational) functions evaluated over a specially chosen set of points, i.e., $C = \{f : D \to \mathbb{F}_q \mid f \in L\}$. The set of points $D \subset \mathbb{F}_q^m$ is the set of solutions to a system $\mathcal{E} = (e_1, \ldots, e_k), e_i \in \mathbb{F}_q(x_1, \ldots, x_m)$ of k carefully chosen rational equations over $\mathbb{F}_q$. $D = \left\{\overline{x} = (x_1, \ldots, x_m) \in \mathbb{F}_q^m \mid e_1(\overline{x}) = \ldots = e_k(\overline{x}) = 0\right\}$ The set L of "legitimate" functions is a linear space that is best thought of the space of "low-degree" rational functions in $x_1, \ldots, x_m$. AG codes are interesting because by fixing the base-field $\mathbb{F}_q$ and letting m grow, one can obtain a family of codes over constant alphabet $\mathbb{F}_q$ with arbitrarily large dimension and block-length. Indeed, the celebrated results of [TVZ82], [GS96] show that using this framework one can obtain explicit constructions of asymptotically good codes that beat the Gilbert-Varshamov bound.

[2]The study of local testability of tensor codes was initiated by [BS06] and further studied in [Val05], [CR05], [DSW06], [BV09], [BSV09], [GM12], [Mei12b], [Vid12]. We use the state-of-the-art testability results of [Vid12] (cf. Theorem 3.9)

Why AG codes?: A key property of RS and Reed-Muller (RM) that is used in the arithmetization steps of previous works (and in particular, in [BS08]) is their "multiplication property": Let f, g be two codewords of the RS code of degree d, i.e., $f, g : \mathbb{F} \to \mathbb{F}$ are evaluations of polynomials of degree at most d. Then their coordinate-wise multiplication is the function $f \cdot g$ defined by $(f \cdot g)(x) \stackrel{\text{def}}{=} f(x) \cdot g(x), x \in \mathbb{F}$. Clearly, $\deg(f \cdot g) \leq 2d$, so we conclude that $f \cdot g$ is a codeword of a code with relative distance $2d/|\mathbb{F}|$. Taking $|\mathbb{F}|$ to be sufficiently large means $f \cdot g$ belongs to a large-distance code. As shown by [Mei10], [Mei12a], this "distance of multiplication code" property is sufficient for a PCP-style arithmetization. AG codes are a natural generalization of "low-degree" codes (under the proper definition of "degree") and, in particular, have the "distance of multiplication code" property needed for PCPs.

The main advantage AG codes have over RS/RM is their constant-size alphabet. All known PCP constructions based on RS/RM (and AG) codes suffer a $\log |\mathbb{F}|$-factor loss in their rate because, roughly speaking, they are used to encode *boolean* assignments to a $circuit-SAT$ instance. Constant-rate RS/RM codes of blocklength n require fields of size $n^{\Omega(1)}$ which implies a rate-loss of $\Omega(\log n)$. Using constant-rate AG codes over a constant-size alphabet allows us to avoid this loss.

Why transitive?: Another property of RS codes that is used in previous arithmetizations is the fact that composing a degree-d polynomial with an affine function results in a degree-d polynomial. This property is combined with the notion of "well-structured graphs" to yield an algebraic constraint satisfaction problem in Step 2 that is of low-degree. We point out that the reason all this works out is because affine functions are automorphisms of the RS code. When generalizing the notion of "well-structured graphs" to our AG-setting it is sufficient to work with AG-codes that have a transitive automorphism group.

The affine-invariance of linear codes has been intensely investigated in recent years in the context of locally testable codes, starting with the work of [KS08] (see [Sud10] for a recent survey). The role the automorphisms of AG codes play in constructing locally correctable and decodable codes has been recently considered in [BSGK+13].

The properties required for our PCP construction.: The multiplication property and the transitivity property discussed above are sufficient for the our PCP construction that is based on the sum-check protocol. Theoretically, this construction could be implemented using any family of error-correcting that has this property. However,

practically, we do not know other examples of such codes.

Our first PCP construction, on the other hand, relies crucially on our codes being AG codes, and in particular on the AG Combinatorial Nullstellensatz theorem to be discussed next. However, this construction is more randomness-efficient, and is also somewhat simpler assuming the AG Combinatorial Nullstellensatz.

C. The road ahead

In the next section we formally state our main results. Section III states some preliminaries about tensor codes and their testability, an asymptotically good family of transitive AG codes and a special case of the AG combinatorial Nullstellensatz sufficient for the analysis of our PCP construction. Section IV gives the mian details of one of the proofs of the Main Theorem 2.3. The complete proofs, accurate statements of AG related theorems and the construction of the transitive AG codes used can all be found in the full paper.

II. Formal Statement of Main Results

Throughout this paper, when we discuss circuits, we always refer to boolean circuits with AND, OR, and NOT gates whose fan-in and fan-out are upper bounded by 2. The *size* of the circuit φ, denoted $|\varphi|$, is defined to be the number of wires of φ. We say that a circuit φ is *satisfiable* if there is an input $x \in \{0,1\}^*$ for φ such that $\varphi(x) = 1$.

Definition 2.1: $circuit - SAT \stackrel{\text{def}}{=} \{\varphi : \varphi \text{ is a satisfiable circuit}\}$.

Definition 2.2 (Non-uniform PCP verifier): Let $L \subseteq \{0,1\}^*$ be a language, and let $q, \ell, v : \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \to (0,1)$. A *(non-uniform) PCP verifier* $V = \{V_n\}_{n=1}^{\infty}$ for L with *query complexity* q, *proof length* ℓ, *rejection probability* ρ and *verifier complexity* v is an infinite family of randomized circuits that satisfy the following requirements:

1) **Input:** The verifier V_n takes as input a string w of length n.
2) **Output:** The verifier V_n outputs a tuple I of coordinates in $\{1, \ldots, \ell(n)\}$ where $|I| \leq q(n)$, and a circuit $\psi : \{0,1\}^I \to \{0,1\}$ of size at most $poly(n)$. For $\pi \in \{0,1\}^{\ell(n)}$ we denote by $\pi|_I$ the restriction of π to I.
3) **Verifier complexity:** The size of the circuit V_n is at most $v(n)$.
4) **Completeness:** For every $w \in L$, there exists a string $\pi \in \{0,1\}^{\ell(n)}$ such that $\mathbb{P}[\psi(\pi|_I) = 1] = 1$, where the probability is over ψ and I generated by the verifier V on input w.
5) **Soundness:** For every string $w \notin L$ and every string $\pi \in \{0,1\}^{\ell(n)}$, it holds that $\mathbb{P}[\psi(\pi|_I) = 0] \geq \rho(n)$, where the probability is over ψ and I generated by the verifier V on input w.

We can now state our main result.

Theorem 2.3 (Main theorem): For every $\varepsilon > 0$ there exists a constant $c_\varepsilon = 2^{O(1/\varepsilon)}$ such that the following holds for every $n \in \mathbb{N}$. There exists a PCP verifier for $circuit - SAT_n$ with query complexity $c_\varepsilon \cdot n^\varepsilon$, proof length $c_\varepsilon \cdot n$, rejection probability $1/2$ and verifier complexity $(c_\varepsilon \cdot n)^{O(1)}$.

The proof of Theorem 2.3 goes by the following schematic sequence of reductions, explained next.

$$circuit - SAT \xrightarrow{(i)} \textsf{Hypercube-CSP} \xrightarrow{(ii)} \textsf{aggregate-AGCSP} \begin{cases} \xrightarrow{(iiia)} \text{Nullstellensatz} \\ \xrightarrow{(iiib)} \text{Sum-check} \end{cases} \quad (1)$$

(i) The first reduction maps an instance φ of $circuit - SAT$ to a graph constraint satisfaction problem over a sub-graph of the hypercube. *(ii)* The next reduction maps the hypercube constraint satisfaction problem into an Algebraic Geometry Constraint Satisfaction Problem (AGCSP). At this point we have two alternate paths to complete the proof of Theorem 2.3. *(iiia)* The first uses our AG combinatorial Nullstellensatz (Theorem 3.14) and relies on particular properties of tensored AG-codes. *(iiib)* The second is based on the sum-check protocol, and relies only on the multiplication property and transitivity of our AG codes, but is less randomness efficient. We elaborate further on all these stages in section IV.

III. Tensors of AG Codes and an AG Combinatorial Nullstellensatz

This section reviews the basic properties of tensor product codes as well as their local testability and describes the AG codes that we use and our AG Combinatorial Nullstellensatz which pertains to tensors of AG codes. We assume familiarity with standard coding theory terms and notations.

A. Tensor Codes

In this section, we define the tensor product operation on codes and present some of its properties. See [MS88] and [Sud01, Lect. 6 (2.4)] for the basics of this subject.

Definition 3.1: Let $R : \mathbb{F}^{k_R} \to \mathbb{F}^{\ell_R}$, $C : \mathbb{F}^{k_C} \to \mathbb{F}^{\ell_C}$ be codes. The *tensor product code* $R \otimes C$ is a code of message length $k_R \cdot k_C$ and block length $\ell_R \cdot \ell_C$ that encodes a message $x \in \mathbb{F}^{k_R \cdot k_C}$ as follows: In order to

encode x, we first view x as a $k_C \times k_R$ matrix, and encode each of its rows via the code R, resulting in a $k_C \times \ell_R$ matrix x'. Then, we encode each of the columns of x' via the code C. The resulting $\ell_C \times \ell_R$ matrix is defined to be the encoding of x via $R \otimes C$.

The following fact lists some of the basic and standard properties of the tensor product operation.

Fact 3.2: Let $R : \mathbb{F}^{k_R} \to \mathbb{F}^{\ell_R}$, $C : \mathbb{F}^{k_C} \to \mathbb{F}^{\ell_C}$ be linear codes. We have the following:

1) An $\ell_C \times \ell_R$ matrix x over $\mathbb{F}$ is a codeword of $R \otimes C$ if and only if all the rows of x are codewords of R and all the columns of x are codewords of C.
2) Let δ_R and δ_C be the relative distances of R and C respectively. Then, the code $R \otimes C$ has relative distance $\delta_R \cdot \delta_C$.
3) The tensor product operation is associative. That is, if $D : \mathbb{F}^{k_D} \to \mathbb{F}^{\ell_D}$ is a code then $(R \otimes C) \otimes D = R \otimes (C \otimes D)$.

The associativity of the tensor product operation allows us to use the following notation:

Notation 3.3 (Iterated tensor code): Let $C : \mathbb{F}^k \to \mathbb{F}^\ell$ be a code. For every $m \in \mathbb{N}$ we denote by $C^{\otimes m} : \mathbb{F}^{k^m} \to \mathbb{F}^{\ell^m}$ the code $C^{\otimes m} = C^{\otimes m-1} \otimes C$. Suppose that C is an evaluation code, i.e., we identify the codewords of C with functions $f : D \to \mathbb{F}$ for some set D. In such case, we will identify the codewords of $C^{\otimes m}$ with functions $g : D^m \to \mathbb{F}$.

Notation 3.4 (Axis-parallel lines): For $i \in [m]$ and $\overline{v} = (v_1, \ldots, v_m) \in D^m$ the set $D^m|_{i,\overline{v}} = \{(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_m) \mid x \in D\}$ is called the *i-axis-parallel line passing through $\overline{v}$*. Similarly, the restriction of g to this axis-parallel line, denoted $g|_{i,\overline{v}}$, is the function with range D defined by $g|_{i,v-i}(x) = g(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_m), \quad x \in D$.

The following can be proven using Fact 3.2:

Fact 3.5: Let C be a linear code whose codewords are identified with functions $f : D \to \mathbb{F}$. Then, a function $g : D^m \to \mathbb{F}$ is a codeword of $C^{\otimes m}$ if and only if for every $1 \leq i \leq m$ and $\overline{v} \in D^m$ it holds that the function $g|_{i,\overline{v}}$ is a codeword of C.

Another useful fact about the tensor products of systematic evaluation codes is the following.

Fact 3.6: Let $C = \{f : D \to \mathbb{F}\}$ be a systematic linear evaluation code whose messages are functions $h : H \to \mathbb{F}$ (where $H \subseteq D$). Then, $C^{\otimes m} = \{f_m : D^m \to \mathbb{F}\}$ is a systematic evaluation code whose messages are functions $h_m : H^m \to \mathbb{F}$.

We also use the following two claims, due to [Mei10].

Claim 3.7 ([Mei10, Claim 3.7]): Let $C = \{f : D \to \mathbb{F}\}$ be a systematic linear evaluation code whose messages are functions $h : H \to \mathbb{F}$ (where $H \subseteq D$), and let $m \in \mathbb{N}$. Then, for every coordinate $\overline{x} \in D^m$ there exist scalars $\alpha_{t,z} \in \mathbb{F}$ (for every $1 \leq t \leq m$ and $z \in H$) such that for every codeword $g \in C^{\otimes m}$ it holds that $g(\overline{x}) = \sum_{z_1 \in H} \alpha_{1,z_1} \cdot \sum_{z_2 \in H} \alpha_{2,z_2} \cdot \ldots \sum_{z_m \in H} \alpha_{m,z_m} \cdot g(z_1, \ldots, z_m)$. Furthermore, the scalars $\alpha_{t,z}$ can be computed in polynomial time given $\overline{x}$ and the generating matrix of C. Moreover, for every $t \in [m]$, the scalars $\{\alpha_{t,z}\}_{z \in H}$ depend only on x_t and on the generating matrix of C (but not on $x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_m$).

Claim 3.8 ([Mei10, Claim 3.8]): Let $C = \{f : D \to \mathbb{F}\}$ be a linear evaluation code, let $m \in \mathbb{N}$, and let $g \in C^{\otimes m}$. Then, for every sequence of scalars $\alpha_{t,z}$ (for every $2 \leq t \leq m$ and $z \in D$) it holds that the function $f : D \to \mathbb{F}$ defined by $f(z_1) = \sum_{z_2 \in D} \alpha_{2,z_2} \cdot \sum_{z_3 \in D} \alpha_{3,z_3} \cdot \ldots \sum_{z_m \in D} \alpha_{m,z_m} \cdot c(z_1, \ldots, z_m)$ is a codeword of C.

Finally, in this work we use the fact that tensor product codes are locally testable.

Theorem 3.9: [Vid12] There exists a randomized polynomial-time tester that satisfies the following requirements:

- The tester takes as input the generating matrix of a linear code $C : \mathbb{F}^k \to \mathbb{F}^\ell$ of relative distance δ_C and an integer m and is given oracle access to a string $w \in \mathbb{F}^{\ell^m}$,
- The tester uses at most $\log(\ell^m) + O(\log m)$ random bits and ℓ^2 queries, and performs $\mathrm{poly}(\ell)$ arithmetic operations.
- **Completeness:** If $w \in C^{\otimes m}$, then the tester accepts with probability 1.
- **Soundness:** If $w \notin C^{\otimes m}$, then the tester rejects with probability at least $\gamma_m \cdot \delta(w, C^{\otimes m})$, where $\gamma_m = \delta_C^{3m}/\mathrm{poly}(m)$.

B. AG codes and the multiplication property

The purpose of this section is to state the key properties of the error correcting codes required for our proof of Main Theorem 2.3. We do so here using a limited amount of algebraic geometry. As mentioned in the introduction, two key properties that we need of our codes are that they are part of *multiplication code* families with constant relative distance, and that they possess a *transitive automorphism group.* These notions are defined next.

Definition 3.10 (Multiplication codes): Let C, C' be two evaluation codes with the same domain D. Define their multiplication $C \cdot C' = \mathsf{span}(\{f \cdot f' \mid f \in C, f' \in C'\})$ where $f \cdot f'$ is the function with domain D and range $\mathbb{F}$

defined by $(f \cdot f')(x) = f(x) \cdot f'(x), x \in D$. We define C^i to be the i-fold multiplication $C \cdot C \cdots C$.

A sequence of evaluation codes $\vec{C} = (C_1, \ldots, C_{d_{\mathsf{mult}}})$ with the same domain D is called a **multiplication code family** of **multiplication degree** d_{mult} if for all $1 \leq i, j \leq d_{\mathsf{mult}}$ with $i + j \leq d_{\mathsf{mult}}$, we have $C_i \cdot C_j \subseteq C_{i+j}$.

Definition 3.11: Given a code $C = \{f : D \to \mathbb{F} \mid f \in L\}$, the *automorphism group* of C is the set of permutations of D that stabilize C. Formally, for any permutation $\pi : D \to D$ and codeword $f \in L$ we define $f \circ \pi : D \to \mathbb{F}$ by $(f \circ \pi)(x) = f(\pi(x))$ for $x \in D$. Then $\mathrm{Aut}(C) = \{\pi : D \to D \mid \{f \circ \pi \mid f \in L\} = C\}$ A code C is called *transitive* if its automorphism group is transitive, i.e., for every $x, y \in D$ there exists $\pi \in \mathrm{Aut}(C)$ such that $\pi(x) = y$.

An asymptotically good family of transitive AG codes was presented in [Sti06]. This family was "sparse": the ratio between blocklengths of consecutive members in this family was super-constant. Applied to our framework, this family would only have led to a result saying that infinitely often $circuit - SAT_n$ has constant-rate PCPs with sublinear query complexity. The full paper gives a family of transitive AG codes that is defined for every message length. We now state the main properties of these codes needed for our proof. In what follows an integer q is called a *square of a prime-power* if $q = p^{2r}$ for prime p and integer r.

Theorem 3.12: For any $q = p^{2r} > 4$ a square of a prime-power there exists a constant $c_q \leq p^{\sqrt{q}-1}$ for which the following holds. Fix rate and distance parameters ρ and δ respectively and a multiplication degree parameter d_{mult} which satisfy $c_q \cdot d_{\mathsf{mult}} \cdot \rho + \delta < 1 - \frac{d_{\mathsf{mult}}}{\sqrt{q}-1}$ Then for every sufficiently large message length k there exists a code C, and a multiplication code family $\vec{C} = (C_1, C_2, \ldots, C_{d_{\mathsf{mult}}})$ with $C_1 = C$ such that C is a transitive, $[|D| \leq \frac{k}{\rho}, \geq k, \geq \delta|D|]_q$-code and for each $j \leq d_{\mathsf{mult}}$, the code C_j has relative distance at least δ.

C. The AG Combinatorial Nullstellensatz

In our proof we will face a problem which generalizes the "zero-testing" problem of previous PCPs. In this problem we have a function $g : D^m \to \mathbb{F}$ which is a codeword of $C^{\otimes m}$ where C is an AG code. Our goal is to test whether g vanishes on a set H^m. As shown in [BS08, Lemma 4.9], the zero-testing problem for the case of multivariate polynomials can be solved using Alon's Combinatorial Nullstellensatz, stated next.

Theorem 3.13 (Combinatorial Nullstellensatz [Alo99]): For $H \subset \mathbb{F}_q$, define $\xi_H(Y) = \prod_{\alpha \in H}(Y - \alpha)$. Let $f(X_1, \ldots, X_m)$ be a polynomial over $\mathbb{F}_q$ of individual degree at most d. Then f vanishes on H^m if and only if there exist m polynomials $f'_1, \ldots, f'_m \in \mathbb{F}_q[X_1, \ldots, X_m]$ of individual degree at most d such that $f(X_1, \ldots, X_m) = \sum_{i=1}^m f'_i(X_1, \ldots, X_m) \cdot \xi_H(X_i)$.

The importance of this theorem to PCP constructions is that it reduces the problem of testing many constraints to that of checking that each of $f'_1, \ldots, f'_m$ are low-degree polynomials along with a consistency check that indeed, $f = \sum_{i=1}^m f'_i \cdot \xi_H(X_i)$.

Next we state a special case of it using the minimal AG formalities and tailored for the purpose of proving Theorem 2.3.

Theorem 3.14: Let $C = \{f : D \to \mathbb{F}\}$ and $\vec{C} = (C_1, \ldots, C_{d_{\mathsf{mult}}})$ be the codes from Theorem 3.12 with multiplication degree $d_{\mathsf{mult}} = 6d$, rate parameter ρ, and distance parameter δ satisfying $c_q \cdot d_{\mathsf{mult}} \cdot \rho + \delta < \frac{1}{2} - \frac{d_{\mathsf{mult}}}{\sqrt{q}-1}$. Then for every $H \subset D$ with $|H| < \left(\frac{1}{12} - \frac{\delta}{6} - \frac{2}{\sqrt{q}-1}\right) \cdot |D|$, there exist $\xi(X) = \xi_H(X) \in C_{2d}$ and $\xi'(X) = \xi'_H(X) \in C_{3d}$ satisfying the following. Suppose $f(X_1, \ldots, X_m) \in (C_d)^{\otimes m}$. Then f vanishes on H^m if and only if there exist $f'_1, \ldots, f'_m : D^m \to \mathbb{F}_q$, with $f'_i \in (C_{4d})^{\otimes m}$ for each $i \in [m]$, such that: $f(X_1, \ldots, X_m) \cdot \prod_{i=1}^m \xi'(X_i) = \sum_{i=1}^m f'_i(X_1, \ldots, X_m) \cdot \xi(X_i)$.

IV. Proof of Main Theorem 2.3

A. From circuit-SAT to Hypercube-CSP

Our first reduction is from circuit-SAT to a family of constraint satisfaction problems on sub-graphs of the hypercube. We start by recalling the notions of graph CSP and the hypercube, then state the main step in this reduction (Theorem 4.3). We start by defining constraint satisfaction problems on graphs and on the hypercube graph formally.

A *constraint graph* G is a graph (V, E) coupled with a finite alphabet Σ, and, for each edge $(u, v) \in E$, a binary constraint $c_{u,v} \subseteq \Sigma \times \Sigma$. The *size* of G, denoted $|G|$, is the number of edges of G.

An *assignment to* G is a function $\sigma : V \to \Sigma$. We say that an assignment σ *satisfies* an edge $(u, v) \in E$ if $(\sigma(u), \sigma(v)) \in c_{u,v}$, and otherwise we say that σ *violates* (u, v).

We say that σ is a *satisfying assignment for* G if it satisfies all the edges of G. If G has a satisfying assignment, we say that G is *satisfiable*, and otherwise we say that it is *unsatisfiable*. graph-CSP is the language of satisfiable constraint graphs.

Definition 4.1: The m-dimensional k-ary hypercube,

denoted $\mathcal{H}_{k,m}$, is the graph whose vertex set is $[k]^m$, and whose edges are defined as follows: For each pair of distinct vertices $u, v \in [k]^m$, the vertices u and v are connected by an edge if and only if the Hamming distance between u and v (when viewed as strings) is exactly 1.

Definition 4.2 (Hypercube CSP): Hypercube-CSP is the sub-language of graph-CSP consisting of satisfiable constraint satisfaction problems over graphs that are sub-graphs of a hypercube. We say that G is a sub-graph of H, denoted $G \leq H$, if G can be obtained by deleting edges and vertices of H. Formally, $\text{Hypercube-CSP} \stackrel{\text{def}}{=} \left\{ G : G \in \text{graph-CSP}, \exists H \in \bigcup_{k,m} \mathcal{H}_{k,m} : G \leq H \right\}$.

We now state the first step in our reduction.

Theorem 4.3 (From circuit-SAT to Hypercube-CSP): There exists a polynomial time procedure that maps every circuit φ of size n and integer $m \in \mathbb{N}$ to a constraint graph $G_{\varphi,m}$ over an alphabet Σ of size 4 that is satisfiable if and only if φ is satisfiable, and whose size is at most $2m4^{m+2} \cdot n$. Furthermore, the graph $G_{\varphi,m}$ is a 4-regular subgraph of the m-dimensional k-ary hypercube, where $k \leq 4((4 \cdot n)^{1/m} + 1)$.

B. From Hypercube-CSP to aggregate-AGCSP

We now discuss the second part of our reduction. The starting point is a hypercube CSP problem obtained from Theorem 4.3. The end point will be an instance of a generalization of algebraic CSPs (cf. [BS08]) to AG code settings.

Aggregated Algebraic Geometry Constraint Satisfaction Problems: All previous algebraic PCPs, starting with [AS98], [ALM+98], [BFLS91], reduce instances of circuit-SAT to various aggregated algebraic CSPs (ACSP) (cf. [BS08, Sec. 3.2] for a definition and examples). These ACSPs are designed for RM and RS codes, which are special (and simple) cases of AG codes. When working with AG codes we require a proper generalization of ACSPs to the AG setting, and we define this generalization next. The following notation will be useful for defining our algebraic CSP.

Notation 4.4: Let $C = \{f : D \to \mathbb{F}\}$, let π be an automorphism of C, and let $g : D^m \to \mathbb{F}$ be a codeword of the tensor code $C^{\otimes m}$. Then, for each $i \in [m]$, we define the function $g^{\pi,i} : D^m \to \mathbb{F}$ by $g^{\pi,i}(x_1, \ldots, x_m) = g(x_1, \ldots, x_{i-1}, \pi(x_i), x_{i+1}, \ldots, x_m)$. Moreover, if $\pi_1, \ldots, \pi_t$ are automorphisms of C, then we define the function $g^{(\pi_1,\ldots,\pi_t)} : D^m \to \mathbb{F}^{1+t\cdot m}$ to be the function obtained by aggregating the $1 + t \cdot m$ functions $g, g^{\pi_1,1}, \ldots, g^{\pi_t,m}$. Formally, $g^{(\pi_1,\ldots,\pi_t)}(\overline{x}) = \left(g(\overline{x}), g^{\pi_1,1}(\overline{x}), \ldots, g^{\pi_t,m}(\overline{x})\right)$.

Definition 4.5 (aggregate-AGCSP): An instance of the aggregate-AGCSP problem is a tuple $\psi = (m, d, t, \mathbb{F}, \vec{C}, H, \pi_1, \ldots, \pi_t, Q^{(\psi)})$ where

- m, d, t are integers
- $\vec{C} = (C_1, \ldots, C_d)$ is a multiplication code family.
- $C \stackrel{\text{def}}{=} C_1$ is a systematic linear evaluation code that encodes messages $h : H \to \mathbb{F}$ to codewords $f : D \to \mathbb{F}$.
- $\pi_1, \ldots, \pi_t$ are automorphisms of C_j for every $j \in [d]$.
- $Q^{(\psi)} : D^m \times \mathbb{F}^{1+t\cdot m} \to \mathbb{F}$ is a function that is represented by a Boolean circuit and satisfies the following property
  - For every codeword $g \in C^{\otimes m}$, it holds that $Q^{(\psi)}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$ is a codeword of $(C_d)^{\otimes m}$.

An *assignment to* ψ is a function $g : D^m \to \mathbb{F}$. Denote by $f^{(\psi,g)}$ the function

$$f^{(\psi,g)} : D^m \to \mathbb{F}, \quad f^{(\psi,g)}(\overline{x}) \stackrel{\text{def}}{=} Q^{(\psi)}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x})). \tag{2}$$

We say g *satisfies* the instance if and only if g is a codeword of $C^{\otimes m}$ for which $f^{(\psi,g)}$ vanishes on H^m, i.e., $f^{(\psi,g)}(\overline{x}) = 0$ for all $\overline{x} \in H^m$.

The problem of aggregate-AGCSP is the problem of deciding whether an instance is *satisfiable* , i.e., if it has a satisfying assignment.

The second step in our reduction is stated next. It gives a non-uniform reduction mapping an instance of Hypercube-CSP derived from Theorem 4.3 to an instance of aggregate-AGCSP.

Theorem 4.6: There exists a polynomial-time procedure which takes the following input: (1) A number $m \in \mathbb{N}$. (2) An alphabet Σ. (3) A constraint graph G over Σ whose underlying graph is a 4-regular subgraph of $\mathcal{H}_{k,m}$. (5) A finite field $\mathbb{F}$. (6) Bases for all the codes in a multiplication code family $\vec{C} = (C_1, \ldots, C_{d_{\mathsf{mult}}})$ of transitive evaluation codes $C_j = \{f : D \to \mathbb{F}\}$, where $C \stackrel{\text{def}}{=} C_1$ has message length at least $2 \cdot k$, and $d_{\mathsf{mult}} \geq |\Sigma|$. (7) For each $\alpha, \beta \in D$, a permutation π of D that (i) maps α to β, and (ii) is an automorphism of C_j for each $j \in [d]$. And outputs an instance of aggregate-AGCSP $\psi = (m, d \stackrel{\text{def}}{=} |\Sigma|, t, \mathbb{F}, \vec{C}, H, \pi_1, \ldots, \pi_t, Q^{(\psi)})$ with $|H| \leq 2k$, that is satisfiable if and only if G is satisfiable.

The next two sections provide give overviews of the two different proofs of our main theorem. Before going into those proofs, we first state the following lemma, which is used in both proofs. Let ψ, C, m, and $Q^{(\psi)}$ be as in the definition of aggregate-AGCSP, and let δ_C be the relative distance of C. Let $g : D^m \to \mathbb{F}$ be an assignment to ψ and let $f^{(\psi,g)}$ be as in the definition

of aggregate-AGCSP. We say that a point $\overline{x} \in D^m$ is *locally legal* for g if for every $i \in [m]$, it holds that $g|_{i,\overline{x}} \in C$ (i.e. g restricted to the axis-parallel line that goes through $\overline{x}$ in direction i is a codeword of C). We have the following result.

Lemma 4.7: Let $\hat{g} : D^m \to \mathbb{F}$ be a codeword of C^m that is τ-close to g for some $0 < \tau < 1$ (i.e., g and $\hat{g}$ disagree on at most τ fraction of the points in D^m). Let $\overline{x}$ be a uniformly distributed point in D^m. Then $\mathbb{P}_{\overline{x} \in D^m}\left[\overline{x} \text{ locally legal for } g \text{ and } f^{(\psi,g)}(\overline{x}) \neq f^{(\psi,\hat{g})}(\overline{x})\right] \leq m \cdot \tau/\delta_C$.

C. A proof of Main Theorem 2.3 using AG combinatorial Nullstellensatz

Applying the pair of reductions described in the previous sections (cf. (1)) converts a circuit φ of size n to an instance ψ of aggregate-AGCSP whose assignments are of length $O(n)$. Using the notation of Theorem 4.6 and Definition 4.5, we see that φ is satisfiable if and only if there exists $g \in C^{\otimes m}$ for which $f^{(\psi,g)}$ defined in (2) vanishes on H^m. The AG combinatorial Nullstellensatz (Theorem 3.14) says that this holds if and only if there exist m auxiliary functions $f'_1, \ldots, f'_m$ that "prove" that $f^{(\psi,g)}$ vanishes on H^m. The verifier thus expects to see the functions $g, f^{(\psi,g)}, f'_1, \ldots, f'_m$ and checks their internal consistency and that each of them indeed belongs to the tensor of an AG code, using Theorem 3.9. Details follow.

Proof of Theorem 2.3: We may assume $\varepsilon < 1$, otherwise the statement is trivial. Let m be the smallest integer that is strictly greater than $2/\varepsilon$. We will start by describing the verifier's operation on input φ of size n, followed by an analysis of its proof length and soundness. The full proof can be found in the full version of the paper.

Verifier's operation: The verifier V applies the reductions in (1), i.e., the reduction of Theorem 4.3 followed by the reduction of Theorem 4.6. The first reduction maps φ to an instance G of Hypercube-CSP over a subgraph of $\mathcal{H}_{k,m}$ where $k \leq 4((4 \cdot n)^{1/m} + 1)$.

For the second reduction let $d = |\Sigma| = 4$ where Σ is the alphabet stated in Theorem 4.3. We give one possible way of fixing parameters. Set $q = 2^{16}$, and note that $c_q = 2^{255}$. Set $d_{\mathsf{mult}} = 6d = 24$. Set $\delta = \frac{1}{100}$ and $\rho = \frac{1}{100 c_q}$, and note that Equation (3.14) is satisfied.

Thus we may take a transitive AG code $C = \{f : D \to \mathbb{F}_q\}$ and multiplication code family $\vec{C} = (C_1, \ldots, C_{d_{\mathsf{mult}}})$ as in Theorem 3.12, such that C has message length $200k$, blocklength $|D| \in [200k, \frac{200k}{\rho}]$, and each C_j has relative distance δ. We will be using Theorem 3.14 on this code. Note that every set $H \subseteq D$ of size $\leq 2k$ satisfies Equation (3.14), because $2k < \left(\frac{1}{100}\right) \cdot 200k \leq \left(\frac{1}{12} - \frac{\delta}{6} - \frac{2}{\sqrt{q}-1}\right) \cdot |D|$.

Now V applies Theorem 4.6 to G with the multiplication code family $\vec{C}$ (all other input parameters needed there are clear from context). For this part (and for the next) we assume that the following are hardwired into the verifier $V = V_{\varepsilon,n}$ (this is where we assume non-uniformity of the verifier): (1) A basis for each C_j, $j \in [d_{\mathsf{mult}}]$. (2) For every $\alpha, \beta \in D$, a permutation π of D that (i) is an automorphism of each C_j, $j \in [d_{\mathsf{mult}}]$, and (ii) maps α to β. (3) The pair of functions $\xi = \xi_H, \xi' = \xi'_H : D \to \mathbb{F}_q$ defined in Theorem 3.14 where $H \subset D, |H| \leq 2k$ is part of the aggregate-AGCSP instance ψ and defined in Theorem 4.6.

Denote by ψ the resulting instance of aggregate-AGCSP. As a proof oracle, the verifier V expects a total of $m+1$ functions, denoted $g, f'_1, \ldots, f'_m$. All of them have domain D^m and range $\mathbb{F}_q$. The verifier expects g to be the assignment satisfying ψ, and $f'_1, \ldots, f'_m$ should "prove" that $f^{(\psi,g)}$ (cf. (2)) vanishes on H^m as per the AG combinatorial Nullstellensatz Theorem 3.14. The verifier performs the following checks while recycling randomness.

1) **Tensor test:** Invoke the local tester of Theorem 3.9 to test that $g \in C^{\otimes m}$ and $f'_\ell \in (C_{4d})^{\otimes m}$ for each $\ell \in [m]$, using the same randomness for all the invocations.
2) **Zero test:** Choose a uniformly distributed point $\overline{x} = (x_1, \ldots, x_m) \in D^m$ and check that
   a) $f^{(\psi,g)}(\overline{x}) \cdot \prod_{r=1}^m \xi'(x_r) = \sum_{j=1}^m f'_j(\overline{x}) \cdot \xi(x_j)$, where $f^{(\psi,g)}(\overline{x})$ is computed by making $1 + m \cdot t$ queries to g.
   b) $\overline{x}$ is a locally legal for g. That is, for every direction $t \in [m]$, the axis-parallel line $g|_{t,\overline{y}}$ is a codeword of C.

If one of those checks fail, the verifier rejects, and otherwise it accepts.

Proof Length.: The proof contains $m+2$ functions with domain size $|D|^m$ and range size q. Recalling the concrete parameters from earlier on in the proof (these parameters are not necessarily optimal) $2/\varepsilon < m \leq 2/\varepsilon + 1$, $|D| \leq \frac{200k}{\rho} \leq 2^{270} \cdot (n^{1/m} + 1)$ we conclude that the proof bit-length, for sufficiently large n, is at most $(m+2) \cdot \log_2 q \cdot |D|^m < \left(\frac{2}{\varepsilon} + 3\right) \cdot 16 \cdot 2^{540/\varepsilon + 270} \cdot n \leq c_\varepsilon \cdot n$ where, asymptotically (i.e., as $\varepsilon \to 0$), $c_\varepsilon \leq 2^{c'/\varepsilon}$ for $c' < 600$.

Soundness.: Suppose φ is not satisfiable. Then by the soundness of Theorem 4.3 and Theorem 4.6 we conclude ψ is not satisfiable, i.e., there does not exist

$g \in C^{\otimes m}$ such that $f^{(\psi,g)}$, as defined in (2), vanishes on H^m. Suppose the verifier is given the proof $g, f'_1, \ldots, f'_m$. We show that the verifier rejects with probability $\Omega_m(1)$.

Let δ be the minimum of the relative distances of C_d and C_{4d} and note that δ does not depend on n. If g is $(\delta^{m+1}/2m)$-far from $(C_d)^{\otimes m}$ or any of $f'_1, \ldots, f'_m$ is $(\delta^{m+1}/2m)$-far from $(C_{4d})^{\otimes m}$, the tensor test rejects with probability at least $\gamma_m \cdot \delta^{m+1}/2m = \Omega_m(1)$ (where γ_m is as defined in Theorem 3.9). Thus, we may focus on the case in which g is $(\delta^{m+1}/2m)$-close to $(C_d)^{\otimes m}$ and all of $f'_1, \ldots, f'_m$ are $(\delta^{m+1}/2m)$-close to $(C_{4d})^{\otimes m}$. In this case, $g, f'_1, \ldots, f'_m$ are close to unique codewords $\hat{g}, \hat{f}'_1, \ldots, \hat{f}'_m$ of $(C_d)^{\otimes m}$ and $(C_{4d})^{\otimes m}$ respectively.

Observe that by the union bound, we get that with probability at least $1 - \delta^{m+1}/2$, all of $f'_1, \ldots, f'_m$ agree with $\hat{f}'_1, \ldots, \hat{f}'_m$ on $\overline{x}$ respectively. Let us focus on this case for now. Also observe that since ψ is not satisfiable, $f^{(\psi,\hat{g})}$ does not vanish on H^m.

Let $\hat{f}' = \sum_{i=1}^m \hat{f'}_i(\overline{x}) \cdot \xi(x_i)$ and $\hat{f}''(\overline{x}) = f^{(\psi,\hat{g})}(\overline{x}) \cdot \prod_{r=1}^m \xi'(x_r)$, noticing $\hat{f}', \hat{f}'' \in (C_{6d})^{\otimes m}$. Since $f^{(\psi,\hat{g})}$ does not vanish on H^m and by the AG-Nullstellensatz Theorem 3.14 we know that $\hat{f}' \neq \hat{f}''$. So by the distance property of $(C_{6d})^{\otimes m} = (C_{d_{\mathsf{mult}}})^{\otimes m}$, we conclude $\hat{f}'$ and $\hat{f}''$ differ on $\overline{x}$ with probability at least δ^m. Now, by Lemma 4.7, we get that with probability at least $1 - m \cdot \delta^{m+1}/2m \cdot \delta \geq 1 - \delta^m/2$ one of the following cases occur:

1) $\overline{x}$ is not locally legal for g, so the zero test rejects.
2) $f^{(\psi,g)}(\overline{x}) = f^{(\psi,\hat{g})}(\overline{x})$, so

$$f^{(\psi,g)}(\overline{x}) \cdot \prod_{r=1}^m \xi'(x_r) = \hat{f}''(\overline{x}) \neq \sum_{i=1}^m \hat{f'}_i(\overline{x}) \cdot \xi(x_i).$$

Since we assumed that all of $f'_1, \ldots, f'_m$ agree with $\hat{f}'_1, \ldots, \hat{f}'_m$ on $\overline{x}$, we get that

$$f^{(\psi,g)}(\overline{x}) \cdot \prod_{r=1}^m \xi'(x_r) \neq \sum_{i=1}^m f'_i(\overline{x}) \cdot \xi(x_i),$$

and therefore the zero test rejects.

We conclude that the test rejects if $\hat{f}'$ and $\hat{f}''$ differ on $\overline{x}$, all of $f'_1, \ldots, f'_m$ agree with $\hat{f}'_1, \ldots, \hat{f}'_m$ on $\overline{x}$, and either $\overline{x}$ is not locally legal for g or $f^{(\psi,g)}(\overline{x}) = f^{(\psi,\hat{g})}(\overline{x})$. This happens with probability at least $\delta^m - \delta^{m+1}/2 - \delta^m/2 = \Omega_m(1)$, as required.

This soundness analysis only ensured a rejection probability of $\Omega(\delta^m)$. If we want to make the rejection probability in the NO case be $\geq 1/2$, then we would have to repeat the verifier's operation (while recycling randomness via expander walks) $O(\frac{1}{\delta^m})$ times.

■

ACKNOWLEDGMENT

This work was partially done at the Oberwolfach Workshop on Complexity Theory, 2012. We are grateful to the organizers for the chance to attend the workshop, and to the center for the wonderful hospitality.

REFERENCES

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy, *Proof verification and intractability of approximation problems*, Journal of ACM **45** (1998), no. 3, 501–555.

[Alo99] Noga Alon, *Combinatorial nullstellensatz*, Combinatorics Probability and Computing **8** (1999), no. 1, 7–30.

[Aro02] Sanjeev Arora, *How NP got a new definition: a survey of probabilistically checkable proofs*, ICM '02: Proceedings of the 2002 International Congress of Mathematicians, vol. 3, 2002, pp. 637–648.

[AS98] Sanjeev Arora and Shmuel Safra, *Probabilistic checkable proofs: A new characterization of NP*, Journal of ACM volume **45** (1998), no. 1, 70–122.

[BCGT13a] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer, *Fast reductions from RAMs to delegatable succinct constraint satisfaction problems*, ITCS 2013.

[BCGT13b] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer,, *On the concrete efficiency of probabilistically-checkable proofs*, STOC 2013.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy, *Checking computations in polylogarithmic time*, STOC, 1991, pp. 21–31.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan, *Short PCPs verifiable in polylogarithmic time*, Proceedings of CCC, IEEE Computer Society, 2005, pp. 120–134.

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan, *Robust PCPs of proximity, shorter PCPs and applications to coding*, SIAM Journal of Computing **36** (2006), no. 4, 120–134.

[BS06] Eli Ben-Sasson and Madhu Sudan, *Robust locally testable codes and products of codes*, Random Struct. Algorithms **28** (2006), no. 4, 387–402.

[BS08] Eli Ben-Sasson and Madhu Sudan, *Short PCPs with polylog query complexity*, SIAM J. Comput. **38** (2008), no. 2, 551–607.

[BSGK+13] Eli Ben-Sasson, Ariel Gabizon, Yohay Kaplan, Swastik Kopparty, and Shubhangi Saraf, *A new family of locally correctable codes based on degree-lifted algebraic geometry codes*, STOC, 2013.

[BSV09] Eli Ben-Sasson and Michael Viderman, *Tensor products of weakly smooth codes are robust*, Theory of Computing **5** (2009), no. 1, 239–255.

[BSVW03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson, *Randomness-efficient low degree tests and short pcps via epsilon-biased sets*, Proceedings of STOC '03, 2003, pp. 612–621.

[BV09] Eli Ben-Sasson and Michael Viderman, *Composition of semi-ltcs by two-wise tensor products*, APPROX-RANDOM 2009.

[Cam98] Peter J. Cameron, *Combinatorics: Topics, techniques, algorithms*, Cambridge University Press, Cambridge CB2 2RU, MA, USA, 1998.

[CR05] Don Coppersmith and Atri Rudra, *On the robust testability of tensor products of codes*, (ECCC) (2005), no. 104.

[Din07] Irit Dinur, *The PCP theorem by gap amplification*, Journal of ACM **54** (2007), no. 3, 241–250.

[DM11] Irit Dinur and Or Meir, *Derandomized parallel repetition via structured PCPs*, Computational Complexity **20** (2011), no. 2, 207–327.

[DR06] Irit Dinur and Omer Reingold, *Assignment testers: Towards combinatorial proof of the PCP theorem*, SIAM Journal of Computing **36** (2006), no. 4, 155–164.

[DSW06] Irit Dinur, Madhu Sudan, and Avi Wigderson, *Robust local testability of tensor products of ldpc codes*, APPROX-RANDOM, 2006, pp. 304–315.

[GM12] Oded Goldreich and Or Meir, *The tensor product of two good codes is not necessarily locally testable*, Inf. Proces. Lett. **112** (2012), no. 8-9, 351–355.

[GO05] Venkatesan Guruswami and Ryan O'Donnell, *The PCP theorem and hardness of approximation*, 2005, Available online at http://www.cs.washington.edu/education/courses/533/05au/.

[GS96] Arnaldo Garcia and Henning Stichtenoth, *On the Asymptotic Behaviour of Some Towers of Function Fields over Finite Fields*, Journal of Number Theory **61** (1996), 248–273.

[GS06] Oded Goldreich and Madhu Sudan, *Locally testable codes and pcps of almost-linear length*, J. ACM **53** (2006), no. 4, 558–655.

[GV87] Arnaldo Garcia and JF Voloch, *Wronskians and linear independence in fields of prime characteristic*, manuscripta mathematica **59** (1987), no. 4, 457–469.

[HS00] Prahladh Harsha and Madhu Sudan, *Small PCPs with low query complexity*, Computational Complexity **9** (2000), no. 3–4, 157–201.

[KR12] Géza Kós and Lajos Rónyai, *Alon's nullstellensatz for multisets*, Combinatorica **32** (2012), no. 5, 589–605.

[KS08] Tali Kaufman and Madhu Sudan, *Algebraic property testing: the role of invariance*, STOC (Cynthia Dwork, ed.), ACM, 2008, pp. 403–412.

[KSY11] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin, *High-rate codes with sublinear-time decoding*, STOC, 2011, pp. 167–176.

[Lei92] F. Thomson Leighton, *Introduction to parallel algorithms and architectures: array, trees, hypercubes*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Nisan Noam, *Algebraic methods for interactive proof systems*, Journal of the ACM **39** (1992), no. 4, 859–868.

[Mei10] Or Meir, *IP = PSPACE using error correcting codes*,(ECCC) (2010), no. 137.

[Mei12a] Or Meir, *Combinatorial PCPs with short proofs*, CCC 2012, pp. 345–355.

[Mei12b] Or Meir, *On the rectangle method in proofs of robustness of tensor products*, Inf. Process. Lett. **112** (2012), no. 6, 257–260.

[Mie09] Thilo Mie, *Short PCPPs verifiable in polylogarithmic time with o(1) queries*, Annals of Mathematics and Artificial Intelligence **56** (2009), 313–338.

[MS88] Florence Jessie MacWilliams and Neil James Alexander Sloane, *The theory of error correcting codes*, Elsevier/North-Holland, Amsterdam, 1988.

[PF79] Nicholas Pippenger and Michael J. Fischer, *Relations among complexity measures*, J. ACM **26** (1979), no. 2, 361–381.

[PS94] Alexander Polishchuk and Daniel A. Spielman, *Nearly-linear size holographic proofs*, STOC, 1994, pp. 194–203.

[Sti93] Henning Stichtenoth, *Algebraic function fields and codes*, Universitext, Springer, 1993.

[Sti06] Henning Stichtenoth, *Transitive and self-dual codes attaining the Tsfasman-Vlăduţ-Zink bound*, IEEE Transactions on Information Theory **52** (2006), no. 5, 2218–2224.

[Sud01] Madhu Sudan, *Algorithmic introduction to coding theory (lecture notes)*, 2001.

[Sud10] Madhu Sudan, *Invariance in property testing*, Property Testing (Oded Goldreich, ed.), Lecture Notes in Computer Science, vol. 6390, Springer, 2010, pp. 211–227.

[TVZ82] M.A. Tsfasman, S.G. Vladut, and T. Zink, *Modular curves, shimura curves, and goppa codes, better than varshamov-gilbert bound*, math. Nachr. **109** (1982), 21–28.

[Val05] Paul Valiant, *The tensor product of two codes is not necessarily robustly testable*, APPROX-RANDOM, 2005, pp. 472–481.

[Vid12] Michael Viderman, *A combination of testability and decodability by tensor products*, APPROX-RANDOM, 2012, pp. 651–662.