Constant rate PCPs for circuit-SAT with sublinear query complexity

Eli Ben-Sasson[*] Yohay Kaplan[*] Swastik Kopparty[†] Or Meir[‡]

June 12, 2013

With an Appendix by Henning Stichtenoth [1]

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 length $n \cdot \operatorname{poly}\log n$ that can be verified using a constant number of queries. In this work, we show that if the query complexity is relaxed to n^{ε}, then one can construct PCPs of length $O(n)$ for circuit-SAT, and PCPs of length $O(t \log t)$ for any language in NTIME(t).

More specifically, for any $\varepsilon > 0$ we present (non-uniform) probabilistically checkable proofs (PCPs) of length $2^{O(1/\varepsilon)} \cdot n$ that can be checked using n^{ε} queries for circuit-SAT instances of size n. Our PCPs have perfect completeness and constant soundness. This is the first constant-rate PCP construction that achieves constant soundness with nontrivial query complexity ($o(n)$).

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 in the appendix to this paper for the first time for every message length, after they have been constructed for infinitely many message lengths by Stichtenoth [Trans. Information Theory 2006].

[*]Department of Computer Science, Technion, Haifa, Israel. Work done while visiting MIT, Cambridge, MA. eli/yohayk@cs.technion.ac.il. 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.

[†]Department of Mathematics & Department of Computer Science, Rutgers University. swastik.kopparty@rutgers.edu. This material is based upon work supported by the National Science Foundation under Grant No. NSF CCF-1253886.

[‡]Institute for Advanced Study, Princeton, NJ. ormeir@ias.edu. Partially supported by NSF grant CCF-0832797.

[1]Sabanci University, MDBF, Istanbul, Turkey. henning@sabanciuniv.edu. This work was partially supported by Tubitak, proj. no. 111T234

Contents

1 Introduction

The PCP theorem [AS98, ALM+98] is one of the major achievements of complexity theory. A PCP (Probabilistically Checkable Proof) 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] and culminated in the works of Ben-Sasson, Sudan, and Dinur which construct PCP verifiers that make $O(1)$ queries to proofs of length only $t(n) \cdot \mathrm{poly}\log t(n)$ [BS08, Din07]. It is an interesting open question whether one can improve on the latter construction and obtain PCPs of length $O(t(n))$, or even $t(n) \cdot \mathrm{poly}\log t(n)$ with smaller degree in the polynomial.

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 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 $\mathsf{circuit\text{-}SAT}$, 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 $\mathsf{circuit\text{-}SAT}$ 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 $L \in \mathsf{NTIME}(t)$.

Theorem 1.1 (Main — Informal) *There exists a constant $c > 0$ such that the following holds for every $\varepsilon > 0$ and integer n. There exists a randomized oracle circuit C_n of size* $\mathrm{poly}(n)$ *that when given as input the description of a circuit φ of size n, acts as follows:*

1. *C_n makes at most n^{ε} queries to its oracle.*
2. *If φ is satisfiable, then there exists a string π_{φ} of length $2^{c/\varepsilon} \cdot n$ such that $C_n^{\pi_{\varphi}}(\varphi)$ accepts with probability 1.*
3. *If φ is not satisfiable, then for every string π, it holds that $C_n^{\pi}(\varphi)$ rejects with probability at least $\frac{1}{2}$.*

Again, by combining Theorem 1.1 with the reduction of [PF79], we obtain the following result.

Corollary 1.2 *For every $\varepsilon > 0$ and time-constructible function $t : \mathbb{N} \to \mathbb{N}$ the following holds for every language $L \in \mathsf{NTIME}(t)$. For every input length n, there exists a randomized oracle circuit C_n of size* $\operatorname{poly}(t)$ *that when given as input $x \in \{0,1\}^n$, acts as follows:*

1. *C_n makes at most $(t(n))^{\varepsilon}$ queries to its oracle.*
2. *If $x \in L$, then there exists a string π_x of length $O_{\varepsilon}(t(n)\log t(n))$ such that $C_n^{\pi_x}(x)$ accepts with probability* 1.
3. *If $x \notin L$, then for every string π, it holds that $C_n^{\pi}(x)$ rejects with probability at least $\frac{1}{2}$.*

Remark 1.3 *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. See also remark A.19.*

1.1 Our techniques

In order to explain the new ideas that we employ to get PCPs of linear length for circuit-SAT, 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 five main 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. I.e., the binary string of length m_1 is encoded via the Reed-Solomon (RS) code of degree 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 $m_2 \geq m_1 \log m_1$. This loses another logarithmic factor in the length of the PCPs, i.e., $m_2 \geq n \log^2 n$.

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 $Q \circ P$, 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 $R \stackrel{\text{def}}{=} Q \circ P$ 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 $Q \circ P$ are low-degree polynomials, or are at least close to low-degree polynomials[2]. Thus, in order to verify that the ACSP instance is satisfiable, we need to be able to very that a function is close to a low-degree polynomial. This is done in [BS08] by constructing a PCP of proximity (PCPP) [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, i.e., denoting the length of the PCP by m_3 we have $m_3 \approx m_2 \geq n \log^2 n$. We elaborate later more on this step.

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 $m = m_3 \cdot \text{poly} \log m_3 = n \cdot \text{poly} \log n$.

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 (note that this is indeed a generalization, as affine transformations are automorphisms of RS-codes).

- For Step 2, we reduce the size of the finite field used in the algebraic CSP to a constant, which prevents the logarithmic blowup incurred when moving from m_1 to m_2. This is done by replacing the Reed-Solomon 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 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 algebraic characterization of multi-variate polynomials vanishing on H^m (a.k.a. the "Combinatorial Nullstellensatz" of [Alo99]) to tensor products of AG-codes. This yields a new "AG Combinatorial Nullstellensatz" which we prove in Section 7. We discuss this point in some more detail in the next sub-section.

  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

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

the AG codes themselves. We then use the fact that tensor codes are locally testable[3] as an analog of low-degree testing.

We mention that we use the tensor codes for another reason in addition to their local testability. Specifically, we use the fact that tensor codes support the sum-check protocol (see [Mei10]), which is used in our second PCP construction.

- 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. This seems like a very interesting question.

1.2 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 case[4] 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 $\mathbb{F}_q$ (see Equation (25) in the Appendix for an example)

$$D = \left\{\overline{x} = (x_1, \ldots, x_m) \in \mathbb{F}_q^m \mid e_1(\overline{x}) = \ldots = e_k(\overline{x}) = 0\right\}$$

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 $\mathbb{F}_q$ and 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.

Why AG codes? A key property of Reed-Solomon 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 $(f \cdot g)(x) \overset{\text{def}}{=} f(x) \cdot g(x), x \in \mathbb{F}$. 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,

[3]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 3.10)

[4]Formally, RS codes are AG codes but are usually not referred to as such in the computational complexity literature. Given that RS codes have genus 0, a more precise statement would be that our construction is the first that utilizes AG codes of positive genus in PCP constructions.

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 [BGK+13].

The Appendix: Dense transitive AG codes A family of codes $\mathcal{F} = \{C_i\}_{i\in\mathbb{N}}$ that has constant rate, constant relative distance and constant alphabet-size is called "asymptotically good". The only previously-known asymptotically good family of transitive AG codes appeared in [Sti06]. The family described there is "sparse": Assuming C_i has blocklength n_i and $n_1 < n_2 < \ldots$, the ratio n_{i+1}/n_i of that family is super-constant,

$$\frac{n_{i+1}}{n_i} \xrightarrow{i\to\infty} \infty.$$

Consequently, using that family we would only be able to obtain an "infinitely-often" type of result: For infinitely often circuit-sizes $k_1 < k_2 < \ldots$, circuits of size k_i have constant-rate PCPs. The new asymptotically good family of transitive AG codes presented by Stichtenoth in the Appendix is "dense", i.e., n_{i+1}/n_i is at most an absolute constant c. This dense family allows us to extend our main result to all circuit-sizes.

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.

The AG Combinatorial Nullstellensatz. As mentioned above, the work of [BS08] solved the zero-testing problem by using an algebraic characterization of polynomials that vanish on a set H. More specifically, it used that fact that a univariate polynomial $p(X)$ vanishes at each $\alpha \in H$ if and only if $\prod_{\alpha\in H}(X-\alpha)$ divides $p(X)$. As shown in [BS08, Lemma 4.9], Alon's Combinatorial Nullstellensatz [Alo99] gives an extension of this characterization to to multi-variate polynomials that vanish on a set H^m. This characterization (cf. (3)) can be used to solve the zero-testing problem multi-variate polynomials.

Our first method for solving the zero-testing problem in the AG settings uses a similar algebraic characterization of H^m-vanishing functions, but now f is not a low-degree polynomial. Rather, it belongs to the tensor product of "low-degree" AG codes, and H is a subset of the point-set D. Theorem 7.4 contains what we view as the natural generalization of the Combinatorial Nullstellensatz, and we hope it will find further applications. To prove it we need to overcome a number of nontrivial technical challenges that arise only over curves of positive genus (i.e., only over algebraic codes that are not RS/RM). One such problem appears even in the simplest case, the univariate one (when $m = 1$): Some point-sets $H \subset D$ cannot be characterized as the roots of a degree-$|H|$ function. This contrasts with the RS-code where every $H \subset \mathbb{F}$ is the set of roots of the polynomial $\prod_{\alpha \in H}(X - \alpha)$. See Section 7 for more details.

1.3 Open problems

PCPs with constant rate and query complexity. The most obvious open question that arises from our work is "what is the smallest possible query complexity for a constant-rate PCP?". In particular, do constant-rate PCPs with constant query complexity exist?

A smooth trade-off. A perhaps less ambitious goal would be to try to obtain a smooth trade-off between the existing PCP constructions. Currently, we have a PCP construction that obtains constant query complexity and length of $n \cdot \mathrm{poly}\log n$, and our construction gives query complexity of n^{ε} and length $O(n)$. Is it possible to obtain a smooth trade-off between the query complexity and the length? As a concrete conjecture, is it possible to construct, for every function q, a PCP with query complexity q and length $n \cdot \mathit{poly} \log_q n$?

Better decision complexity. The most straightforward way to obtain a trade-off as in the last paragraph would be to apply composition to our PCPs. Unfortunately, our PCPs do not compose efficiently. The main obstacle is that the decision complexity of our PCPs (defined in Section 2 below) is too large - in particular, it is polynomial, rather than linear, in the query complexity. Improving the decision complexity of our PCPs is another open question that arises from our work.

We note that the large decision complexity results from the fact that we do not know how to verify the membership of a codeword in an AG code in linear time. In fact, even the fastest algorithms for verifying membership in a Reed-Solomon code run in time $n \log n$, which is not sufficiently efficient for our purposes.

1.4 The road ahead

In the next section we formally state our main results. Section 3 states the required preliminaries about error correcting codes: tensor codes and their testability, the 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 4 gives the proof of Main Theorem 1.1. It does so in a succinct manner, leaving the proofs of various reductions to later sections (Sections 5–7). Of particular importance is Section 7 where the AG combinatorial Nullstellensatz is proved. Finally, the Appendix by Henning Stichtenoth constructs the asymptotically good "dense" family of transitive AG codes needed for our result.

2 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 *The* circuit satisfiability problem*, circuit-SAT, is the problem of deciding whether a circuit is satisfiable. Formally,*

$$\textsf{circuit-SAT} \stackrel{\text{def}}{=} \{\varphi : \varphi \textit{ is a satisfiable circuit}\}\,.$$

Recall that a PCP verifier for a language L is an algorithm that verifies a claim of the form $w \in L$ by querying few bits from an auxiliary proof π. It is common to define a PCP verifier as an oracle machine that is given oracle access to the proof π and is allowed to make only few queries to this oracle. In this work, we will use a slightly different definition of PCPs, taken from the PCP literature (e.g., [BGH+06]), which allows keeping track of some additional important parameters of the PCP, namely, the randomness complexity and the decision complexity of the PCP.

The *randomness complexity* of a PCP verifier is just the number of coin tosses the verifier uses. The *decision complexity* is the complexity of the predicates that the verifier applies to the answers it gets from its oracle: More specifically, in the definition of PCPs, we view the verifier as machine that outputs its queries (as a list of coordinates), and a predicate (represented as a circuit) that should be applied to the answers given to those queries. We view the verifier as accepting if the predicate accepts the answers to the queries, and otherwise we view the verifier as rejecting. The *decision complexity* of the PCP is the complexity of the aforementioned predicate. The randomness complexity and decision complexity of a PCP are usually used in the PCP literature in order to facilitate the composition of PCPs. While in this work we do not use composition, we still keep track of those parameters since they might be of use for future works.

Definition 2.2 (Non-uniform PCP verifier, following [BGH+06]) *Let $L \subseteq \{0,1\}^*$ be a language, and let $r, q, \ell, d, v : \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \to (0,1)$. A* (non-uniform) PCP verifier $V = \{V_n\}_{n=1}^{\infty}$ *for L with* query complexity q, proof length ℓ, rejection probability ρ, randomness complexity r, decision complexity d, *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 $|I| \le q(n)$, and a circuit $\psi : \{0,1\}^I \to \{0,1\}$ of size at most $d(n)$. 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. ***Randomness complexity:*** *On every input w, and on every sequence of coin tosses, V_n tosses at most $r(n)$ coins.*
5. ***Completeness:*** *For every $w \in L$, there exists a string $\pi \in \{0,1\}^{\ell(n)}$ such that*

$$\mathbb{P}\left[\psi\left(\pi|_I\right) = 1\right] = 1,$$

where the probability is over ψ and I generated by the verifier V on input w.

6. ***Soundness:*** *For every string $w \notin L$ and every string $\pi \in \{0,1\}^{\ell(n)}$, it holds that*

$$\mathbb{P}\left[\psi\left(\pi|_I\right) = 0\right] \geq \rho(n),$$

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 $\textsf{circuit-SAT}_n$ with query complexity $c_\varepsilon \cdot n^\varepsilon$, proof length $c_\varepsilon \cdot n$, rejection probability $1/2$, verifier complexity $(c_\varepsilon \cdot n)^{O(1)}$, randomness complexity $\log n + c_\varepsilon$, and decision complexity $c_\varepsilon \cdot n^\varepsilon$.*

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

$$\textsf{circuit-SAT} \;\; {}^{(i)}\xrightarrow{\text{Theorem 4.5}} \;\; \textsf{Hypercube-CSP} \;\; {}^{(ii)}\xrightarrow{\text{Theorem 4.9}} \;\; \textsf{aggregate-AGCSP} \begin{cases} {}^{(iiia)}\xrightarrow{\text{Section 4.3}} & \text{Nullstellensatz} \\ {}^{(iiib)}\xrightarrow{\text{Section 4.4}} & \text{Sum-check} \end{cases} \tag{1}$$

(i) The first reduction maps an instance φ of $\textsf{circuit-SAT}$ to a graph constraint satisfaction problem over a sub-graph of the hypercube. This reduction is stated in Section 4.1 and proved in Section 5. *(ii)* The next reduction maps the hypercube constraint satisfaction problem into an Algebraic Geometry Constraint Satisfaction Problem (AGCSP). This part is stated in Section 4.2 and proved in Section 6. 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 3.16) 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. The Nullstellensatz proof appears in Section 4.3 and the sum-check proof appears in Section 4.4.

3 Tensors of AG Codes and an AG Combinatorial Nullstellensatz

This section starts by reviewing preliminaries of error correcting codes. It then reviews the basic properties of tensor product codes as well as their local testability. We conclude by describing the AG codes that we use and our AG Combinatorial Nullstellensatz which pertains to tensors of AG codes.

3.1 Error Correcting Codes

For any $n \in \mathbb{N}$, we denote $[n] \overset{\text{def}}{=} \{1, \ldots, n\}$. For any two strings x, y of equal length n and over any alphabet, the *relative Hamming distance* between x and y is the fraction of coordinates on which x and y differ, and is denoted by $\delta(x,y) = |\{i \in [n] : x_i \neq y_i\}|/n$. Also, if S is a set of strings of length n, we denote by $\delta(x,S)$ the distance of x to the closest string in S.

All the error-correcting codes that we consider in this paper are *linear codes*, to be defined next. Let $\mathbb{F}$ be a finite field, and let $k, \ell \in \mathbb{N}$. A *(linear) code* C is a linear one-to-one function from $\mathbb{F}^k$ to $\mathbb{F}^\ell$, where k and ℓ are called the code's *message length* and *block length*, respectively. We will sometimes identify C with its image $C(\mathbb{F}^k)$. Specifically, we will write $c \in C$ to indicate the fact that there exists $x \in \mathbb{F}^k$ such that $c = C(x)$. In such case, we also say that c is a *codeword* of C.

The *relative distance* of a code C is the minimal relative Hamming distance between two distinct codewords of C, and is denoted by $\delta_C = \min_{c_1 \neq c_2 \in C} \{\delta(c_1, c_2)\}$. The *rate* of the code C is the ratio k/ℓ. If $d \overset{\text{def}}{=} \delta_C \cdot \ell$, we also say that C is a $[\ell, k, d]_{\mathbb{F}}$-code.

Due to the linearity of C, there exists an $\ell \times k$ matrix G, called a *generator matrix* of C, such that for every $x \in \mathbb{F}^k$ it holds that $C(x) = G \cdot x$. Observe that given a generator matrix of C one can encode messages by C as well as verify that a string in $\mathbb{F}^\ell$ is a codeword of C in time that is polynomial in ℓ. Moreover, observe that the code C always encodes the all-zeros vector in $\mathbb{F}^k$ to the all-zeros vector in $\mathbb{F}^\ell$.

We say that C is *systematic* if the first k symbols of a codeword contain the encoded message, that is, if for every $x \in \mathbb{F}^k$ it holds that $(C(x))|_{[k]} = x$. By applying a change of basis (which can be implemented using Gaussian elimination), we may assume, without loss of generality, that C is systematic.

Evaluation codes. Throughout the paper, it will often be convenient to think of the codewords of a code as functions rather than strings. Specifically, we will usually identify the codewords of an $[\ell, k, d]_{\mathbb{F}}$-code $C : \mathbb{F}^k \to \mathbb{F}^\ell$ with some k-dimensional space L of functions $C = \{f : D \to \mathbb{F} \mid f \in L\}$ where D is some set of size ℓ. When C is an AG code, as will soon be the case, D will be a set of points on a curve and L will be a Riemann-Roch space of a divisor on the curve. We will refer to codes that are viewed in this way as *evaluation codes*. We will say that an evaluation code is *systematic* if its messages can be viewed as functions $h : H \to \mathbb{F}$ for some $H \subseteq D$, such that the encoding f of a message h satisfies $f|_H = h$.

3.2 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 notation such as $C \otimes C \otimes C$, and more generally:

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 $\underbrace{C \otimes C \otimes \ldots \otimes C}_{m}$. Formally, $C^{\otimes m} = C^{\otimes m-1} \otimes C$. 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*

$$g|_{i,v_{-i}}(x) = g(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_m), \quad x \in D.$$

Using Fact 3.2, one can prove by induction the following.

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 $1 \le i \le m$ 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 $1 \le t \le m$ 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 $x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_m$).

Remark 3.8 *The "moreover" part in Claim 3.7 does not appear in [Mei10], but is implicit in the proof there.*

Claim 3.9 ([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 $2 \le t \le m$ 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. In particular, we use the following result of [Vid12].

Theorem 3.10 (Testing of tensor codes) *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 $\gamma_m = \delta_C^{3m}/\mathrm{poly}(m)$.*

3.3 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 and hence postpone the definitions and proofs to Section 7. 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 3.11 (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 3.12 (Codes with a transitive automorphism group) *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 $\mathsf{circuit\text{-}SAT}_n$ has constant-rate PCPs with sublinear query complexity. The main result presented in the appendix to this paper (Theorem A.14) 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 3.13 (Asymptotically good transitive AG-codes for every message length (Theorem A.14)) *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*

$$c_q \cdot d_{\mathsf{mult}} \cdot \rho + \delta < 1 - \frac{d_{\mathsf{mult}}}{\sqrt{q}-1} \tag{2}$$

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 $C_1 = C$, satisfying:

Basic *parameters.* *$C = \{f : D \to \mathbb{F}_q\}$ is a linear evaluation code over $\mathbb{F}_q$ of dimension at least k, relative distance at least δ, and with length $|D| \leq \frac{1}{\rho} \cdot k$. Succinctly, C is an $[|D| \leq \frac{k}{\rho}, \geq k, \geq \delta|D|]_q$-code.*

Transitivity. *All the codes C_j are jointly transitive: i.e., for every $\alpha, \beta \in D$ there exists an automorphism $\pi \in \bigcap_{j=1}^{d_{\mathsf{mult}}} \mathrm{Aut}(C_j)$ such that $\pi(\alpha) = \beta$.*

Distance *of Multiplication codes.* *For each $j \leq d_{\mathsf{mult}}$, the code C_j has relative distance at least δ.*

Remark 3.14 *The appendix states the above result a bit differently. More specifically, in the appendix, the codes are not defined for every message length, but only for an infinite sequence $k_1, k_2, \ldots$ of message lengths that satisfies $k_{i+1} \leq c_q \cdot k_i$ for every i. On the other hand, the appendix states a better trade-off between ρ, δ, d, for those message lengths, namely,*

$$d \cdot \rho + \delta < 1 - \frac{1}{\sqrt{q}-1}.$$

The parameters in Theorem 3.13 are weaker because we are asking for codes for every message length k. For the "missing message lengths", we use a code corresponding to the smallest k_i larger than k.

3.4 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. (g actually belongs to $(C^d)^{\otimes m}$ but C^d is essentially an AG code too.) Our goal is to test whether g vanishes on a set H^m, i.e., whether $g(x) = 0$ for all $x \in H^m$. 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 3.15 (Combinatorial Nullstellensatz [Alo99]) *For $H \subset \mathbb{F}_q$ let $\xi_H(Y) = \prod_{\alpha \in H}(Y - \alpha)$ be the nonzero monic polynomial of degree $|H|$ that vanishes on H. Let $f(X_1, \ldots, X_m)$ be a polynomial over $\mathbb{F}_q$ of individual degree at most d. Then $f(X_1, \ldots, X_m)$ 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*

$$f(X_1, \ldots, X_m) = \sum_{i=1}^{m} f'_i(X_1, \ldots, X_m) \cdot \xi_H(X_i). \tag{3}$$

The importance of this theorem to PCP constructions is that it reduces the problem of testing many constraints — each constraint is of the form $f(x_1, \dots, x_m) = 0$ and there are $|H|^m$ of them — to that of checking that each of $f'_1, \dots, f'_m$ are low-degree polynomials along with a consistency check that indeed, $f = \sum_{i=1}^{m} f'_i \cdot \xi_H(X_i)$.

In Section 7 we shall properly define and prove the AG combinatorial Nullstellensatz. Next we state a special case of it using the minimal AG formalities and tailored for the purpose of proving Theorem 2.3. Comparing it to the previous theorem one main difference is that we require *two* auxiliary functions ξ, ξ' as opposed to only one above. The full-generality result is stated in Theorem 7.4.

Theorem 3.16 (Special case of AG Combinatorial Nullstellensatz) *Let $C = \{f : D \to \mathbb{F}\}$ and $\vec{C} = (C_1, \dots, C_{d_{\mathsf{mult}}})$ be the codes from Theorem 3.13 with multiplication degree $d_{\mathsf{mult}} = 6d$, rate parameter ρ, and distance parameter δ satisfying*

$$c_q \cdot d_{\mathsf{mult}} \cdot \rho + \delta < \frac{1}{2} - \frac{d_{\mathsf{mult}}}{\sqrt{q} - 1}. \tag{4}$$

Then for every $H \subset D$ with

$$|H| < \left(\frac{1}{12} - \frac{\delta}{6} - \frac{2}{\sqrt{q} - 1} \right) \cdot |D|, \tag{5}$$

there exist $\xi(X) = \xi_H(X) \in C_{2d}$ and $\xi'(X) = \xi'_H(X) \in C_{3d}$ satisfying the following. Suppose $f(X_1, \dots, X_m) \in (C_d)^{\otimes m}$. Then f vanishes on H^m if and only if there exist $f'_1, \dots, f'_m : D^m \to \mathbb{F}_q$, with $f'_i \in (C_{4d})^{\otimes m}$ for each $i \in [m]$, such that:

$$f(X_1, \dots, X_m) \cdot \prod_{i=1}^{m} \xi'(X_i) = \sum_{i=1}^{m} f'_i(X_1, \dots, X_m) \cdot \xi(X_i). \tag{6}$$

In Section 7, we show how to instantiate parameters in Stichtenoth's code construction and in our AG Combinatorial Nullstellensatz to get the precise statements of Theorem 3.13 and Theorem 3.16.

4 Proof of Main Theorem 2.3

4.1 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 4.5). The proof of this theorem is deferred to Section 5. It follows similar reductions that were used in previous works in the PCP literature starting from [BFLS91, PS94], which were in turn based on routing techniques (see, e.g., [Lei92]). We start by defining constraint satisfaction problems on graphs and on the hypercube graph formally.

Definition 4.1 (Constraint graph) *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.

Definition 4.2 (Graph CSP) *The* graph constraint satisfaction problem*, graph-CSP, is the problem of deciding whether a constraint graph G is satisfiable. Formally,*

$$\textsf{graph-CSP} \stackrel{\text{def}}{=} \{G : G \text{ is a satisfiable constraint graph}\}.$$

Definition 4.3 (The hypercube graph) *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. In other words, u and v are connected by an edge if and only if there exists $i \in [d]$ such that $u_j = v_j$ for all $j \neq i$.*

Definition 4.4 (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,*

$$\textsf{Hypercube-CSP} \stackrel{\text{def}}{=} \left\{ G : G \in \textsf{graph-CSP} \text{ and } G \leq H \text{ for some } H \in \bigcup_{k,m} \mathcal{H}_{k,m} \right\}.$$

We now state the first step in our reduction, its proof appears in Section 5.

Theorem 4.5 (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 $k \leq 4((4 \cdot n)^{1/m} + 1)$.*

4.2 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 4.5. 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 Reed-Muller and Reed-Solomon 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 4.6 (Axis-parallel composition with automorphism) *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 4.7 (Aggregated Algebraic Geometry CSP (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.*
- *$C \overset{\text{def}}{=} C_1$ 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*

$$f^{(\psi,g)} : D^m \to \mathbb{F}, \quad f^{(\psi,g)}(\overline{x}) \overset{\text{def}}{=} Q^{(\psi)}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x})). \tag{7}$$

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.*

Remark 4.8 (Non-aggregated AGCSP) *A constraint satisfaction problem is defined as a* set *of constraints whereas in the definition above (as well as in all previous definitions of algebraic CSPs) the constraint-set is "captured" by a single object. In our case this object is the function $Q^{(\psi)}$. The reduction of Hypercube-CSP to aggregate-AGCSP stated next and proved in Section 6 will clarify that $Q^{(\psi)}$ really is an* aggregate *of a large set of constraints (as was done in previous ACSPs).*

The second step in our reduction is stated next. It gives a non-uniform reduction mapping an instance of Hypercube-CSP derived from Theorem 4.5 to an instance of aggregate-AGCSP. The proof appears in Section 6.

Theorem 4.9 *There exists a polynomial-time procedure with the following input-output behavior:*

- ***Input:***

  - *A number $m \in \mathbb{N}$.*
  - *An alphabet Σ.*
  - *A constraint graph G over Σ whose underlying graph is a 4-regular subgraph of $\mathcal{H}_{k,m}$.*
  - *A finite field $\mathbb{F}$.*
  - *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 $d_{\mathsf{mult}} \geq |\Sigma|$.*
  - *For each $\alpha, \beta \in D$, a permutation π of D that (1) maps α to β, and (2) is an automorphism of C_j for each $j \in [d]$.*

- ***Output:*** *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.

Remark 4.10 *Note that in Theorem 4.9, the assignments to G are of length $O(k^m \cdot \log|\Sigma|)$ and the assignments to ψ are of length $|D|^m \cdot \log|\mathbb{F}|$. In our settings, we will have $|\Sigma| = O(1)$, $|\mathbb{F}| = O(1)$, $m = O(1)$ and $|D| = O(k)$, so the reduction will preserve the length of the assignments up to a constant factor.*

Remark 4.11 *Note that the fact that the procedure in Theorem 4.9 runs in polynomial time implies in particular that the length of ψ is polynomial in the length of the inputs to the procedure. This in particular implies that the size of the circuit computing $Q^{(\psi)}$ is polynomial in the length of the inputs to the procedure.*

The next two sections provide two different proofs of our main theorem, one based on the AG combinatorial Nullstellensatz and one based on the sum-check protocol. Before going into those proofs, we first prove the following lemma, which is used in both proofs. Intuitively, the lemma says: Suppose we have an assignment g to ψ that is not a codeword of $C^{\otimes m}$, but is close to some codeword $\hat{g}$. We would have liked to argue that in such a case $f^{(\psi,g)}$ and $f^{(\psi,\hat{g})}$ are close to each other. While this is not necessarily true, the lemma says that $f^{(\psi,g)}$ and $f^{(\psi,\hat{g})}$ agree on most points that are "locally legal", which is a condition that can be checked by the verifier using relatively few queries.

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 4.12 *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*

$$\mathbb{P}_{\overline{x} \in D^m}\left[\overline{x} \text{ is locally legal for } g \text{ and } f^{(\psi,g)}(\overline{x}) \neq f^{(\psi,\hat{g})}(\overline{x})\right] \leq m \cdot \tau / \delta_C. \tag{8}$$

Proof: We begin by fixing some point $\overline{x}$ that is both locally legal and satisfies $f^{(\psi,g)}(\overline{x}) \neq f^{(\psi,\hat{g})}(\overline{x})$. Recall that $f^{(\psi,g)}$ is computed by evaluating g on all the axis-parallel lines that pass through $\overline{x}$, and the same goes for $f^{(\psi,\hat{g})}$ and $\hat{g}$. Thus, the assumption that $f^{(\psi,g)}(\overline{x}) \neq f^{(\psi,\hat{g})}(\overline{x})$ implies that there exists some $i \in [m]$ such that $g|_{i,\overline{x}} \neq \hat{g}|_{i,\overline{x}}$. Moreover, since we assume that g is locally legal, we get that $g|_{i,\overline{x}}$ is a legal codeword. Note that also $\hat{g}|_{i,\overline{x}}$ is a legal codeword, due to the assumption that $\hat{g}$ is a codeword of C^m and to Fact 3.5. We conclude that if $\overline{x}$ is both locally legal and satisfies $f^{(\psi,g)}(\overline{x}) \neq f^{(\psi,\hat{g})}(\overline{x})$, it must lie on some axis-parallel line $L \subseteq D^m$ such that $g|_L$ and $\hat{g}|_L$ are distinct codewords of C. We refer to such an axis-parallel line L as a *faulty line.*

Therefore, in order to upper bound the probability in Equation 8, it suffices to upper bound the fraction of points that are contained in faulty lines L. To this end, it suffices to show that for every direction $i \in [m]$, the fraction of points that are contained in faulty lines in direction i is at most τ/δ_C, and this will imply the required upper bound by the union bound.

Fix a direction $i \in [m]$. Observe that every faulty line L in direction i contains at least $\delta_C \cdot |D|$ points on which g and $\hat{g}$ differ, since $g|_L$ and $\hat{g}|_L$ are distinct codewords of C. On the other hand, since g is τ-close to $\hat{g}$, the total number of points on which g and $\hat{g}$ differ is at most $\tau \cdot |D|^m$. Since distinct lines in direction i are disjoint, we get that the total number of faulty lines in direction i is at most $\frac{\tau}{\delta_C} |D|^{m-1}$. Finally, since every line contains exactly $|D|$ points, it follows that the total number of points that are contained in faulty lines in direction i is at most $\frac{\tau}{\delta_C} \cdot |D|^m$, as required. ■

4.3 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 4.9 and Definition 4.7, we see that φ is satisfiable if and only if there exists $g \in C^{\otimes m}$ for which $f^{(\psi,g)}$ defined in (7) vanishes on H^m. The AG combinatorial Nullstellensatz (Theorem 3.16) 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 3.10. 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 $2/\varepsilon$. Our proof will start by describing the verifier's operation on input φ of size n, followed by an analysis of its proof length, completeness, soundness, randomness complexity, query complexity, verifier complexity and decision complexity.

Verifier's operation The verifier V applies the reductions in (1), i.e., the reduction of Theorem 4.5 followed by the reduction of Theorem 4.9. 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 4.5. 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 (4) 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 3.13, 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 3.16 on this code. Note that every set $H \subseteq D$ of size $\leq 2k$ satisfies Equation (5), because

$$2k < \left(\frac{1}{100}\right) \cdot 200k \leq (\frac{1}{12} - \frac{\delta}{6} - \frac{2}{\sqrt{q}-1}) \cdot |D|.$$

Now V applies Theorem 4.9 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):

- A basis for each C_j, $j \in [d_{\mathsf{mult}}]$.
- For every $\alpha, \beta \in D$, a permutation π of D that (1) is an automorphism of each C_j, $j \in [d_{\mathsf{mult}}]$, and (2) maps α to β.
- The pair of functions $\xi = \xi_H, \xi' = \xi'_H : D \to \mathbb{F}_q$ defined in Theorem 3.16 where $H \subset D, |H| \leq 2k$ is part of the aggregate-AGCSP instance ψ and defined in Theorem 4.9. (Note that our earlier discussion showed that Equation (5) is satisfied and thus Theorem 3.16 does apply here.)

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. (7)) vanishes on H^m as per the AG combinatorial Nullstellensatz Theorem 3.16. The verifier performs the following checks while recycling randomness.

1. **Tensor test:** Invoke the local tester of Theorem 3.10 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)

$$2/\varepsilon < m \leq 2/\varepsilon + 1, \quad |D| \leq \frac{200k}{\rho} \leq 2^{270} \cdot (n^{1/m} + 1)$$

we conclude that the proof bit-length, for sufficiently large n, is at most

$$(m+2) \cdot \log_2 q \cdot |D|^m < \left(\frac{2}{\varepsilon} + 3\right) \cdot 16 \cdot 2^{540/\varepsilon + 270} \cdot n \leq c_\varepsilon \cdot n \tag{9}$$

where, asymptotically (i.e., as $\varepsilon \to 0$), $c_\varepsilon \leq 2^{c'/\varepsilon}$ for $c' < 600$.

Completeness. Suppose φ is satisfiable. Then by the completeness of Theorem 4.5 and Theorem 4.9 we know that ψ is satisfiable. This means there exists $g \in C^{\otimes m}$ such that the function $f^{(\psi,g)}$ defined in (7) belongs to $(C_d)^{\otimes m}$ and vanishes on H^m. Hence, by the AG-Nullstellensatz Theorem 3.16 there exist $f'_1, \dots, f'_m \in (C_{4d})^{\otimes m}$ such that (13) holds with respect to $f = f^{(\psi,g)}$. Inspection reveals that all of the verifier tests pass with probability 1 and we conclude the PCP system has perfect completeness as claimed.

Soundness. Suppose φ is not satisfiable. Then by the soundness of Theorem 4.5 and Theorem 4.9 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 (7), vanishes on H^m. Suppose the verifier is given the proof $g, f'_1, \dots, f'_m$. 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 $f'_1, \dots, f'_m$ 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 3.10). Thus, we may focus on the case in which g is $(\delta^{m+1}/2m)$-close to $(C_d)^{\otimes m}$ and all of $f'_1, \dots, f'_m$ are $(\delta^{m+1}/2m)$-close to $(C_{4d})^{\otimes m}$. In this case, $g, f'_1, \dots, f'_m$ are close to unique codewords $\hat{g}, \hat{f}'_1, \dots, \hat{f}'_m$ 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 $f'_1, \dots, f'_m$ agree with $\hat{f}'_1, \dots, \hat{f}'_m$ 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 3.16 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 4.12, 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 $f'_1, \dots, f'_m$ agree with $\hat{f}'_1, \dots, \hat{f}'_m$ 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 $f'_1, \dots, f'_m$ agree with $\hat{f}'_1, \dots, \hat{f}'_m$ 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 $O(\frac{1}{\delta^m})$ times. The randomness for these repetitions can be somewhat reduced using by-now-standard randomness-efficient samplers, e.g., based on expander-walks (cf. [Gol11]).

Randomness complexity. Each invocation of the local tester of Theorem 3.10 requires $\log_2(|D|^m) + O(\log m)$ random bits, and choosing $\overline{x}$ requires $\log_2(|D|^m)$ random bits. Since the verifier recycles the randomness, all the operation requires a total of $\log_2(|D|^m) + O(\log m) = \log_2 n + O_m(1)$ random bits.

For the case of rejection probability $\geq 1/2$, the randomness complexity is $\log_2 n + O(\log m + \frac{1}{\delta^m}) = \log_2 n + 2^{O(\frac{1}{\varepsilon})}$.

Query complexity. Each invocation of the local tester of Theorem 3.10 requires $|D|^2$ queries, for a total of $(m+1)\cdot|D|^2$ queries. The zero test uses $1 + m\cdot t = O(m\cdot|D|^2)$ queries to g to compute $f^{(\psi,g)}(\overline{x})$, another m queries to $f'_1,\ldots,f'_m$, and another $m\cdot|D|$ queries to verify that $\overline{x}$ is locally legal for g. All in all, the verifier uses $O(m\cdot|D|^2)$ queries. Now, recall that $|D| = O(k)$ and that $k = O(n^{1/m})$, and therefore the query complexity of the verifier is $O(m\cdot n^{2/m})$. By setting m to be sufficiently large constant, we can ensure that the query complexity is smaller than $O(\text{poly}(\frac{1}{\varepsilon})\cdot n^{\varepsilon})$.

For the case of rejection probability $\geq 1/2$, the query complexity is $O\left(\frac{1}{\delta^m}\cdot n^{\varepsilon}\right) = 2^{O(\frac{1}{\varepsilon})}\cdot n^{\varepsilon}$.

Verifier complexity. The verifier clearly runs in time $O_{\varepsilon}(1)\cdot\text{poly}(n)$.

Decision complexity. The decision predicate of the verifier consists of:

1. $\text{poly}(|D|)$ arithmetic operations on the field elements it queries.
2. Verifying membership in the code C, which can also be done using $\text{poly}(|D|)$ arithmetic operations since C is linear.
3. Invoking the local tester of Theorem 3.10, which uses $\text{poly}(|D|)$ operations.

Since $q = O(1)$, we conclude that the running time of V is polynomial in $|D|$. Since $|D| = O(n)^{O(\varepsilon)}$, we get that the decision complexity is at most $n^{O(\varepsilon)}$ for sufficiently large n.

■

4.4 A proof of Main Theorem 2.3 using the sum-check protocol

In this section, we give an alternative proof of our main theorem (Theorem 2.3) using the sum-check protocol. The PCP constructed this way is less efficient in terms of randomness, but has the advantage that it requires less sophisticated algebraic machinery. In particular, it requires only the multiplication property and transitivity property of the codes, and does not require the AG combinatorial Nullstellensatz (Theorem 3.16).

Theorem 4.13 (2.3, restated with larger randomness complexity) *For every $\varepsilon > 0$ and every $n \in \mathbb{N}$, there exists a (non-uniform) PCP verifier V_n for instances of circuit-SAT of size n with query complexity $O_{\varepsilon}(n^{\varepsilon})$, proof length $O_{\varepsilon}(n)$, rejection probability $\Omega_{\varepsilon}(1)$, verifier complexity $O_{\varepsilon}(1)\cdot n^{O(1)}$, randomness complexity $2\log n + O_{\varepsilon}(1)$, and decision complexity $O_{\varepsilon}(n^{\varepsilon})$.*

The verifier works by applying the generalization of the sum-check protocol to general codes of [Mei10]. We describe this protocol below, both for the sake of completeness and in order to analyze the resulting proof length (which is not specified by [Mei10]). Fix $\varepsilon > 0$, and let $m = O(1/\varepsilon)$

be an integer to be chosen later. Fix an instance φ of circuit-SAT of size n. We describe the action of the PCP verifier $V \stackrel{\text{def}}{=} V_n$ on the instance φ. The verifier V starts by applying the reduction of Theorem 4.5 to φ, which results in an instance G of Hypercube-CSP of size $O_m(n)$ over a subgraph of $\mathcal{H}_{k,m}$ for $k = O(n^{1/m})$. Next, V applies to G the reduction of Theorem 4.9 with $d = |\Sigma| = 4$ (where Σ is the alphabet of G), with C being the good code of Theorem 3.13 with message length $2k$, and with the other parameters chosen such that they satisfy the requirements below. We assume that the code C is hardwired to the verifier (this is where we use the non-uniformity of the verifier). The latter reduction results in an instance of aggregate-AGCSP

$$\psi = (m, d, t, \mathbb{F}, \vec{C}, H, \pi_1, \dots, \pi_t, Q^{(\psi)})$$

satisfying the following properties:

- ψ is satisfiable if and only if φ is satisfiable.
- $|\mathbb{F}| = O_m(1)$.
- $\vec{C} = (C_1, \dots, C_d)$ is a multiplication code family.
- $C \stackrel{\text{def}}{=} C_1$ has block length equal to $\ell \stackrel{\text{def}}{=} O_m(n^{1/m})$.
- The assignments to ψ are of bit-length $O(\ell^m \cdot \log |\mathbb{F}|) = O_m(n)$.
- $t \le \ell^2$.
- For each $j \in [d]$, the code C_j has relative distance at least $\delta_C \stackrel{\text{def}}{=} 1 - \frac{1}{10m}$.

Note that these parameters satisfy Equation (2) (by choosing $|\mathbb{F}| = q = \text{poly}(m)$, $\delta = 1 - \frac{1}{10m}$, $d_{\mathsf{mult}} = d = 4$, and $\rho = O(\frac{1}{d_{\mathsf{mult}} c_q m})$).

As usual, we view the codewords of each C_j as functions from D to $\mathbb{F}$, and its messages as functions from H to $\mathbb{F}$.

The verifier V checks that ψ is satisfiable roughly as follows: V expects to be given in the proof string a satisfying assignment g to ψ, and needs to check that $f^{(\psi,g)}$ vanishes on H^m. To this end, V considers an encoding G of $f^{(\psi,g)}|_{H^m}$ via a tensor code $(C')^m$, and wishes to verify that $G(\overline{x}) = 0$ for a random coordinate $\overline{x}$. In order to do the latter verification, the verifier V uses Claim 3.7 to write $G(\overline{x})$ in the following "sum-check form":

$$G(\overline{x}) = \sum_{z_1 \in H} \alpha_{1,z_1} \cdot \dots \sum_{z_m \in H} \alpha_{m,z_m} \cdot f^{(\psi,g)}(z_1, \dots, z_m).$$

Then, V uses the sum-check protocol to verify that the latter exponential sum evaluates to zero. This can be done by requiring the prover to provide all the partial sums of the form

$$\sum_{z_{t+1} \in H} \alpha_{t+1,z_{t+1}} \cdot \dots \sum_{z_m \in H} \alpha_{m,z_m} \cdot f^{(\psi,g)}(y_1, \dots, y_{t-1}, w, z_{t+1}, \dots, z_m).$$

In the rest of this section, we give a more formal and rigorous description of how V verifies that ψ is satisfiable. We first describe how the proof strings of satisfiable instances ψ look like, and then describe the action of V on ψ.

The proof strings of V. Suppose that ψ is satisfiable. We describe the proof string that convinces V to accept ψ with probability 1. The proof string contains a satisfying assignment g to ψ. Recall that the function $f^{(\psi,g)} : D^m \to \mathbb{F}$ is defined by

$$f^{(\psi,g)} = Q^{(\psi)}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x})).$$

Recall that g is a codeword of $C^{\otimes m}$, and therefore, by the property of the function $Q^{(\psi)}$ (see Definition 4.7), it holds that $f^{(\psi,g)}$ is a codeword of $(C_d)^{\otimes m}$.

Let $C' = \{f' : D' \to \mathbb{F}\}$ be a linear systematic evaluation code of relative distance $1 - \frac{1}{10m}$ and rate $\Omega_m(1)$, whose messages are functions $h' : H \to \mathbb{F}$. Note that $(C')^{\otimes m}$ has relative distance $\geq \frac{9}{10}$. Let $G : (D')^m \to \mathbb{F}$ be the encoding of $f^{(\psi,g)}|_{H^m}$ by C', and recall that since g is satisfying, G is the all-zeros codeword (as $f^{(\psi,g)}$ vanishes on H^m).

Notation 4.14 *For every $\overline{x} \in (D')^m$ and $t \in [m]$, define by $x_{<t}$ (respectively, $x_{>t}$) the prefix $(x_1, \ldots, x_{t-1})$ (respectively, the suffix $(x_{t+1}, \ldots, x_m)$). In particular, $x_{<1}$ and $x_{>m}$ are the empty vectors. We use the same notation for vectors $\overline{y} \in D^m$.*

Recall that by Claim 3.7, for every $\overline{x} \in (D')^m$ we can write

$$G(\overline{x}) = \sum_{z_1 \in H} \alpha_{1,z_1} \cdot \ldots \sum_{z_m \in H} \alpha_{m,z_m} \cdot f^{(\psi,g)}(z_1, \ldots, z_m).$$

Now, the proof string π contains, for each such $\overline{x}$, for every $\overline{y} \in D^m$, and for each $t \in [m]$, the function $f_{y_{<t},x_{>t}} : D \to \mathbb{F}$ defined by

$$f_{y_{<t},x_{>t}}(w) = \sum_{z_{t+1} \in H} \alpha_{t+1,z_{t+1}} \cdot \ldots \sum_{z_m \in H} \alpha_{m,z_m} \cdot f^{(\psi,g)}(y_1, \ldots, y_{t-1}, w, z_{t+1}, \ldots, z_m).$$

Note that by Claim 3.9, the function $f_{y_{<t},x_{>t}}$ is a codeword of C_d. Also note that the function $f_{y_{<t},x_{>t}}$ is indeed determined by $y_{<t}$, and $x_{>t}$ since the scalars α_{j,z_j} are determined by x_j for each $t+1 \leq j \leq m$ by Claim 3.7. In particular, the function $f_{y_{<t},x_{>t}}$ is independent of $x_1, \ldots, x_t$ and $y_t, \ldots, y_m$.

We show that π is of length $O_m(\ell^m) = O_m(n)$. To this end, note that since $|\mathbb{F}| = O_m(1)$, it does not matter whether we measure its length in field elements or in bits. Thus, it suffices to analyze length of g and the functions $f_{y_{<t},x_{>t}}$. It is easy to see that the assignment g is of length ℓ^m. As for the functions $f_{y_{<t},x_{>t}}$, observe that the length of each function $f_{y_{<t},x_{>t}}$ (represented by its truth table) is $|D| = \ell$. We show that there are at most $O_m(\ell^{m-1})$ such functions, so all those functions contribute a total of $O_m(\ell^m)$ field elements. To this end, note that for each $t \in [m]$, the total number of functions $f_{y_{<t},x_{>t}}$ is $|D|^{t-1} \cdot |D'|^{m-t} = O_m(\ell^{m-1})$, since $f_{y_{<t},x_{>t}}$ is determined by $y_1, \ldots, y_{t-1}, x_{t+1}, \ldots, x_m$. Thus, the total number of such functions for all $t \in [m]$ is at most $m \cdot O_m(\ell^{m-1}) = O_m(\ell^{m-1})$. We conclude that the length of π is $O_m(\ell^m) = O_m(n)$, as required.

The action of V on ψ. We turn to describe the action of the verifier V on arbitrary instance ψ and proof string π. We partition π to truth tables of functions g and $f_{y_{<t},x_{>t}}$, though now we have no guarantee on the relationships between those functions. Let $\hat{g}$ be the codeword of $C^{\otimes m}$ closest to g, breaking ties arbitrarily, and let us denote $\hat{f} \overset{\text{def}}{=} f^{(\psi,\hat{g})}$. The verifier starts by applying the local testing procedure of Theorem 3.10 to g.

Next, define a code C' as before, and let G be the encoding of $\hat{f}|_{H^m}$ via $(C')^{\otimes m}$. The verifier V chooses a uniformly distributed point $\overline{x} \in (D')^m$. By Claim 3.7, there are scalars $\alpha_{t,z}$ satisfying

$$G(\overline{x}) = \sum_{z_1 \in H} \alpha_{1,z_1} \cdot \ldots \sum_{z_m \in H} \alpha_{m,z_m} \cdot \hat{f}(z_1, \ldots, z_m). \tag{10}$$

The verifier V computes the scalars $\alpha_{t,z}$. Now, V chooses a uniformly distributed $\overline{y} \in D^m$, and checks that the following conditions hold:

1. $\overline{y}$ is 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.
2. For every $t \in [m]$, the function $f_{y_{<t},x_{>t}}$ is a codeword of C_d.
3. $\sum_{z_1 \in H} \alpha_{1,z_1} f_{y_{<1},x_{>1}}(z_1) = 0$.
4. For every $1 \le t \le m-1$, it holds that $f_{y_{<t},x_{>t}}(y_t) = \sum_{z_t \in H} \alpha_{t,z_t} \cdot f_{y_{<t+1},x_{>t+1}}(z_{t+1})$.
5. $f_{y_{<m},x_{>m}}(y_m) = f^{(\psi,g)}(\overline{y})$ (where the right hand side is computed by making $1 + t \cdot m$ accesses to g).

The verifier accepts if all the above conditions hold, and rejects otherwise. We note that the verifier chooses the points $\overline{x}, \overline{y}$ using the same randomness that was used to invoke the local testing procedure of Theorem 3.10, in order to save randomness.

This concludes the description of the verifier. We turn to analyze its completeness, soundness, query complexity, and decision complexity.

Completeness. Suppose that ψ is satisfiable. We choose the proof string π as explained above. In this case, we get that $g = \hat{g}$, and hence $f^{(\psi,g)} = f^{(\psi,\hat{g})} = \hat{f}$. Since g is a codeword of $C^{\otimes m}$, it follows by Fact 3.5 that all the axis-parallel lines of g are codewords of C, hence Condition 1 holds. It holds that $f^{(\psi,g)}$ is a codeword of $(C_d)^{\otimes m}$ by the property of $Q^{(\psi)}$ (see Definition 4.7), and therefore, all the functions $f_{y_{<t},x_{>t}}$ are codewords of C_d, so Condition 2 above holds. Condition 3 above holds since G is the all-zeros codeword (because $f^{(\psi,g)}|_{H^m}$ is all-zeros), and since the sum on the left-hand side is equal to $G(\overline{x})$ due to the definition of $f_{x_{>1}}$, and to Equation 10. Condition 4 above holds by the definition of the functions $f_{y_{<t},x_{>t}}$. Finally, it is not hard to see that the function $f_{y_{<m}}(\cdot)$ is, by definition, equal to the function $f^{(\psi,g)}(y_1, \ldots, y_{m-2}, y_{m-1}, \cdot)$, so Condition 5 holds as well. We conclude that for this choice of π, the verifier V accepts with probability 1.

Soundness. Suppose that ψ is unsatisfiable. We first note that if g is $\delta_C^m/2m$-far from the code $C^{\otimes m}$, the local tester of $C^{\otimes m}$ rejects g with probability at least $\gamma_m \cdot \delta_C^m/2m = \Omega_m(1)$ (where γ_m is defined as in Theorem 3.10). We thus focus on the case where g is $\delta_C^m/2m$-close to $C^{\otimes m}$, and in particular, to the unique codeword $\hat{g}$. Since ψ is unsatisfiable, we get that $\hat{f} \overset{\text{def}}{=} f^{(\psi,\hat{g})}$ can not vanish on H^m. This implies that G is a non-zero codeword of $(C')^{\otimes m}$, and in particular, $G(\overline{x}) \neq 0$ with probability at least $\delta_{(C')^{\otimes m}} = \left(1 - \frac{1}{10m}\right)^m \ge \frac{9}{10}$.

Let us focus now on the case where $G'(\overline{x}) \neq 0$, and assume that V errs otherwise. For each $t \in [m]$ and $y_1, \ldots, y_{t-1} \in D$ define a function $\hat{f}_{y_{<t},x_{>t}} : D \to \mathbb{F}$ by

$$\hat{f}_{y_{<t},x_{>t}}(y_t) \overset{\text{def}}{=} \sum_{z_{t+1} \in H} \alpha_{t+1,z_{t+1}} \cdot \ldots \sum_{z_m \in H} \alpha_{m,z_m} \cdot \hat{f}(y_1, \ldots, y_t, z_{t+1}, \ldots, z_m).$$

Now, let $\overline{y}$ be the vector chosen by the verifier. One of the following three cases must occur:

1. $f_{y_{<1},x_{>1}} = \hat{f}_{y_{<1},x_{>1}}$: In this case, the verifier rejects immediately since
$$\sum_{z_1 \in H} \alpha_{1,z_1} \hat{f}_{y_{<t},x_{>t}}(z_1) = G(\overline{x}) \neq 0,$$
which means that Condition 3 is violated.

2. $f_{y_{<m},x_{>m}} \neq \hat{f}_{y_{<m},x_{>m}}$: In this case, $f_{y_{<m},x_{>m}}$ and $\hat{f}_{y_{<m},x_{>m}}$ are two distinct codewords of C_d, and therefore they disagree on at least $1 - \frac{1}{10m}$ of their inputs. In particular, with probability at least $1 - \frac{1}{10m}$ it holds that
$$f_{y_{<m},x_{>m}}(y_m) \neq \hat{f}_{y_{<m},x_{>m}}(y_m) = \hat{f}(y_{<m}, y_m).$$
Note that we used here the fact that y_m is a uniformly distributed element of D that is independent of $f_{y_{<m},x_{>m}}$ and $\hat{f}_{y_{<m},x_{>m}}$.

   Now, suppose the latter equality holds. Then, in order for the verifier not to reject, the following two conditions must hold:

   - $f^{(\psi,g)}(\overline{y}) \neq \hat{f}(\overline{y})$, since otherwise it would hold that $f_{y_{<m},x_{>m}}(y_m) \neq f^{(\psi,g)}(\overline{y})$, in which case the verifier would reject since Condition 5 would be violated.
   - $\overline{y}$ is locally legal for g, since otherwise the verifier would reject since Condition 1 would be violated.

   Finally, by Lemma 4.12, the probability that both conditions hold is at most $m \cdot \delta_C^m / (2m \cdot \delta_C) \leq \frac{1}{2}$ (recall that $\hat{f} \overset{\text{def}}{=} f^{(\psi,\hat{g})}$). Thus, the probability that the verifier does not reject in this case is at most $\frac{1}{10m} + \frac{1}{2} \leq \frac{2}{3}$.

3. For some $t \in [m-1]$ it holds that $f_{y_{<t},x_{>t}} \neq \hat{f}_{y_{<t},x_{>t}}$ and $f_{y_{<t+1},x_{>t+1}} = \hat{f}_{y_{<t+1},x_{>t+1}}$: In this case, $f_{y_{<t},x_{>t}}$ and $\hat{f}_{y_{<t},x_{>t}}$ are two distinct codewords of C_d, and therefore they disagree on at least $1 - \frac{1}{10m}$ of their inputs. In particular, with probability at least $1 - \frac{1}{10m}$ it holds that
$$\begin{aligned} f_{y_{<t},x_{>t}}(y_t) &\neq \hat{f}_{y_{<t},x_{>t}}(y_t) \\ &= \sum_{z_t \in H} \alpha_{t,z_t} \hat{f}_{y_{<t+1},x_{>t+1}}(z_{t+1}) \\ &= \sum_{z_t \in H} \alpha_{t,z_t} f_{y_{<t+1},x_{>t+1}}(z_{t+1}), \end{aligned}$$
in which case the verifier rejects. Here we again used the fact y_t is a uniformly distributed element of D that is independent of $f_{y_{<t},x_{>t}}$ and $\hat{f}_{y_{<t},x_{>t}}$.

Taking union bound over all the above cases, we get that the verifier rejects with probability at least $1 - \frac{2}{3} - \frac{1}{10m} \cdot m \geq \frac{2}{10}$. After subtracting from it the probability that $G(\overline{x}) = 0$, we conclude that the verifier rejects with probability $\frac{1}{10} = \Omega_m(1)$, as required.

Query complexity. The verifier V queries its oracle for three purposes:

1. Testing that g is close to $C^{\otimes m}$ - this uses ℓ^2 queries.
2. Checking that $\overline{y}$ is locally legal for g - this requires at most $m \cdot \ell$ queries, since there are m lines, each containing ℓ points.

3. Reading the truth tables of the functions $f_{y_{<t}, x_{>t}}$ - this requires $m \cdot \ell$ queries - there are m such functions, and the truth table of each of them is of length ℓ (since it supposed to be a codeword of C_d).

4. Computing $f^{(\psi,g)}(\overline{y})$ - this requires $1 + m \cdot t = O_m(\ell^2)$ queries to g.

We conclude that the total number of queries is $O_m(\ell^2) = O_m(n^{O(1/m)})$. By choosing m to be sufficiently large, we get that latter bound is at most n^{ε} for sufficiently large n.

Verifier complexity. The verifier clearly runs in time $O_{\varepsilon}(1) \cdot \mathrm{poly}(n)$.

Decision complexity. The decision predicate of the verifier consists of:

1. $\mathrm{poly}(\ell)$ arithmetic operations on the field elements of g and the functions $f_{y_{<t}, x_{>t}}$.

2. Verifying membership in the code C, which can also be done using $\mathrm{poly}(\ell)$ arithmetic operations since C is linear.

3. Invoking the local tester of Theorem 3.10, which uses $\mathrm{poly}(\ell)$ operations.

Since $|\mathbb{F}| = O(1)$, we conclude that the running time of V is polynomial in the number of its queries, which is $O(n^{1/m})$. Thus, by setting m to be sufficiently large, we get that the decision complexity is at most n^{ε} for sufficiently large n.

Randomness complexity. Invoking the local testing procedure of Theorem 3.10 requires at most $\log O_m(n) = \log n + O_m(1)$ random bits. Choosing the points $\overline{x}$ and $\overline{y}$ requires $2 \log n + O_m(1)$ random bits, since each of those points requires $\log O_m(n)$. Since the verifier recycles the randomness complexity used for the local testing, we get that the total number of random bits used by the verifier is $2 \log n + O_m(1)$.

5 Hypercube CSPs — Proof of Theorem 4.5

In this section we prove the first step in the reduction (1) — Theorem 4.5. In particular, we show that every instance of circuit-SAT can be reduced to an instance Hypercube-CSP with only a constant factor blowup in the length. Such a reduction is useful since arithmetization is more efficient when applied to instances of Hypercube-CSP.

The reduction goes in two steps. First, in Proposition 5.1, we reduce the circuit-SAT problem to graph-CSP (see Definition 4.2). This reduction yields a constraint graph whose underlying graph resembles the topology of the original circuit-SAT instance. Then, in Lemma 5.2, we embed the latter constraint graph into the hypercube, thus reducing graph-CSP to Hypercube-CSP. This part follows similar reductions that were used in previous works in the PCP literature starting from [BFLS91, PS94], which were in turn based on routing techniques (see, e.g., [Lei92]).

Proposition 5.1 (Reduction from circuit-SAT to graph-CSP) *There exists a polynomial time procedure that maps every circuit φ of size n to a constraint graph G_{φ} of size n over alphabet $\Sigma = \{0,1\}^2$ such that G_{φ} is satisfiable if and only if φ is satisfiable.*

Proof: Fix a circuit φ. We describe the construction of the corresponding constraint graph G_φ. For each gate g of φ, the graph G_φ has a corresponding vertex v_g, and for each wire (g_1, g_2) of φ the graph G_φ has a corresponding edge (v_{g_1}, v_{g_2}). The alphabet of G_φ is $\Sigma = \{0,1\}^2$. Intuitively, an assignment $\sigma : V \to \Sigma$ should label a vertex v_g with a symbol $b_1 b_2 \in \{0,1\}^2$ if the first input wire of the gate g carries the value b_1, and the second input wire carries the value b_2 (if g has only one input wire, the extra bit in the symbol can be chosen arbitrarily). The constraint of an edge (v_{g_1}, v_{g_2}) of G will verify that $\sigma(v_{g_2})$ contains the output of g_1 on the inputs contained in $\sigma(v_{g_1})$.

More formally, for each edge $e = (v_{g_1}, v_{g_2})$ of G_φ, we define the constraint c_e that corresponds to e as follows. Without loss of generality, assume that (g_1, g_2) is the first input wire of g_2. Then, c_e contains a pair of symbols $(a_1 a_2, b_1 b_2) \in \left(\{0,1\}^2\right)^2$ if and only if the following conditions hold:

1. b_1 is the output of the gate g_1 when fed with inputs a_1, a_2.
2. if g_2 is the output gate, then it outputs 1 when fed with inputs b_1, b_2.

It is not hard to see that G_φ is satisfiable if and only if φ is satisfiable. ■

We now turn to the second step of the reduction. Recall that we denote by $\mathcal{H}_{k,m}$ the m-dimensional k-ary hypercube.

Lemma 5.2 *There exists a polynomial time procedure when given as input*

- *A constraint graph $G = (V, E)$ of size n and alphabet Σ; and*
- *An integer $m \in \mathbb{N}$,*

outputs a constraint graph G' of size $\leq 2m \cdot 4^{m+2} n$ and alphabet Σ, whose underlying graph is a 4-regular subgraph of $\mathcal{H}_{4\lceil (4n)^{1/m} \rceil, m}$, such that G' is satisfiable if and only if G is satisfiable.

By combining Proposition 5.1 and Lemma 5.2, Theorem 4.5 follows immediately:

Theorem 5.3 (4.5, restated) *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 $k \leq 4((4 \cdot n)^{1/m} + 1)$.*

The proof of Lemma 5.2 is based on the following proposition, whose proof is deferred to end of this section.

Proposition 5.4 *There exists a polynomial time procedure that takes as input a permutation π on $[k]^m$ and outputs a collection $\mathcal{P}$ of vertex-disjoint paths on $\mathcal{H}_{2k,m}$ of length $2m$ that connect every vertex $v \in [k]^m$ to $\pi(v)$ (note that the paths are on the $2k$-ary hypercube and not on the k-ary hypercube). Here, we allow a vertex v to be both the first vertex of one path in $\mathcal{P}$ and the last vertex of another path in $\mathcal{P}$, but other than that we require the paths to be vertex disjoint.*

Remark 5.5 *Proposition 5.4 is a generalization of known results for the Boolean hypercube (see, e.g., [Lei92]), and the particular proof we use follows closely the proof of [DM11, Fact 4.5].*

We turn to prove Lemma 5.2.

Proof of Lemma 5.2: We describe the action of the procedure on inputs $G = (V, E)$ and m. We first observe that without loss of generality, we may assume that the graph G is 4-regular, since one can use standard techniques to transform any constraint graph of size n into a 4-regular constraint graph of size $4n$ in polynomial time[5]. This is similar to the reduction of **3SAT** to the special case of **3SAT** in which every variable appears at most three times. Since G is 4-regular, we can partition its edges to four disjoint matchings $M_1, \dots, M_4$ in polynomial time (see, e.g., [Cam98, Proposition 18.1.2]).

Let $k = \lceil |V|^{1/m} \rceil \le (4n)^{1/m} + 1$, and let us identify the vertices V of G with $[k]^m$. The procedure will construct the constraint graph G' by embedding G in the hypercube $\mathcal{H} \overset{\text{def}}{=} \mathcal{H}_{4k,m}$. More specifically, observe that the graph $\mathcal{H}$ contains four copies of the hypercube $\mathcal{H}_{2k,m}$: the first copy is the induced subgraph over the vertices in $[2k]^m$, the second over $([k] \cup ([3k] - [2k]))^m$, etc. Let us denote these induced subgraphs by $\mathcal{H}_1, \dots, \mathcal{H}_4$ respectively. The procedure will embed the edges of the matchings $M_1, \dots, M_4$ on vertex-disjoint paths in $\mathcal{H}_1, \dots, \mathcal{H}_4$ respectively by applying Proposition 5.4 to each of those matchings, and those paths will form the edges of G'. Details follow.

The constraint graph G' will be a subgraph of the graph $\mathcal{H}$. In what follows, we describe how the edges of G' are constructed. For each $j \in [4]$, the procedure acts as follows. The procedure views M_j as a permutation π on $[k]^m$ (recall that we identify the vertices of G with $[k]^m$). The procedure then applies Proposition 5.4 to the permutation π and the graph $\mathcal{H}_j$, thus obtaining a collection $\mathcal{P}$ of vertex-disjoint paths in $\mathcal{H}_j$ that connect every pair of vertices in $[k]^m$ that are matched by π. In particular, for each edge $e = (u.v) \in M_j$ there is a path $p_e \in \mathcal{P}$ whose first vertex is u and whose last vertex is v. Now, for each edge $e = (u.v) \in M_j$, the procedure adds all the edges in p_e to G', where all the edges in p_e except for the last one are associated with the equality constraint, and the last edge in p_e is associated with the same constraint c_e of the edge e in G.

It should be clear that G' is satisfiable if and only if G is satisfiable. Moreover, since G' contains at most $2m$ edges for each edge of G (as each path p_e is of length at most $2m$), we get that G' is of size at most

$$2m(4\lceil (4n)^{1/m} \rceil)^m \le 2m4^{m+2}n.$$

It is also easy to see that the degree of G' is upper bounded by 4, and by adding dummy edges, we can make it 4-regular. Finally, it is not hard to see that the procedure runs in polynomial time, as required. ■

Finally, we prove Proposition 5.4.

Proof of Proposition 5.4: The procedure works by recursion on m. For $m = 1$, finding the required collection $\mathcal{P}$ is trivial - the procedure simply outputs the edges of the form $(v, \pi(v))$. Assume that $m > 1$. Consider the bipartite graph G whose two sides are copies of $[k]^{n-1}$, and in which two vertices $(i_1, \dots, i_{m-1}), (j_1, \dots, j_{m-1}) \in [k]^{m-1}$ are connected by an edge if and only if there exist $i_m, j_m \in [k]$ such that $\pi(i_1, \dots, i_m) = (j_1, \dots, j_m)$. Clearly, G is a k-regular graph, and therefore by [Cam98, Proposition 18.1.2] it is possible to partition the edges of G to matchings $\pi_1, \dots, \pi_k$ in polynomial time.

[5]The factor-4 blowup comes from first reducing out-degree of the circuit by doubling the number of gates, and then making the graph 4-regular by doubling the number of gates again.

Now, the procedure views the matchings $\pi_1, \ldots, \pi_k$ as permutations on $[k]^{m-1}$, and invokes itself recursively to find corresponding collections of vertex-disjoint paths $\mathcal{P}_1, \ldots, \mathcal{P}_k$ in $\mathcal{H}_{2k,m-1}$. Finally, in order to construct the output collection $\mathcal{P}$, the procedure constructs a path for each vertex $(i_1, \ldots, i_d) \in [k]^m$ as follows: Let $(j_1, \ldots, j_m) = \pi(i_1, \ldots, i_m)$. Recall that in the graph G there is an edge between the vertices $(i_1, \ldots, i_{m-1})$ and $(j_1, \ldots, j_{m-1})$, and suppose that this edge belongs to the matching π_t. Let $p = (v_1, \ldots, v_l)$ be the path in $\mathcal{P}_t$ that connects $(i_1, \ldots, i_{m-1})$ to $(j_1, \ldots, j_{m-1})$ in $\mathcal{H}_{2k,m-1}$ where $v_1 = (i_1, \ldots, i_{m-1})$ and $v_l = (j_1, \ldots, j_{m-1})$. Now, we choose the path p' that connects $(i_1, \ldots, i_m)$ to $(j_1, \ldots, j_m)$ in $\mathcal{H}_{2k,m}$ to be $(v'_0, \ldots, v'_{l+1})$ where $v'_0 = (i_1, \ldots, i_m)$, $v'_{l+1} = (j_1, \ldots, j_m)$, and for each $i \in [l]$, the vertex $v'_i \in [2k]^m$ is obtained from the vertex $v_i \in [2k]^{m-1}$ by appending $k+t$ to v_i. In other words, for each $i \in [l]$, we define v'_i by setting $(v'_i)_j = (v_i)_j$ for every $j \in [d-1]$ and $(v'_i)_m = k+t$.

We prove that the paths in $\mathcal{P}$ constructed this way are vertex disjoint and of length at most $2m$ by induction on m. For $d = m$ the proof is trivial. For $m > 1$, consider two paths p'_1 and p'_2. We show that p'_1 and p'_2 are vertex-disjoint. Let p_1 and p_2 be the paths in $\mathcal{H}_{2k-1,m}$ from which p'_1 and p'_2 were obtained. If p_1 and p_2 belong to two distinct collections $\mathcal{P}_{t_1}$ and $\mathcal{P}_{t_2}$, then it is clear that p'_1 and p'_2 must be vertex-disjoint, since the last coordinate of all the vertices in p'_1 (resp. p'_2) except for the first and the last vertices will be equal to $k+t_1$ (resp. $k+t_2$), and $k+t_1 \neq k+t_2$. If p_1 and p_2 belong to the same collection $\mathcal{P}_t$, then they are vertex-disjoint by the induction assumption, and therefore p'_1 and p'_2 must be vertex-disjoint as well. Finally, observe that by the induction assumption, the paths p_1 and p_2 are of length at most $2(m-1)$, and since p'_1 and p'_2 only add two edges to p_1 and p_2, we get that they are of length at most $2m$, as required.

We conclude by observing that the procedure runs in polynomial time. It is not hard to see that the running time of the procedure is described by the recursion formula $T(k,m) = k \cdot T(k, m-1) + \text{poly}(k^m)$, or in other words, $T(k,m) \leq k \cdot T(k, m-1) + k^{c_0 \cdot m}$ for some constant c_0. We prove that there exists a constant $c_1 \in \mathbb{N}$ such that $T(k,m) \leq k^{c_1 \cdot m}$ by induction on m: For $m = 1$ this is trivial, and for $m > 1$, it holds that

$$T(k,m) = k \cdot T(k, m-1) + k^{c_0 \cdot m} \leq k \cdot k^{c_1 \cdot (m-1)} + k^{c_0 \cdot m} \leq k^{c_1 \cdot m},$$

where the first inequality holds by the induction assumption and the second inequality holds for sufficiently large choice of c_1. ■

6 From Hypercube-CSP to aggregate-AGCSP

In this section we prove the second step in Reduction (1) — Theorem 4.9. First we reduce the Hypercube-CSP problem to a non-aggregated AG constraint satisfaction problem (AGCSP), one that is, indeed, a CSP according to the standard definition of the term. Then, in Section 6.2, we apply an aggregation step to reach the language aggregate-AGCSP that is the end-point of the reduction stated in Theorem 4.9.

6.1 A non-aggregated algebraic CSP

We turn to define our algebraic constraint satisfaction problem. We first recall some necessary notation, and then define the AGCSP problem.

Notation 6.1 (4.6, restated) *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 6.2 (AGCSP) *An instance of the AGCSP problem is a tuple*

$$\psi = (m, d, t, \mathbb{F}, C, H, \pi_1, \ldots, \pi_t, \{Q_{\overline{v}} \mid \overline{v} \in H^m\})$$

where

- *m, d, t are integers*
- *C is a systematic linear 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.*
- *For each $\overline{v} \in H^m$, the function $Q_{\overline{v}}$ is a polynomial of degree $< d$ over $1 + t \cdot m$ variables. The polynomials $Q_{\overline{v}}$ will serve as the constraints.*

An assignment *is a function $g : D^m \to \mathbb{F}$. We say g* satisfies *the instance if and only if*

- *g is a codeword of $C^{\otimes m}$, and*
- *for every $\overline{v} \in H^m$ it holds that*

$$Q_{\overline{v}}\left(g^{(\pi_1,\ldots,\pi_t)}(\overline{v})\right) = 0.$$

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

We now show how to reduce the hypercube CSP from Definition 4.4 (cf. Section 5) to our algebraic CSP.

Theorem 6.3 *There exists a polynomial time procedure with the following input-output behavior:*

- ***Input:***
  - *A number $m \in \mathbb{N}$.*
  - *An alphabet Σ.*
  - *A constraint graph G over Σ whose underlying graph is a 4-regular subgraph of $\mathcal{H}_{k,m}$.*
  - *A finite field $\mathbb{F}$.*
  - *A basis for the transitive evaluation code $C = \{f : D \to \mathbb{F}\}$ of message length at least $2k$.*
  - *For each $\alpha, \beta \in D$, a permutation π of D that (1) maps α to β, and (2) is an automorphism of C.*

- ***Output:*** *An instance of AGCSP*

$$\psi = (m, d \overset{\text{def}}{=} |\Sigma|, t, \mathbb{F}, C, H, \pi_1, \ldots, \pi_t, \{Q_{\overline{v}} \mid \overline{v} \in H^m\})$$

  with $|H| \leq 2k$, that is satisfiable if and only if G is satisfiable.

Proof: Since C is linear, we may assume without loss of generality that it is systematic. Let $H \subseteq D$ be such that the messages of C can be viewed as functions $h : H \to \mathbb{F}$, and such that the encoding $f : D \to \mathbb{F}$ of a message h satisfies $f|_H = h$. Note that $|H| \geq 2k$. Let $\pi_1, \ldots, \pi_t$ be a sequence of automorphisms of C_i such that for every $\alpha, \beta \in H$ there exists some automorphism π_j such that $\pi_j(\alpha) = \beta$ - those automorphisms are just a subset of the automorphisms given in the input.

We define the instance ψ of AGCSP that is the output of the procedure. We set

$$\psi = (m, d, t, \mathbb{F}, C, H, \pi_1, \ldots, \pi_t, \{Q_{\overline{v}} \mid \overline{v} \in H^m\})$$

where $d \overset{\text{def}}{=} |\Sigma|$, $t = O(k^2)$, and where the polynomials $\{Q_{\overline{v}} \mid \overline{v} \in H^m\}$ are defined as follows. Let $H_0 \subseteq H$ be an arbitrary subset of size k, and let us identify H_0^m with the vertices of G (i.e., the vertices of $\mathcal{H}_{k,m}$). Let us identify Σ with some subset of $\mathbb{F}$. We will have two kinds of polynomials $Q_{\overline{v}}$:

1. Those $Q_{\overline{v}}$'s for which $\overline{v} \in (H - H_0) \times H_0^{m-1}$. Those polynomials will check that the assignment g assigns to H_0^m values that are in Σ.

2. Those $Q_{\overline{v}}$'s for which $\overline{v} \in H_0^m$. Those polynomials will check that the assignment g, when restricted to H_0^m, is a satisfying assignment for G.

All the rest of the polynomials $Q_{\overline{v}}$ will be equal to the zero polynomial.

We start by defining the polynomials $Q_{\overline{v}}$ of the second kind, i.e., for $\overline{v} \in H_0^m$. For each vertex $\overline{v}$ of G, the polynomial $Q_{\overline{v}}$ verifies that the assignment g satisfies the constraints of $\overline{v}$ in G, and is constructed by arithmetizing those constraints. Fix $\overline{v} = (v_1, \ldots, v_m) \in H_0^m$, and let $\overline{u}_1$, $\overline{u}_2$, and $\overline{u}_3$ be its neighbors in G. Observe that we can express $\overline{u}_1$, $\overline{u}_2$, and $\overline{u}_3$ as

$$\begin{aligned}
\overline{u}_1 &= (v_1, \ldots, v_{j_1-1}, \pi_{l_1}(v_{j_1}), v_{j_1+1}, \ldots, v_m) \\
\overline{u}_2 &= (v_1, \ldots, v_{j_2-1}, \pi_{l_2}(v_{j_2}), v_{j_2+1}, \ldots, v_m) \\
\overline{u}_3 &= (v_1, \ldots, v_{j_3-1}, \pi_{l_3}(v_{j_3}), v_{j_3+1}, \ldots, v_m)
\end{aligned}$$

for some $l_1, l_2, l_3 \in [t]$ and $j_1, j_2, j_3 \in [m]$. Now, we denote variables of the polynomial $Q_{\overline{v}}$ by z and $\{y_{l,j}\}_{l=1,j=1}^{t,m}$, and define $Q_{\overline{v}}$ to be the unique polynomial that satisfies the following two requirements:

1. $Q_{\overline{v}}$ involves only the variables z, y_{l_1,j_1}, y_{l_2,j_2}, and y_{l_3,j_3}, and has degree at most $|\Sigma| - 1 < d$ in each of them.

2. For every assignment σ to G, the polynomial $Q_{\overline{v}}$ evaluates to 0 when $z, y_{l_1,j_1}, y_{l_2,j_2}, y_{l_3,j_3}$ are assigned $\sigma(\overline{v}), \sigma(\overline{u}_1), \sigma(\overline{u}_2), \sigma(\overline{u}_3)$ respectively if and only if σ satisfies the constraints $(\overline{v}, \overline{u}_1)$, $(\overline{v}, \overline{u}_2)$, and $(\overline{v}, \overline{u}_3)$.

Note that such a polynomial $Q_{\overline{v}}$ indeed exists, since it can be constructed using interpolation. This concludes the definition of polynomial $Q_{\overline{v}}$ for $\overline{v} \in H_0^m$.

We turn to define the polynomial $Q_{\overline{v}}$ of the first kind, i.e., for $\overline{v} \in (H - H_0) \times H_0^{m-1}$. Let $\eta : (H - H_0) \to H_0$ be a surjection onto H_0. Fix $\overline{v} = (H - H_0) \times H_0^{m-1}$, and let

$$\overline{u} = (\eta(v_1), v_2, \dots, v_m) \in H_0^m.$$

The polynomial $Q_{\overline{v}}$ verifies that the assignment g assigns to $\overline{u}$ a value in Σ, and is constructed as follows. Let π_{l_0} be an automorphism of C_i that maps v_1 to $\eta(v_1)$. Again, we denote variables of the polynomial $Q_{\overline{v}}$ by z and $\{y_{l,j}\}_{l=1,j=1}^{t,m}$. Now, we define $Q_{\overline{v}}$ to be the unique polynomial that involves only the variable $y_{l_0,1}$, and whose roots are exactly the values in Σ. Note that such a polynomial $Q_{\overline{v}}$ indeed exists, since it can be constructed using interpolation

Completeness. Suppose that G has a satisfying assignment $\sigma : H_0^m \to \Sigma$. Let $g : D^m \to \Sigma$ be a codeword of $C^{\otimes m}$ such that $g|_{H_0^m} = \sigma$: such a codeword exists since C is systematic, and therefore $C^{\otimes m}$ is systematic with messages being functions from H^m to $\mathbb{F}$. We claim that g is satisfying assignment of ψ. To this end, let us consider the two kinds of polynomials $Q_{\overline{v}}$ separately:

- Let $\overline{v} \in H_0^m$, and let $\overline{u}_1, \overline{u}_2, \overline{u}_3$, l_1, l_2, l_3, j_1, j_2, j_3 be defined as before. We need to show that $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v})) = 0$. Observe that in the expression $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v}))$, the variables z, y_{l_1,j_1}, y_{l_2,j_2}, y_{l_3,j_3} are assigned $g(\overline{v})$, $g^{\pi_{l_1,j_1}}(\overline{v})$, $g^{\pi_{l_2,j_2}}(\overline{v})$, $g^{\pi_{l_3,j_3}}(\overline{v})$ respectively, that $g(\overline{v}) = \sigma(\overline{v})$, and that for each $s \in [3]$ it holds that $g^{\pi_{l_s,j_s}}(\overline{v}) = g(\overline{u}_s) = \sigma(\overline{u}_s)$. Now, since σ satisfies the constraints of $\overline{v}$, we get that $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v})) = 0$ by the definition of $Q_{\overline{v}}$.
- Let $\overline{v} \in (H - H_0) \times H_0^{m-1}$, and let $\overline{u} \in H_0^m$ be defined as before. We need to show that $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v})) = 0$. Observe that in the expression $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v}))$, the variable $y_{l_0,1}$ is assigned $g^{\pi_{l_0},1}(\overline{v}) = g(\overline{u}) = \sigma(\overline{u})$. Since $\sigma(\overline{u}) \in \Sigma$, we get that $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v})) = 0$ by the definition of $Q_{\overline{v}}$.

Soundness. Suppose that ψ has a satisfying assignment $g : D^m \to \mathbb{F}$. We show that $\sigma \stackrel{\text{def}}{=} g|_{H_0^m}$ is a satisfying assignment of G. First, observe that the image of σ is indeed contained in Σ: Let $\overline{u} \in H_0^m$. We show that $g(\overline{u}) \in \Sigma$. Let $\overline{v} \in (H - H_0) \times H_0^{m-1}$ be such that $\overline{u} = (\eta(v_1), v_2, \dots, v_m)$ (such a $\overline{v}$ must exist since η is a surjection), and let l_0 be defined as before. Then, in the expression $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v}))$, the variable $y_{l_0,1}$ is assigned $g(\overline{u})$. Since $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v})) = 0$, we get by the definition of $Q_{\overline{v}}$ that $g(\overline{u}) \in \Sigma$ (as $Q_{\overline{v}}$ is of the first type).

Next, to show that σ is indeed a satisfying assignment for ψ, we show that for each $\overline{v} \in H_0^m$, the assignment σ satisfies the constraints of $\overline{v}$ in G. Fix $\overline{v}$ and let $\overline{u}_1, \overline{u}_2, \overline{u}_3$, l_1, l_2, l_3, j_1, j_2, j_3 be defined as before. Then, in the expression $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v}))$, the variables z, y_{l_1,j_1}, y_{l_2,j_2}, y_{l_3,j_3} are assigned $\sigma(\overline{v})$, $\sigma(\overline{u}_1)$, $\sigma(\overline{u}_2)$, and $\sigma(\overline{u}_3)$. Since $Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v})) = 0$, we get by the definition of $Q_{\overline{v}}$ that σ satisfies the constraints of $\overline{v}$ (as $Q_{\overline{v}}$ is of the second type). ■

6.2 Aggregating the constraint polynomials

The difference between AGCSP and aggregate-AGCSP is that (1) an instance of AGCSP contains a list of constraint polynomials, while an instance of aggregate-AGCSP contains a single constraint function $Q^{(\psi)}$, and (2) the code C in the AGCSP instance needs to be part of a multiplication code family $\vec{C}$. In this section we show how to reduce AGCSP to aggregate-AGCSP, thus proving Theorem 4.9.

Definition 6.4 (4.7, aggregate-AGCSP, restated) *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.*
- $C \overset{\text{def}}{=} C_1$ *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*

$$f^{(\psi,g)} : D^m \to \mathbb{F}, \quad f^{(\psi,g)}(\overline{x}) \overset{\text{def}}{=} Q^{(\psi)}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x})) \tag{11}$$

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.*

Theorem 6.5 (4.9, restated) *There exists a polynomial-time procedure with the following input-output behavior:*

- ***Input:***
  - *A number* $m \in \mathbb{N}$.
  - *An alphabet* Σ.
  - *A constraint graph* G *over* Σ *whose underlying graph is a* 4*-regular subgraph of* $\mathcal{H}_{k,m}$.
  - *A finite field* $\mathbb{F}$.
  - *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* $C \overset{\text{def}}{=} C_1$ *has message length at least* $2 \cdot k$, *and* $d_{\mathsf{mult}} \geq |\Sigma|$.
  - *For each* $\alpha, \beta \in D$, *a permutation* π *of* D *that (1) maps* α *to* β, *and (2) is an automorphism of* C_j *for each* $j \in [d]$.
- ***Output:*** *An instance of aggregate-AGCSP*

$$\psi = (m, d \overset{\text{def}}{=} |\Sigma|, t, \mathbb{F}, \vec{C}, H, \pi_1, \ldots, \pi_t, Q^{(\psi)})$$

with $|H| \leq 2k$, *that is satisfiable if and only if* G *is satisfiable.*

In order to prove Theorem 4.9 we will make use of the following two propositions, which follow easily from the characterization of tensor codes (Fact 3.5).

Proposition 6.6 *If $(C_1,\dots,C_d)$ is a multiplication code family, then the code $((C_1)^{\otimes m})^d$ (which is the d-fold multiplication of the code $(C_1)^{\otimes m}$, see Definition 3.11) is contained in $(C_d)^{\otimes m}$.*

Proposition 6.7 *For every codeword $g \in C^{\otimes m}$, an automorphism π of C and $i \in [m]$, it holds that the function $g^{\pi,i}$ is a codeword of $C^{\otimes m}$. Recall that $g^{\pi,i} : D^m \to \mathbb{F}$ is defined by*

$$g^{\pi,i}(x_1,\dots,x_m) \overset{\text{def}}{=} g(x_1,\dots,x_{i-1},\pi(x_i),x_{i+1},\dots,x_m).$$

Proof of Theorem 4.9: The procedure of Theorem 4.9 works as follows. First, it invokes the procedure of Theorem 6.3, giving it the code $C = C_1$ as input. This results in an instance

$$\psi_0 = (m, d \overset{\text{def}}{=} |\Sigma|, t, \mathbb{F}, C, H, \pi_1,\dots,\pi_t, \{Q_{\overline{v}} \mid \overline{v} \in H^m\})$$

of AGCSP. Then, the procedure constructs its output ψ from ψ_0 by replacing the polynomials $Q_{\overline{v}}$ with a corresponding function $Q^{(\psi)}$ (and also includes the entire multiplication code family $\vec{C}$ in ψ). Roughly, we construct $Q^{(\psi)}$ by summing all the terms of the form $1_{\overline{v}} \cdot Q_{\overline{v}}$, where $1_{\overline{v}}$ is the indicator function of $\overline{v}$. Details follow.

We will need some notation. For every $\overline{v} \in H^m$, let $1_{\overline{v}} : H^m \to \mathbb{F}$ be the indicator function of $\overline{v}$, i.e., the function that takes the value 1 on $\overline{v}$ and 0 everywhere else. In addition, let $1_{H^m} : H^m \to \mathbb{F}$ be the all-ones function, i.e., the function that takes the value 1 everywhere in H^m. We let $\tilde{1}_{\overline{v}} : D^m \to \mathbb{F}$ be any codeword of $C^{\otimes m}$ such that $\tilde{1}_{\overline{v}}|_{H^m} = 1_{\overline{v}}$ (i.e., its restriction to H^m equals $1_{\overline{v}}$), and we let $\tilde{1}_{H^m} : D^m \to \mathbb{F}$ be any codeword of $C^{\otimes m}$ such that $\tilde{1}_{H^m}|_{H^m} = 1_{H^m}$ (i.e., its restriction to H^m equals 1_{H^m}). Note that such codewords exist because C is a systematic code whose set of message coordinates contains H.

We first modify the polynomials $Q_{\overline{v}}$ such that they are homogeneous, i.e., such that all their monomials are of degree exactly $d-1$. Fix $\overline{v} \in H^m$, and consider the polynomial $Q_{\overline{v}} : \mathbb{F}^{1+t\cdot m} \to \mathbb{F}$. Let us denote the variables of this polynomial by z and $\{y_{l,j}\}_{l,j=1}^{t,m}$ as before. To make $Q_{\overline{v}}$ homogeneous, we add a new variable s, and multiply each monomial of $Q_{\overline{v}}$ by the appropriate power of s to increase its degree to $d-1$. Let us denote by $Q'_{\overline{v}}(z, (y_{l,j})_{l,j=1}^{t,m}, s)$ the resulting polynomial.

Next, consider the function $P_{\overline{v}} : D^m \times \mathbb{F}^{1+t\cdot m} \to \mathbb{F}$, which is constructed by plugging $\tilde{1}_{H^m}(\overline{x})$ to the variable s:

$$P_{\overline{v}}(\overline{x}, z, (y_{l,j})_{l,j=1}^{t,m}) = Q'_{\overline{v}}\left(z, (y_{l,j})_{l,j=1}^{t,m}, \tilde{1}_{H^m}(\overline{x})\right).$$

Observe that for every $\overline{v} \in H^m$, it holds that $P_{\overline{v}}(\overline{v}, z, (y_{l,j})_{l,j=1}^{t,m}) = Q_{\overline{v}}(z, (y_{l,j})_{l,j=1}^{t,m})$, since $\tilde{1}_{H^m}(\overline{v}) = 1$ for every such $\overline{v}$.

Finally, we define the function $Q^{(\psi)} : D^m \times \mathbb{F}^{1+t\cdot m} \to \mathbb{F}$ by

$$Q^{(\psi)}(\overline{x}, z, (y_{l,j})_{l=1,j=1}^{l,t}) \overset{\text{def}}{=} \sum_{\overline{u} \in H^m} \tilde{1}_{\overline{u}}(\overline{x}) \cdot P_{\overline{u}}(\overline{x}, z, (y_{l,j})_{l,j=1}^{t,m}).$$

We now establish the soundness of the reduction. Fix an codeword $g : D^m \to \mathbb{F}$ of $C^{\otimes m}$ and let $\overline{v} \in H^m$. We show that $f^{(\psi,g)}(\overline{v}) = Q_{\overline{v}}(g^{(\pi_1,\dots,\pi_t)}(\overline{v}))$. Note that this in particular implies that g

satisfies ψ if and only if it satisfies ψ_0. It holds that

$$\begin{aligned} f^{(\psi,g)}(\overline{v}) &= \sum_{\overline{u}\in H^m} \tilde{1}_{\overline{u}}(\overline{v}) \cdot P_{\overline{u}}(\overline{v}, g^{(\pi_1,\ldots,\pi_t)}(\overline{v})) \\ \text{(Since } \tilde{1}_{\overline{u}}(\overline{v}) = 0 \text{ for all } \overline{u} \in H^m - \{\overline{v}\}) &= P_{\overline{v}}(\overline{v}, g^{(\pi_1,\ldots,\pi_t)}(\overline{v})) \\ \text{(Since } P_{\overline{v}} = Q_{\overline{v}} \text{ on all } \overline{x} \in H^m) &= Q_{\overline{v}}(g^{(\pi_1,\ldots,\pi_t)}(\overline{v})), \end{aligned}$$

as required.

We turn to show that $Q^{(\psi)}$ satisfy the requirements of Definition 4.7. Fix an assignment $g : D^m \to \mathbb{F}$. We show that the function $f^{(\psi,g)}$ is a codeword of $(C_d)^{\otimes m}$. To this end, first note that by Proposition 6.6, it suffices to prove that $f^{(\psi,g)}$ is a codeword of $\left(C^{\otimes m}\right)^d$. Furthermore, it suffices to prove that each $P_{\overline{v}}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$ is a codeword of $\left(C^{\otimes m}\right)^{d-1}$, since this would imply that each

$$\tilde{1}_{\overline{u}}(\overline{x}) \cdot P_u(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$$

is a codeword of $\left(C^{\otimes m}\right)^d$ (being the coordinate-wise multiplication of codewords of $C^{\otimes m}$ and $\left(C^{\otimes m}\right)^{d-1}$, respectively). This, in turn, would imply that $f^{(\psi,g)}$ is a codeword of $(C_d)^{\otimes m}$ (being the sum of codewords of $\left(C^{\otimes m}\right)^d$).

We thus focus on proving that each $P_{\overline{v}}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$ is a codeword of $\left(C^{\otimes m}\right)^{d-1}$. It suffices to observe that $P_{\overline{v}}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$ is a sum of coordinate-wise multiplications of $(d-1)$ codewords of $C^{\otimes m}$. To see this, observe that $P_{\overline{v}}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$ is obtained from $Q'_{\overline{v}}$ by substituting the variables z, $y_{l,j}$ and s with $g(\overline{x})$, $g^{\pi,j}(\overline{x})$, and $\tilde{1}_{H^m}(\overline{x})$. Each of the latter functions are codewords of $C^{\otimes m}$ (the functions $g^{\pi,j}(\overline{x})$ are codewords of $C^{\otimes m}$ due to Proposition 6.7). Now, each monomial of $Q'_{\overline{v}}$ contributes to $P_{\overline{v}}$ a coordinate-wise multiplication of exactly $d-1$ of those functions, so each such monomial contributes a codeword of $\left(C^{\otimes m}\right)^{d-1}$. Finally, $P_{\overline{v}}$ is the sum of all those monomials, and therefore $P_{\overline{v}}(\overline{x}, g^{(\pi_1,\ldots,\pi_t)}(\overline{x}))$ is a codeword of $\left(C^{\otimes m}\right)^{d-1}$, as required.

It remains to show that $Q^{(\psi)}$ can be computed in polynomial time given the polynomials $Q_{\overline{v}}$, and this will imply that $Q^{(\psi)}$ can be represented by a small circuit. This follows from the fact that the codewords $\tilde{1}_{\overline{v}}$ and $\tilde{1}_{H_m}$ can be computed in time $\text{poly}(D^m)$ by using the generator matrix of $C^{\otimes m}$, which we can compute from the generator matrix of C (the rest of the analysis of the running time is obvious).

■

7 Combinatorial Nullstellensatz over algebraic curves

This section formally states and proves the Combinatorial Nullstellensatz over algebraic curves (AG Combinatorial Nullstellensatz). In our proof of the PCP theorem a special case of it (Theorem 3.16) is used to solve the "zero-testing" problem over tensors of AG codes.

7.1 Terms and notation

We start with a brief reminder of the important terms and notation from the theory of AG codes. We must stress that this section is only a reminder and will not serve as an introduction to AG codes for readers who aren't already familiar with them. We refer readers to [Sti93] for a full introduction to this area.

F/K is the **function field** F over the **base field** K. There is an associated projective, irreducible, non-singular algebraic curve $\mathcal{C}$.

$K(x)$ is the **rational function field** of K in the variable x. Its associated algebraic curve is the projective line $\mathbb{P}K$.

$\boldsymbol{P}_F$ is the set of **places** of F/K. Each place corresponds to a set of K-equivalent points on the algebraic curve $\mathcal{C}$. If K is algebraically closed, this set is of size 1, and we will use the terms point and place interchangeably.

$\mathcal{O}_P$ is the **local ring** of the **place** P.

F_P is the **residue field** of P.

v_P is the **discrete valuation** associated with the place P.

$f(P) \in F_P$ is the **evaluation** of the function f at the place P.

A function f is **regular** at a place P if $v_P(f) \geq 0$ (equivalently, if $f \in \mathcal{O}_P$).

t_P is a **local parameter of** P (i.e. $v_P(t_P) = 1$)

For a **divisor** G, $v_P(G)$ denotes the coefficient of P in G.

$\mathsf{Princ}\,(z)$ is the **principal divisor** of z.

$\mathsf{Poles}\,(z)$ is the **pole divisor** of z.

$\mathsf{Zeros}\,(z)$ is the **zero divisor** of z

$\mathsf{Support}(G)$ is the set of places P for which $v_P(G) \neq 0$.

$\mathcal{L}(G)$ is the **Riemann-Roch space** of the divisor G.

$\ell(G)$ is the dimension of the Riemann-Roch space associated with G.

$g(\mathcal{C})$ is the **genus** of the curve $\mathcal{C}$.

$\mathrm{Aut}(F/K)$ is the group of automorphisms of the function field.

$P'|P$ means that $P' \in \boldsymbol{P}_{F'}$ **lies over** $P \in \boldsymbol{P}_F$ in the function field extension F'/F

The **Riemann-Roch** Theorem states that for any divisor G:

- $\dim(\mathcal{L}(G)) \geq \deg(G) + 1 - g$.
- If $\deg(G) \geq 2g - 2$, then $\dim(\mathcal{L}(G)) = \deg(G) + 1 - g$.

Lemma 7.1 (Basic facts about divisors) *Let G, G' be divisors over a curve $\mathcal{C}$. We denote $G \geq G'$ if for each place P we have $v_P(G) \geq v_P(G')$. We denote by $G > G'$ if $G \geq G'$ and there is at least one place P on which $v_P(G) > v_P(G')$. Then*

1. *If $G \geq G'$ then $\mathcal{L}(G) \supseteq \mathcal{L}(G')$.*
2. *If f is a function and $\mathsf{Princ}(f) \geq -G$ then $f \in \mathcal{L}(G)$.*

7.2 Formal statements of Nullstellensätze

We start with an AG code defined by a projective nonsingular curve $\mathcal{C}$ over $\mathbb{F}_q$, a set D of $\mathbb{F}_q$-rational points and a positive divisor G whose support does not intersect D. The AG code $\mathsf{C} = C_{\mathcal{L}}(\mathcal{C}, D, G)$ consists of evaluations of functions in the Riemann-Roch space $\mathcal{L}(G)$ at the set of points D. The m-dimensional tensor code of C is naturally viewed as a set of "multivariate" algebraic functions defined on the m-dimensional variety $\mathcal{C}^m$, and evaluated at the set of points D^m. Let us first describe this interpretation concretely.

We will use the formal variables $X_1, \ldots, X_m$ to represent the m coordinates of a point of $\mathcal{C}^m$. Thus for a function $f \in \mathcal{L}(G)$, the function $f(X_i)$ represents a rational function on $\mathcal{C}^m$ which "ignores" all coordinates except the ith. For divisors $G_1, \ldots, G_m$ on $\mathcal{C}$, define the space $\mathcal{L}(G_1, \ldots, G_m)$ of multivariate algebraic functions on $\mathcal{C}^m$ as follows:

$$\mathcal{L}(G_1, \ldots, G_m) \stackrel{\text{def}}{=} \mathsf{span}\left\{ \prod_{i=1}^{m} f_i(X_i) \mid f_i \in \mathcal{L}(G_i) \right\}.$$

Note that $\mathcal{L}(G_1, \ldots, G_m)$ is isomorphic to the tensor product $\mathcal{L}(G_1) \otimes \ldots \otimes \mathcal{L}(G_m)$ (and thus the dimension of $\mathcal{L}(G_1, \ldots, G_m)$ equals $\prod_{i=1}^{m} \dim(\mathcal{L}(G_i))$).

Lemma 7.2 (Tensored AG codes) *Let $\mathsf{C}_i = C_{\mathcal{L}}(\mathcal{C}_i, D_i, G_i), i = 1, \ldots, m$ be m AG codes over $\mathbb{F}_q$. The codewords of the tensor product code $\bigotimes_{i=1}^{m} \mathsf{C}_i$ are the evaluations of functions in $\mathcal{L}(G_1, \ldots, G_m)$ over $\prod_i D_i$:*

$$\bigotimes_{i=1}^{m} \mathsf{C}_i = \left\{ \langle f(x_1, \ldots, x_m) \rangle_{(x_1, \ldots, x_m) \in \prod_{i=1}^{m} D_i} \mid f \in \mathcal{L}(G_1, \ldots, G_m) \right\}.$$

Proof: By induction on m. The base case $m = 1$ follows immediately from the definition of a tensor code. For $m > 1$ fix $x_m \in D_m$. Notice that if $f \in \mathcal{L}(G_1, \ldots, G_m)$ then $f(X_1, \ldots, X_{m-1}, x_m) \in \mathcal{L}(G_1, \ldots, G_{m-1})$, hence by induction

$$\langle f(x_1, \ldots, x_m) \rangle_{x_i \in D_i \text{ for } i=1,\ldots,m-1} \in \bigotimes_{i=1}^{m-1} \mathsf{C}_i.$$

Similarly, fixing $x_1, \ldots, x_{m-1}$ we see $f(x_1, \ldots, x_{m-1}, X_m) \in \mathcal{L}(G_m)$ implying that each row parallel to the mth axis belongs to C_m and this completes the proof. ■

For a divisor G we use $\mathcal{L}^m(G)$ to denote the space $\mathcal{L}(G, G, \ldots, G)$. To perform zero testing of tensored AG codes we will be interested in an algebraic characterization of functions $f(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$ that vanish on all the points of H^m, for some given subset $H \subseteq D$. As explained in the introduction, the problem we deal with is a generalization of Alon's Combinatorial Nullstellensatz (Theorem 3.15). When generalizing it to arbitrary AG codes over curves of strictly positive genus a certain problem emerges, one that does not appear in the case of genus-zero codes. There we may assume that for any $H \subset \mathbb{F}_q$ there exists a function $\xi_H(X)$ whose degree *equals* $|H|$ and which vanishes *precisely* on H with multiplicity 1. On a general curve the existence of such a ξ_H holds only for rather special sets H which we call *splitting sets*. For splitting sets we obtain the following analog of Theorem 3.15, which will be proved in section 7.4.

Theorem 7.3 (AG Combinatorial Nullstellensatz for splitting sets) *Let $\mathcal{C}$ be an algebraic curve over a field K, G be a divisor on it, and $H \subseteq \mathcal{C}$ be a set of K-rational points, disjoint from* $\mathsf{Support}(G)$, *satisfying* $\deg G \geq |H| + 2g(\mathcal{C}) - 1$. *Suppose H is such that there exists $\xi(X) = \xi_H(X) \in \mathcal{L}(G)$ satisfying (1) $\xi(x) = 0$ for each $x \in H$, and (2)* $\deg \mathsf{Zeros}(\xi(X)) = |H|$. *(Thus $H = \mathsf{Support}(\mathsf{Zeros}(\xi))$, and ξ has only zeros of multiplicity 1.) Let $f(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$. Then*

f vanishes at each point of H^m if and only if for each $i \in [m]$ there exists $f'_i(X_1, \dots, X_m) \in \mathcal{L}^m(G)$ such that

$$f(X_1, \dots, X_m) = \sum_{i=1}^{m} f'_i(X_1, \dots, X_m)\xi(X_i). \tag{12}$$

When H does not have this special property, this form of Combinatorial Nullstellensatz cannot hold. Instead, we prove a generalization of the Combinatorial Nullstellensatz that gives an algebraic characterization of vanishing functions. It holds for any set H over any curve $\mathcal{C}$ but requires *two* auxiliary functions ξ_H and ξ'_H.

Theorem 7.4 (AG Combinatorial Nullstellensatz for arbitrary sets) *Let $\mathcal{C}$ be an algebraic curve over a perfect field K, G be a K-divisor on it, and $H \subseteq \mathcal{C}$ be a set of K-rational points, disjoint from $\mathsf{Support}(G)$, satisfying $\deg G \geq |H| + 2g(\mathcal{C}) - 1$. Then there exists $\xi(X) = \xi_H(X) \in \mathcal{L}(2G)$, $\xi'(X) = \xi'_H(X) \in \mathcal{L}(3G)$ satisfying the following. Suppose $f(X_1, \dots, X_m) \in \mathcal{L}^m(G)$. Then f vanishes on H^m if and only if there exist $f'_1, \dots, f'_m \in \mathcal{L}^m(4G)$ such that*

$$f(X_1, \dots, X_m) \cdot \prod_{i=1}^{m} \xi'(X_i) = \sum_{i=1}^{m} f'_i(X_1, \dots, X_m) \cdot \xi(X_i). \tag{13}$$

Remark 7.5 *The codes described in the appendix do contain splitting sets. As $x_0 - a$, when $\mathsf{Trace}(a) \neq 0$, has exactly $\frac{n}{l(l-1)}$ zeros (where n is the length of the code over $\mathbb{F}_{l^2}$), all of multiplicity 1 and on rational points, and those are its only zeros. Bigger sets can be gotten by looking at the zeros of products of such functions.*

This means that Lemma 7.3 is sufficient for our purposes. However, Theorem 7.4 allows for substituting any transitive AG code and using any set H in the PCP construction, and furthermore we believe the extended theorem and its proof (notably, theorem 7.10) to be of independent interest.

7.3 Instantiating parameters

We now show how to instantiate parameters into Theorem A.14 (and Lemma A.17) to get Theorem 3.13, and then how to use Theorem 7.3 to get Theorem 3.16.

Proof of Theorem 3.13: Let $\varepsilon > 0$ be a constant (independent of k) such that:

$$c_q \cdot d_{\mathsf{mult}} \cdot \rho + \delta < 1 - \varepsilon - \frac{d_{\mathsf{mult}}}{\sqrt{q} - 1}.$$

Theorem A.14 and Lemma A.17 give us, for any n', an AG curve $\mathcal{C}$ with a special set of $\mathbb{F}_q$-rational points D whose cardinality $\in [n', c_q \cdot n']$. Choose:

$$n' = \frac{d_{\mathsf{mult}} \cdot k}{1 - \varepsilon - \delta - \frac{d_{\mathsf{mult}}}{\sqrt{q}-1}},$$

and consider the AG curve given as above. From Theorem A.14 and Lemma A.17, we have the following properties.

1. There is a group Γ of automorphisms of $\mathcal{C}$ which acts transitively on D.

2. $n \overset{\text{def}}{=} |D| \le n' \cdot c_q < k \cdot \frac{1}{\rho}$.

3. $\mathcal{C}$ has genus at most $\frac{n}{\sqrt{q}-1}$.

4. There is a divisor G^* on $\mathcal{C}$, invariant under the action of Γ, with:

   (a)
   $$\frac{\deg(G^*)}{n} \in \left[\frac{1-\varepsilon-\delta}{d_{\mathsf{mult}}}, \frac{1-\delta}{d_{\mathsf{mult}}}\right],$$
   (this follows by taking G^* to be a suitable multiple of the divisor A_i from Equation (32), for k sufficiently large)

   (b) The support of G^* is disjoint from D.

   (c) If we let C be the AG code $C_{\mathcal{L}}(\mathcal{C}, D, G^*)$, then C is an AG code of length n, invariant under the action of Γ.

5. For $j \in [d_{\mathsf{mult}}]$, we define C_j to be the AG code $C_{\mathcal{L}}(\mathcal{C}, D, jG^*)$. Then $\vec{C} = (C_1, \ldots, C_{d_{\mathsf{mult}}})$ is a multiplication code family. Furthermore, each C_j is invariant under the action of Γ.

6. For $j \in [d_{\mathsf{mult}}]$, C_j has distance at least
   $$1 - \frac{\deg(jG^*)}{n} \ge \delta.$$

7. By the Riemann-Roch theorem,
   $$\begin{aligned}\dim(C) &\ge \deg(G^*) - \frac{n}{\sqrt{q}-1}\\ &\ge \left(\frac{1-\varepsilon-\delta}{d_{\mathsf{mult}}} - \frac{1}{\sqrt{q}-1}\right) \cdot n\\ &\ge \left(\frac{1-\varepsilon-\delta}{d_{\mathsf{mult}}} - \frac{1}{\sqrt{q}-1}\right) \cdot \frac{d_{\mathsf{mult}} \cdot k}{1-\varepsilon-\delta-\frac{d_{\mathsf{mult}}}{\sqrt{q}-1}}\\ &\ge k.\end{aligned}$$

This concludes the proof of Theorem 3.13. ■

Now we show that these codes also satisfy the conclusion of Theorem 3.16.

Proof of Theorem 3.16: We keep the notation of the proof of Theorem 3.13. We take the code given by Theorem 3.13 with the parameter $\varepsilon = \frac{1}{2}$. We simply have to check that the hypotheses of Theorem 7.4 are satisfied.

Put $G = dG^*$. We have $C = C_{\mathcal{L}}(\mathcal{C}, D, G^*)$, and $C_d = C_{\mathcal{L}}(\mathcal{C}, D, G)$.

The hypothesis of Theorem 3.16 has a set $H \subseteq D$ with
$$|H| \le \left(\frac{1}{6}\left(\frac{1}{2} - \delta\right) - \frac{2}{\sqrt{q}-1}\right) \cdot n.$$

This implies that

$$\begin{aligned}|H| &\leq \frac{d}{d_{\mathsf{mult}}}\left(\frac{1}{2}-\delta\right)n - \frac{2n}{\sqrt{q}-1} \\ &= \frac{d}{d_{\mathsf{mult}}}\left(1-\varepsilon-\delta\right)n - \frac{2n}{\sqrt{q}-1} \\ &\leq \deg(G) - 2g(\mathcal{C}).\end{aligned}$$

Thus we may apply Theorem 3.16. This gives us functions $\xi_H \in \mathcal{L}(2G)$ and $\xi'_H \in \mathcal{L}(3G)$, which we may view as elements of C_{2d} and C_{3d} respectively.

Now take an element $f(X_1,\ldots,X_m) \in (C_d)^{\otimes m}$, which we view as an element of $\mathcal{L}^m(G)$. By Theorem 7.4, f vanishes on H^m if and only if there exists $f'_1,\ldots,f'_m \in \mathcal{L}^m(4G)$ (which we view as elements of $(C_{4d})^{\otimes m}$) such that:

$$f(X_1,\ldots,X_m)\cdot\prod_{i=1}^{m}\xi'(X_i) = \sum_{i=1}^{m} f'_i(X_1,\ldots,X_m)\cdot\xi(X_i).$$

This implies the desired result. ■

7.4 Proof of the AG Combinatorial Nullstellensatz for splitting sets

We begin with a lemma on interpolation. It is a special case of the upcoming Lemma 7.11, and we omit the proof.

Lemma 7.6 (Interpolation) *Let $\mathcal{C}$ be a curve over K and let G be a K-divisor on $\mathcal{C}$. For every set of K-rational points $H \subseteq \mathcal{C}$ disjoint from $\mathsf{Support}(G)$ and function $a : H \to K$, if $\deg(G) \geq |H| + 2g - 1$, there exists $f(X) \in \mathcal{L}(G)$ with $f(x) = a(x)$ for each $x \in H$.*

We now prove the AG Combinatorial Nullstellensatz for splitting sets.

Proof of Theorem 7.3: We start with the easier part of the implication: Suppose $f(X_1,\ldots,X_m) = \sum_{i=1}^{m} f'_i(X_1,\ldots,X_m)\xi(X_i)$. If $(x_1,\ldots,x_m) \in H^m$, then we have that $\xi(x_i) = 0$ for each $i \in [m]$, and so each term in $f(x_1,\ldots,x_m)$ vanishes on $(x_1,\ldots,x_m) \in H^m$ and so f too vanishes on H^m.

The other direction of the implication is proved by induction on m.

For the base case of $m = 1$ suppose $f(x_1) = 0$ for each $x_1 \in H$. We take $f'_1(X_1) = \frac{f(X_1)}{\xi(X_1)}$, clearly (12) holds so we only have to prove that $f'_1 \in \mathcal{L}(G)$. In other words, we have to show

$$\mathsf{Princ}(f') + G \geq 0.$$

By definition

$$\mathsf{Princ}(f') = \mathsf{Princ}\left(\frac{f(X_1)}{\xi(X_1)}\right) = \mathsf{Princ}\left(f(X_1)\right) - \mathsf{Princ}\left(\xi(X_1)\right) \geq \mathsf{Princ}(f) - \mathsf{Zeros}(\xi).$$

By assumption the support of $\mathsf{Zeros}(\xi)$ is H and furthermore $v_{x_1}(\xi) = 1$ for every place $x_1 \in H$. By assumption f vanishes on H so $v_{x_1}(f) \geq v_{x_1}(\xi)$ for every $x_1 \in H = \mathrm{Supp}(\mathsf{Zeros}(\xi))$. So $\frac{f(X_1)}{\xi(X_1)}$ has no poles on the zeros of $\xi(X_1)$ and this implies

$$\mathsf{Princ}(f) - \mathsf{Zeros}(\xi) \geq \mathsf{Poles}(f).$$

By assumption $\mathsf{Poles}(f) + G \geq 0$ so we conclude $\mathsf{Princ}(f') + G \geq 0$ as required.

For the inductive case of $m > 1$ suppose $f(x_1, \ldots, x_m) = 0$ for each $(x_1, \ldots, x_m) \in H^m$. For every $x \in H$, we have that $f(X_1, \ldots X_{m-1}, x)$ vanishes on H^{m-1}. So by induction,

$$f_x \stackrel{\text{def}}{=} f(X_1, \ldots, X_{m-1}, x) = \sum_{i=1}^{m-1} f'_{x,i}(X_1, \ldots, X_{m-1})\xi(X_i),$$

with $f'_{x,i}(X_1, \ldots, X_{m-1}) \in \mathcal{L}^{m-1}(G)$.

Recalling $\deg G \geq |H| + 2g(\mathcal{C}) - 1$, Lemma 7.6 implies the existence of a function $\delta_x(X_m) \in \mathcal{L}(G)$ satisfying

$$\delta_x(y) = \begin{cases} 1 & y = x \\ 0 & y \in H \setminus \{x\} \end{cases}$$

Define

$$h(X_1, \ldots, X_m) = f(X_1, \ldots, X_m) - \sum_{x \in H} \delta_x(X_m) f_x(X_1, \ldots, X_{m-1}).$$

Note that

$$\forall x \in H, \quad h(X_1, \ldots, X_{m-1}, x) \equiv 0. \tag{14}$$

Since $f \in \mathcal{L}^m(G)$ and $\delta_x(X_m) f_x(X_1, \ldots, X_{m-1}) \in \mathcal{L}^m(G)$, we have $h(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$. Let $\beta_1(X_1, \ldots, X_{m-1}), \ldots, \beta_\ell(X_1, \ldots, X_{m-1})$ be a basis for $\mathcal{L}^{m-1}(G)$. Then we can write

$$h(X_1, \ldots, X_m) \equiv \sum_{i=1}^{\ell} \beta_i(X_1, \ldots, X_{m-1}) \cdot a_i(X_m), \tag{15}$$

for some $a_i(X_m) \in \mathcal{L}(G)$.

Evaluating X_m at $x \in H$ we get from (14) that $a_i(x) = 0$ for each $i \in [\ell], x \in H$. By the $m = 1$ case, we have that

$$a_i(X_m) \equiv b_i(X_m) \cdot \xi(X_m), \text{ for some } b_i(X_m) \in \mathcal{L}(G) \tag{16}$$

Define

$$f'_m(X_1, \ldots, X_m) \equiv \sum_{i=1}^{\ell} \beta_i(X_1, \ldots, X_{m-1}) \cdot b_i(X_m),$$

and notice that (15) and (16) imply $h(X_1, \ldots, X_m) \equiv f'_m(X_1, \ldots, X_m) \cdot \xi(X_m)$.

Thus

$$f(X_1, \ldots, X_m) \equiv \sum_{x \in H} f_x(X_1, \ldots, X_{m-1}) \delta_x(X_m) + h(X_1, \ldots, X_m)$$

which can be rewritten as

$$f(X_1, \ldots, X_m) \equiv \sum_{i=1}^{m-1} \left(\sum_{x \in H} f'_{x,i}(X_1, \ldots, X_{m-1}) \delta_x(X_m) \right) \cdot \xi(X_i) + f'_m(X_1, \ldots, X_m) \cdot \xi(X_m),$$

thereby completing the proof. ■

7.5 Local Derivatives and multiplicities

Our strategy for proving the AG Combinatorial Nullstellensatz for general sets is as follows. Recall that we want to show the existence of functions $\xi, \xi' \in \mathcal{L}(G)$ such that if $f(X_1, \ldots, X_m)$ vanishes on all the points of H^m, then there exist $f_1'(X_1, \ldots, X_m), f_2'(X_1, \ldots, X_m), \ldots, f_m'(X_1, \ldots, X_m)$ such that

$$f(X_1, \ldots, X_m) \cdot \prod_{j=1}^{m} \xi'(X_i) = \sum_{i=1}^{m} f_i'(X_1, \ldots, X_m) \cdot \xi(X_i).$$

Even though the given H may not be a splitting set, we can still find a low degree function ξ (of degree not much more than $|H|$) that vanishes on H. For the moment let us assume that all the zeros of ξ are of multiplicity 1. Let J be the zeros of ξ not in H. Then we take ξ' to be a function that vanishes at all the points of J but not at any of the points of H. Notice $f(X_1, \ldots, X_m)$ vanishes on H^m iff the function $f(X_1, \ldots, X_m) \cdot \prod_{i=1}^{m} \xi'(X_i)$ vanishes on $(H \cup J)^m$. Assuming $H \cup J$ is a splitting set (it is the set of zeros of ξ), we may apply the AG Combinatorial Nullstellensatz for splitting sets (Theorem 7.3) and complete the proof.

In general, however, ξ may vanish at some points with multiplicity > 1. Because of this possibility, we will need to show a version of the AG Combinatorial Nullstellensatz for splitting sets with multiplicities. This will require us to use several properties of derivatives, multiplicities and power-series representations of algebraic functions on curves and products of curves. We briefly review these notions next. In the next subsection we will use these to prove the AG Combinatorial Nullstellensatz for splitting sets with multiplicities. Finally we use this to prove the AG Combinatorial Nullstellensatz for general sets.

For the rest of this section we assume that we are working with an algebraic curve $\mathcal{C}$ over an algebraically closed base field K. Our strategy will be to first prove the AG Combinatorial Nullstellensatz for splitting sets with multiplicities and the AG Combinatorial Nullstellensatz for general sets over algebraically closed fields. Finally we will use a linear algebra argument to deduce the AG Combinatorial Nullstellensatz for general sets over all fields (and in particular, over finite fields, which we need for the proof of the main PCP Theorem 2.3).

Let x be a point on $\mathcal{C}$ and let t_x be a local parameter for x. Every rational function f on $\mathcal{C}$ which is regular at x can be written as $f = \sum_{i=0}^{\infty} a_i t_x^i$, with each $a_i \in K$, where the meaning of this equality is that for every $k \geq 0$, we have $v_x(f - \sum_{i=0}^{k} a_i t_x^i) > k$ (equivalently, this equality holds in $\hat{\mathcal{O}}_x$, the *completion* of the local ring $\mathcal{O}_x$).

Having fixed a local parameter, we now define local derivatives with respect to that local parameter. Suppose f is regular at x. Let $f = \sum_{i=0}^{\infty} a_i t_x^i$ be the power series representation of f at x. Then we define the ith *local Hasse derivative of* f *at* x *w.r.t.* t_x, denoted $f^{(i)}(x)$, to equal a_i (see also [GV87]). Note that $f^{(0)}(x)$ is simply $f(x)$, and is independent of the choice of the local parameter. On the other hand, for $i \geq 1$, $f^{(i)}(x)$ depends on the choice of the local parameter. We say that f vanishes at x with multiplicity $\geq e$ if $f^{(i)}(x) = 0$ for each $i < e$. This is equivalent to $v_x(f) \geq e$, and thus the property of vanishing at x with multiplicity $\geq e$ is independent of the choice of local parameter.

Remark 7.7 *Notice that $f^{(i)}$ is not a rational function! Suitably defined, it is an "i-fold differential form", and it cannot be assigned a value at a point x without a choice of a local parameter t_x. In our arguments we will never treat $f^{(i)}$ as an element of the function field, i.e., as a "global" function on $\mathcal{C}$, we will only look at $f^{(i)}(x)$ at a specific place x where we have already fixed a choice of local parameter t_x.*

Now we deal with multivariate functions. $\mathcal{C}^m$ is the m-fold product variety of the curve $\mathcal{C}$.

Let $(x_1, \ldots, x_m)$ be a point on $\mathcal{C}^m$. Every rational function f on $\mathcal{C}^m$ which is regular at $(x_1, \ldots, x_m)$ can be written as:

$$f(X_1, \ldots, X_m) = \sum_{\mathbf{i} \in \mathbb{Z}^m, \mathbf{i} \geq 0} a_{\mathbf{i}} \prod_{j=1}^{m} t_{x_j}^{i_j}(X_j),$$

with $a_{\mathbf{i}} \in K$ (again this equality is in the completion $\hat{\mathcal{O}}_{(x_1,\ldots,x_m)}$ of the local ring $\mathcal{O}_{(x_1,\ldots,x_m)}$). The completion is isomorphic to the formal power series field in m variables $K[[Y_1, \ldots, Y_m]]$. Then we define the $\mathbf{i} \in \mathbb{Z}^m$ local derivative of f at the point $(x_1, \ldots, x_m)$ with respect to $(t_{x_1}, \ldots, t_{x_m})$ by:

$$f^{(i_1,\ldots,i_m)}(x_1, \ldots, x_m) = a_{(i_1,\ldots,i_m)}.$$

We say $f(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity $\geq (e_1, \ldots, e_m)$ if for every nonzero $a_{\mathbf{i}}$ in the above representation, there exists some j for which $i_j \geq e_j$. Note that (1) if some $e_j = 0$, the condition that $f(X_1, \ldots, X_m)$ vanishes at some point with individual multiplicity $\geq (e_1, \ldots, e_m)$ is vacuous, and (2) this definition does not allow us to speak about *the* individual multiplicity with which f vanishes at $(x_1, \ldots, x_m)$, but only about the relation "individual multiplicity $\geq (e_1, \ldots, e_m)$". We next show that the statement "individual multiplicity $\geq (e_1, \ldots, e_m)$" is independent of choice of local parameters.

Theorem 7.8 (Consistency of individual multiplicity) *For $f \in \mathcal{L}^m(G)$ and $(e_1, \ldots, e_m)$, the property of f vanishing at $(x_1, \ldots, x_m)$ with individual multiplicity $\geq (e_1, \ldots, e_m)$ is independent of the choice of local parameters chosen at $x_1, \ldots, x_m$.*

Proof: Represent f as a power series in the local parameters $t_1, t_2, \ldots, t_m$ at $x_1, \ldots, x_m$. We want to shift from this representation to a power series in $t'_1, t'_2, \ldots, t'_m$. To do this we utilize the fact that $t_i = t'_i \cdot u_i$ for some u_i with $v_{x_i}(u_i) = 0$, and get a power series in $t'_1, t'_2, \ldots, t'_m$ where the coefficient to $\mathbf{t_i}'$ is $a_{\mathbf{i}}\mathbf{u}^{\mathbf{i}}$. We then represent the $u'_i s$ as power series in $t'_1, t'_2, \ldots, t'_m$ (note that they have a non-zero free coefficient) and get that any minimal term with a non-zero coefficient still has a non-zero coefficient and added terms can only be greater (in at least one of the powers) than already existing terms with non-zero coefficients. Thus individual multiplicity does not depend on the choice of local parameters. ■

Recall from Remark 7.7 that derivatives of a rational function are not rational functions. An important exception is when we take the derivative of a function with respect to a subset of variables $X_1, \ldots, X_{m'}$ at the point $(x_1, \ldots, x_{m'})$; In this case, the resulting object is a rational function in the remaining variables $(X_{m'+1}, \ldots, X_m)$. Suppose $x_1, \ldots, x_{m'} \in \mathcal{C}$. Write:

$$f(X_1, \ldots, X_m) = \sum_{\mathbf{i} \in \mathbb{Z}^{m'}, \mathbf{i} \geq 0} a_{\mathbf{i}}(X_{m'+1}, \ldots, X_m) \prod_{j=1}^{m'} t_{x_j}^{i_j}(X_j).$$

Then we define:

$$f^{(i_1,\ldots,i_{m'},0,\ldots,0)}(x_1, \ldots, x_{m'}, X_{m'+1}, \ldots, X_m) = a_{(i_1,\ldots,i_{m'})}(X_{m'+1}, \ldots, X_m).$$

(This slight overload of notation should not cause confusion in our future use of it.)

Taking derivatives in one set of variables, followed by derivatives in another set of variables, is equivalent to taking derivatives in both sets of variables in one shot.

Concretely, let $m = m_1 + m_2 + m_3$, and rename the variables $X_1, \ldots, X_m$ as $A_1, \ldots, A_{m_1}, B_1, \ldots, B_{m_2}, C_1, \ldots C_{m_3}$. Suppose $f(A_1, \ldots, A_{m_1}, B_1, \ldots, B_{m_2}, C_1, \ldots C_{m_3})$ is a regular function at $(a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_2}, c_1, \ldots, c_{m_3}) \in \mathcal{C}^m$. Let

$$g(B_1, \ldots, B_{m_2}, C_1, \ldots, C_{m_3}) = f^{(i_1, \ldots, i_{m_1}, 0, \ldots, 0)}(a_1, \ldots, a_{m_1}, B_1, \ldots, B_{m_2}, C_1, \ldots, C_{m_3}),$$

$$h(C_1, \ldots, C_{m_3}) = g^{(j_1, \ldots, j_{m_2}, 0, \ldots, 0)}(b_1, \ldots, b_{m_2}, C_1, \ldots, C_{m_3}).$$

Then:

$$h(C_1, \ldots, C_{m_3}) = f^{(i_1, \ldots, i_{m_1}, j_1, \ldots, j_{m_2}, 0, \ldots, 0)}(a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_2}, C_1, \ldots, C_{m_3}).$$

The following consequence of this discussion will be used later on.

Proposition 7.9 *If f vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity at least $(e_1, \ldots, e_m)$, then for every $j < e_m$, $f^{(0,0,\ldots,j)}(X_1, \ldots, X_{m-1}, x_m)$ vanishes at $(x_1, \ldots, x_{m-1})$ with individual multiplicity at least $(e_1, \ldots, e_{m-1})$.*

7.6 AG Combinatorial Nullstellensatz for splitting sets with multiplicity

We now state the AG Combinatorial Nullstellensatz for splitting sets with multiplicity. It extends the statement of the AG Combinatorial Nullstellensatz for splitting sets (without multiplicity) to the case where the function ξ may have zeros of multiplicity > 1. The equivalent result for polynomials was shown in [KR12].

Theorem 7.10 (AG combinatorial Nullstellensatz for splitting sets with multiplicity) *Let $\mathcal{C}$ be an algebraic curve over a field K, G be a divisor on it, and D be a divisor whose support is disjoint from the support of G, with $\deg(G) \geq \deg(D) + 2g - 1$. Suppose there exists $\xi(X) \in \mathcal{L}(G)$ such that $D = \mathsf{Zeros}(\xi(X))$. Let $f(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$.*

Then $f(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity at least $(v_{x_1}(D), v_{x_2}(D), \ldots, v_{x_m}(D))$ for each $(x_1, \ldots, x_m) \in \mathsf{Support}(D)^m$ iff:

$$\exists f'_1(X_1, \ldots, X_m), \ldots, f'_m(X_1, \ldots, X_m) \in \mathcal{L}^m(G) \text{ such that:}$$
$$f(X_1, \ldots, X_m) = \sum_{i=1}^{m} f'_i(X_1, \ldots, X_m) \cdot \xi(X_i).$$

As in the case of Theorem 7.3, we will need a lemma on interpolation, this time with multiplicity.

Lemma 7.11 (Interpolation with multiplicity) *Let G, D be divisors such that $\deg G \geq \deg D + 2g(\mathcal{C}) - 1$, $D \geq 0$ and $\mathsf{Support}(D) \cap \mathsf{Support}(G) = \emptyset$. Suppose we are given, for each $x \in \mathsf{Support}(D)$ and each $j < v_x(D)$ an element $c_{x,j} \in K$. Then there exists $f(X) \in \mathcal{L}(G)$ such that for each $x \in \mathsf{Support}(D)$ and each $j < v_x(D)$, we have $f^{(j)}(x) = c_{x,j}$.*

Proof: Consider the mapping $\varphi : \mathcal{L}(G) \to \prod_{x \in \mathsf{Support}(D)} K^{v_x(D)}$ defined by

$$\varphi(f) = \left(\left(f_x^{(i)} \right)_{i=0}^{v_x(D)-1} \right)_{x \in \mathsf{Support}(D)}$$

i.e., the mapping that takes f to its first $v_x(D)$ derivatives at each $x \in \mathsf{Support}(D)$.

The dimension of $\mathcal{L}(G)$ equals $\deg(G) + 1 - g$, by the Riemann-Roch theorem.

The mapping φ is linear and has kernel $\mathcal{L}(G - D)$, so by the Riemann-Roch theorem, the dimension of the kernel is $\deg(G - D) + 1 - g$. So the dimension of the image of φ is exactly $\deg D$, which equals the dimension (as a vector space over K) of the space $\prod_{x \in \mathsf{Support}(D)} F_x^{v_x(D)}$.

Thus φ is surjective, and the result follows. ■

We now prove Theorem 7.10. The proof closely follows the proof in the case of splitting sets without multiplicity (Theorem 7.4).

Proof of Theorem 7.10: First suppose there exist $f_1'(X_1, \ldots, X_m), \ldots, f_m'(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$ such that:

$$f(X_1, \ldots, X_m) = \sum_{i=1}^{m} f_i'(X_1, \ldots, X_m) \cdot \xi(X_i).$$

Notice that the right hand side vanishes at each $(x_1, \ldots, x_m) \in \mathsf{Support}(D)^m$ with individual multiplicity at least $(v_{x_1}(\xi(X_1)), \ldots, v_{x_m}(\xi(X_m))) = (v_{x_1}(D), \ldots, v_{x_m}(D))$. Thus so does the left hand side, as desired.

For $m = 1$: Suppose $f(X_1)$ vanishes at each x_1 with multiplicity at least $v_{x_1}(D) = v_{x_1}(\xi(X_1))$. We have $\mathsf{Princ}\left(\frac{f(X_1)}{\xi(X_1)}\right) = \mathsf{Princ}\,(f(X_1)) - \mathsf{Princ}\,(\xi(X_1))$. So $\frac{f(X_1)}{\xi(X_1)} \in \mathcal{L}(G + \mathsf{Princ}(\xi)) = \mathcal{L}\,(G + \mathsf{Zeros}\,(\xi) - \mathsf{Poles}\,(\xi))$. But our hypothesis implies that $f(X_1)$ vanishes whenever $\xi(X_1)$ vanishes (and with at least as high multiplicity), and so $\frac{f(X_1)}{\xi(X_1)}$ has no poles on the zeros of $\xi(X_1)$. So $\frac{f(X_1)}{\xi(X_1)} \in \mathcal{L}(G - \mathsf{Poles}\,(\xi)) \subseteq \mathcal{L}(G)$. Thus we may take $f_1'(X_1) = \frac{f(X_1)}{\xi(X_1)}$.

For general m suppose $f(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$ vanishes at each $(x_1, \ldots, x_m) \in \mathcal{C}^m$ with individual multiplicity at least $(v_{x_1}(D), v_{x_2}(D), \ldots, v_{x_m}(D))$.

For every $x \in \mathcal{C}$ and every $j < v_x(D)$, we deduce from Proposition 7.9 that $f^{(0,0,\ldots,0,j)}(X_1, \ldots X_{m-1}, x)$ vanishes at each $(x_1, \ldots, x_{m-1}) \in \mathcal{C}^{m-1}$ with individual multiplicity at least $(v_{x_1}(D), \ldots, v_{x_{m-1}}(D))$. So by induction,

$$f_{x,j} \stackrel{\text{def}}{=} f^{(0,0,\ldots,0,j)}(X_1, \ldots, X_{m-1}, x) = \sum_{i=1}^{m-1} f'_{x,j,i}(X_1, \ldots, X_{m-1})\xi(X_i), \tag{17}$$

with $f'_{x,j,i}(X_1, \ldots, X_{m-1}) \in \mathcal{L}^{m-1}(G)$.

For each $x \in \mathsf{Support}(D)$ and each $j < v_x(D)$, let $\delta_{x,j}(X_m) \in \mathcal{L}(G)$ be such that for each $y \in \mathsf{Support}(D)$ and each $j' < v_y(D)$:

$$\delta_{x,j}^{(j')}(y) = \begin{cases} 1 & y = x \text{ and } j' = j \\ 0 & \text{otherwise} \end{cases}$$

Such a $\delta_{x,j}$ exists by Lemma 7.11, since $\deg G \geq \deg D + 2g(\mathcal{C}) - 1$. Define

$$h(X_1, \ldots, X_m) = f(X_1, \ldots, X_m) - \sum_{x \in \mathsf{Support}(D)} \sum_{j < v_x(D)} \delta_{x,j}(X_m) f_{x,j}(X_1, \ldots, X_{m-1}). \tag{18}$$

Note that $\forall x \in \mathsf{Support}(D), j < v_x(D)$, we have $h^{(0,0,\ldots,0,j)}(X_1,\ldots,X_{m-1},x) = 0$ (formally, as an element of $\mathcal{L}^{m-1}(G)$). Since $f \in \mathcal{L}^m(G)$ and $\delta_{x,j}(X_m)f_{x,j}(X_1,\ldots,X_{m-1}) \in \mathcal{L}^m(G)$, we have $h(X_1,\ldots,X_m) \in \mathcal{L}^m(G)$, Let $\beta_1(X_1,\ldots,X_{m-1}),\ldots,\beta_\ell(X_1,\ldots,X_{m-1})$ be a basis for $\mathcal{L}^{m-1}(G)$. Then we can write

$$h(X_1,\ldots,X_m) = \sum_{i=1}^{\ell} \beta_i(X_1,\ldots,X_{m-1}) \cdot a_i(X_m) \tag{19}$$

for some $a_i(X_m) \in \mathcal{L}(G)$.

Then for each $x \in \mathsf{Support}(D)$, $j < v_x(D)$, we have:

$$0 = h^{(0,0,\ldots,0,j)}(X_1,\ldots,X_{m-1},x) = \sum_{i=1}^{\ell} \beta_i(X_1,\ldots,X_{m-1}) \cdot a_i^{(j)}(x).$$

Therefore for each i, x, j we have $a_i^{(j)}(x) = 0$. Thus for each $i \in [\ell]$, we have that $a_i(X)$ vanishes at each x with multiplicity at least $v_D(x)$. By the $m = 1$ case, we have that $a_i(X_m) = b_i(X_m) \cdot \xi(X_m)$ for some $b_i(X_m) \in \mathcal{L}(G)$.

Define

$$f'_m(X_1,\ldots,X_m) = \sum_{i=1}^{\ell} \beta_i(X_1,\ldots,X_{m-1}) \cdot b_i(X_m), \tag{20}$$

and notice that $h(X_1,\ldots,X_m) = f'_m(X_1,\ldots,X_m) \cdot \xi(X_m)$.

Thus from (17) and (18) we get

$$f(X_1,\ldots,X_m) = \sum_{x\in\mathsf{Support}(D)} \sum_{j<v_D(x)} f_{x,j}(X_1,\ldots,X_{m-1})\delta_{x,j}(X_m) + h(X_1,\ldots,X_m)$$

and using (19) and (20) we conclude

$$f(X_1,\ldots,X_m) = \sum_{i=1}^{m-1} \left(\sum_{x\in\mathsf{Support}(D)} \sum_{j<v_D(x)} f'_{x,j,i}(X_1,\ldots,X_{m-1})\delta_{x,j}(X_m) \right) \cdot \xi(X_i) + f'_m(X_1,\ldots,X_m) \cdot \xi(X_m),$$

as desired. ∎

Remark 7.12 *A virtually identical proof would work for the case where we have m different ξ's with m different zero divisors, one for each variable. Additionally, a more accurate accounting of degree bounds through the proof would give a tighter bound on the coefficients f_i, specifically: $f_i\xi(X_i) \in \mathcal{L}^m(G)$.*

7.7 Proof of the AG Combinatorial Nullstellensatz for general sets

In this section we'll prove the AG Combinatorial Nullstellensatz for general sets, Theorem 7.4. We start by proving an additional interpolation lemma.

Lemma 7.13 (Interpolation with a nonzeroness condition) *Let G be a nonnegative divisor and H a set of points satisfying $H \cap \mathsf{Support}(G) = \emptyset$ and $\deg G \geq |H| + 2g(\mathcal{C}) - 1$. Then there exists $\xi \in \mathcal{L}(2G)$ such that $\xi|_H = 0$ and $\mathsf{Support}(\mathsf{Zeros}(\xi)) \cap \mathsf{Support}(G) = \emptyset$.*

Proof: Suppose $G = \sum_{i=1}^{k} n_i P_i$. For each $i \in [k]$, define $G_i = G + P_i$.
By the Riemann-Roch theorem, we have:

$$\dim(\mathcal{L}(G - H)) = \deg(G) - |H| + 1 - g,$$

and for each $i \in [k]$, we have:

$$\dim(\mathcal{L}(G_i - H)) = \deg(G_i) - |H| + 1 - g.$$

Thus for each i, we have $\dim(\mathcal{L}(G_i - H)) > \dim(\mathcal{L}(G - H))$. Let $\xi_i \in \mathcal{L}(G_i - H) \setminus \mathcal{L}(G - H)$. Note that each ξ_i vanishes on H, and that for each i, ξ_i must have a pole at P_i of order $n_i + 1$.

Now consider $\xi = \sum_{i=1}^{k} \xi_i$. First notice that ξ vanishes on H (since each ξ_i does). Next notice that ξ has a pole of order exactly $n_i + 1$ at each P_i (because ξ_i has such a pole, and all the ξ_j with $j \neq i$ have poles of order $\leq n_i$ at P_i). Finally notice that $\xi \in \mathcal{L}(G + \sum_{i=1}^{k} P_i) \subseteq \mathcal{L}(2G)$. ■

Proof of Theorem 7.4: We begin by proving the theorem over an algebraically closed field K.
Let $\xi \in \mathcal{L}(2G)$ be given by Lemma 7.13 such that $\xi(x) = 0$ for each $x \in H$ and $\mathsf{Support}(\mathsf{Zeros}(\xi)) \cap \mathsf{Support}(G) = \emptyset$. Let $D = \mathsf{Zeros}(\xi)$. Let $\xi' \in \mathcal{L}(3G)$ be given by Lemma 7.11 be such that

$$v_x(\xi') = \begin{cases} v_x(D) - 1 & x \in H \\ v_x(D) & x \in \mathsf{Support}(D) \setminus H \end{cases}$$

Such a ξ' exists because $\deg(3G) \geq \deg(2G) + 2g - 1 \geq \deg(D) + 2g - 1$.

For each $x \in H$, the local expansion of $\xi'(X)$ begins with $a t_x(X)^{v_x(D)-1}$, for some nonzero $a \in K$. For each $x \in \mathsf{Support}(D) \setminus H$, the local expansion of $\xi'(X)$ begins with $a t_x(X)^{v_x(D)}$, for some (possibly zero) $a \in K$.

Define $\xi'_m(X_1, \ldots, X_m) = \prod_{j=1}^{m} \xi'(X_i)$. Notice that $\xi'_m(X_1, \ldots, X_m) \in \mathcal{L}^m(3G)$.

First suppose that $f(X_1, \ldots, X_m) \in \mathcal{L}^m(G)$ is such that $f(x_1, \ldots, x_m) = 0$ for each $(x_1, \ldots, x_m) \in H^m$. Observe that $f(X_1, \ldots, X_m) \cdot \xi'_m(X_1, \ldots, X_m) \in \mathcal{L}^m(4G)$.

We now show that for all $(x_1, \ldots, x_m) \in \mathsf{Support}(D)^m$, $f(X_1, \ldots, X_m) \cdot \xi'_m(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity at least $(v_{x_1}(D), \ldots, v_{x_m}(D))$.

1. For $(x_1, \ldots, x_m) \in \mathsf{Support}(D)^m \setminus H^m$, we have that $\xi'_m(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity at least $(v_{x_1}(D), \ldots, v_{x_m}(D))$. Thus $f(X_1, \ldots, X_m) \cdot \xi'_m(X_1, \ldots, X_m)$ also has this property.

2. For $(x_1, \ldots, x_m) \in H^m$, the local expansion of $\xi'_m(X_1, \ldots, X_m)$ at $(x_1, \ldots, x_m)$ is of the form

$$\xi'_m(X_1, \ldots, X_m) = a \prod_{j=1}^{m} t_{x_j}(X_j)^{v_{x_j}(D)-1} + \ldots, \tag{21}$$

   where $a \neq 0$, and all the remaining terms vanish with individual multiplicity $\geq (v_{x_1}(D), \ldots, v_{x_m}(D))$.

   Since f vanishes at $(x_1, \ldots, x_m)$, the local expansion of $f(X_1, \ldots, X_m)$ at $(x_1, \ldots, x_m)$ has a 0 constant term. Combined with Equation (21), we get that $f(X_1, \ldots, X_m) \cdot \xi'_m(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity at least $(v_{x_1}(D), \ldots, v_{x_m}(D))$.

Thus by the Multiplicity Nullstellensatz (Theorem 7.10), we conclude that there exist $f'_1(X_1,\ldots,X_m),\ldots,f'_m(X_1,\ldots,X_m) \in \mathcal{L}^m(4G)$ such that:

$$f(X_1,\ldots,X_m)\cdot\xi'(X_1,\ldots,X_m) = \sum_{j=1}^{m} f'_j(X_1,\ldots,X_m)\cdot\xi(X_j),$$

as desired.

We now show the other direction. Suppose there exist $f'_1(X_1,\ldots,X_m),\ldots,f'_m(X_1,\ldots,X_m) \in \mathcal{L}^m(4G)$ such that:

$$f(X_1,\ldots,X_m)\cdot\xi'(X_1,\ldots,X_m) = \sum_{j=1}^{m} f'_j(X_1,\ldots,X_m)\cdot\xi(X_j).$$

Observe that the right hand side vanishes at each $(x_1,\ldots,x_m) \in \mathsf{Support}(D)^m$ with individual multiplicity $\geq (v_{x_1}(D),\ldots,v_{x_m}(D))$. Thus the left hand side should too.

Fix $(x_1,\ldots,x_m) \in H^m$. If $f(x_1,\ldots,x_m)$ equals some nonzero a', then by Equation (21) the local expansion of $f(X_1,\ldots,X_m)\cdot\xi'_m(X_1,\ldots,X_m)$ at $(x_1,\ldots,x_m)$ is of the form:

$$f(X_1,\ldots,X_m)\cdot\xi'_m(X_1,\ldots,X_m) = a'\cdot a\prod_{j=1}^{m} t_{x_j}(X_j)^{v_{x_j}(D)-1} + \ldots,$$

and thus $f(X_1,\ldots,X_m)\cdot\xi'_m(X_1,\ldots,X_m)$ would not vanish at $(x_1,\ldots,x_m)$ with individual multiplicity $\geq (v_{x_1}(D),\ldots,v_{x_m}(D))$. Thus $f(x_1,\ldots,x_m)$ must equal 0 for each $(x_1,\ldots,x_m) \in H^m$.

This completes the proof over an algebraically closed field K.

We now show how to deduce theorems 7.4 and 7.10 over an arbitrary perfect field K_0.

Indeed, given a curve $\mathcal{C}_0$ defined over K_0, a K_0-rational divisor G_0 on $\mathcal{C}_0$, an $f_0 \in \mathcal{L}^m(G_0)$, and a set of K_0-rational places $H_0 \subseteq \mathcal{C}_0$ we lift all this to the algebraic closure K (formally, we are extending the field of constants of $\mathcal{C}_0$ from K_0 to K, see [Sti93, Section 3.6]). We get an algebraic curve $\mathcal{C}$ defined over K, a K-rational divisor G with $\deg(G) = \deg(G_0)$ (the high-degree points in G_0 split into many degree 1 points in G), an $f \in \mathcal{L}^m(G)$, and a set of rational points $H \subseteq \mathcal{C}$ (with $|H| = |H_0|$). A crucial fact is that any K_0-basis of $\mathcal{L}(G_0)$ is also a K-basis of $\mathcal{L}(G)$ (see [Sti93, Proposition 3.6.3]), and thus any K_0-basis of $\mathcal{L}^m(G_0)$ is also a K-basis of $\mathcal{L}^m(G)$.

The functions f and f_0 are intimately related. By basic properties of the lifting, we have that f vanishes on H^m if and only if f_0 vanishes on H_0^m. Furthermore, if we express f_0 as a linear combination of a certain basis of $\mathcal{L}^m(G_0)$, then viewing that basis of $\mathcal{L}^m(G_0)$ as a basis of $\mathcal{L}^m(G)$, we have that f is the *same* linear combination of that basis.

In the proof of theorem 7.4 over algebraically closed field K, observe that one can choose the functions ξ, ξ' to be rational over K_0. This is because the proof of existence of ξ and ξ' only used the Riemann-Roch theorem, and the genus of $\mathcal{C}$ and $\mathcal{C}_0$ are equal (again, by [Sti93, Proposition 3.6.3]). Let $\ell_1,\ldots,\ell_L$ be a basis for $\mathcal{L}^m(4G_0)$. Let $(A_{ij} \mid i \in [m], j \in [L])$ be formal variables. Write $f'_{0,i}(X_1,\ldots,X_m) = \sum_{j\in[L]} A_{ij}\ell_j$. Consider the equation:

$$f_0(X_1,\ldots,X_m)\cdot\prod_{i=1}^{m}\xi'(X_i) = \sum_{i=1}^{m}\xi(X_i)\cdot f'_{0,i}(X_1,\ldots,X_m). \tag{22}$$

Both sides of the equation are in $\mathcal{L}^m(6G_0)$ (here we use the K_0-rationality of ξ and ξ'). Thus we can represent both sides in terms of a basis for $\mathcal{L}^m(6G_0)$, with the coefficients of the basis for the

RHS of the equation being K_0-linear combinations of the A_{ij}, and the coefficients of the basis for the LHS of the equation being elements of K_0.

Thus there exist $f'_{0,1}, \ldots, f'_{0,m} \in \mathcal{L}^m(4G_0)$ satisfying Equation (22) if and only if a certain system of linear equations over K_0 in the variables A_{ij} has a solution in K_0.

Now view $\ell_1, \ldots, \ell_L$ as a basis for $\mathcal{L}^m(4G)$. Write $f'_i(X_1, \ldots, X_m) = \sum_{j \in [L]} A_{ij}\ell_j$. Consider the equation:

$$f(X_1, \ldots, X_m) \cdot \prod_{i=1}^{m} \xi'(X_i) = \sum_{i=1}^{m} \xi(X_i) \cdot f'_i(X_1, \ldots, X_m). \tag{23}$$

As before, we have that there exist $f'_1, \ldots, f'_m \in \mathcal{L}^m(4G)$ satisfying Equation (23) if and only if the *same* system of linear equations over K_0 in the variables A_{ij} has a solution in K. But clearly such a system has a solution in K if and only if it has a solution in K_0.

Thus Equations (22) and (23) either both have solutions or both do not have solutions. We now complete the proof of the general AG Combinatorial Nullstellensatz. By the AG Combinatorial Nullstellensatz over algebraically closed fields, Equation (23) has a solution if and only if f vanishes on H^m, and this happens if and only if f_0 vanishes on H_0^m. Thus Equation (22) has a solution if and only if f_0 vanishes on H_0^m, as desired. ■

Acknowledgements

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, Preliminary version in FOCS 1992.

[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, Preliminary version in FOCS 1992.

[BCGT13a] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer, *Fast reductions from RAMs to delegatable succinct constraint satisfaction problems*, Proceedings of the 4th Innovations in Theoretical Computer Science Conference, ITCS '13, 2013.

[BCGT13b] ———, *On the concrete efficiency of probabilistically-checkable proofs*, Proceedings of the 45th ACM Symposium on the Theory of Computing, STOC '13, 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*, CCC '05: Proceedings of the 20th Annual IEEE Conference on Computational Complexity (Washington, DC, USA), 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.

[BGK$^+$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.

[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] ———, *Short PCPs with polylog query complexity*, SIAM J. Comput. **38** (2008), no. 2, 551–607, Preliminary version in STOC 2005.

[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 the 35th Annual ACM Symposium on Theory of Computing, STOC '03, 2003, pp. 612–621.

[BV09] Eli Ben-Sasson and Michael Viderman, *Composition of semi-ltcs by two-wise tensor products*, APPROX-RANDOM (Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, eds.), Lecture Notes in Computer Science, vol. 5687, Springer, 2009, pp. 378–391.

[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*, Electronic Colloquium on Computational Complexity (ECCC) (2005), no. 104.

[Din07] Irit Dinur, *The PCP theorem by gap amplification*, Journal of ACM **54** (2007), no. 3, 241–250, Preliminary version in STOC 2006.

[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/.

[Gol11] Oded Goldreich, *A sample of samplers: A computational perspective on sampling*, Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation, Springer, 2011, pp. 302–332.

[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, Preliminary version in STACS '91.

[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 J. Karloff, and Noam Nisan, *Algebraic methods for interactive proof systems*, J. ACM **39** (1992), no. 4, 859–868.

[Mei10] Or Meir, *IP = PSPACE using error correcting codes*, Electronic Colloquium on Computational Complexity (ECCC) (2010), no. 137, To appear in SIAM Journal on Computing.

[Mei12a] ______, *Combinatorial PCPs with short proofs*, IEEE Conference on Computational Complexity, 2012, pp. 345–355.

[Mei12b] ______, *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] ______, *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.

Appendix: Dense families of transitive codes

Henning Stichtenoth

We use standard notation of coding theory such as in [4]. By a code C we mean a linear code over the finite field $\mathbb{F}_q$. The parameters (length, dimension, minimum distance) of C are denoted by $n = n(C)$, $k = k(C)$ and $d = d(C)$, respectively. The relative parameters of C (rate and relative minimum distance) are $R = R(C) = k(C)/n(C)$ and $\delta = \delta(C) = d(C)/n(C)$.

The symmetric group $\mathcal{S}_n$ acts on the space $\mathbb{F}_q^n$ via $\pi(c_1, \ldots, c_n) = (c_{\pi(1)}, \ldots, c_{\pi(n)})$, for $\pi \in \mathcal{S}_n$, and the automorphism group of the code $C \subseteq \mathbb{F}_q^n$ is defined as $\mathrm{Aut}(C) = \{\pi \in \mathcal{S}_n \mid \pi(C) = C\}$. The code C is called *transitive* if its automorphism group is a transitive subgroup of $\mathcal{S}_n$, i.e., for any two indices $i, j \in \{1, \ldots, n\}$ there exists $\pi \in \mathrm{Aut}(C)$ such that $\pi(i) = j$.

A family of codes $(C_i)_{i=1}^{\infty}$ with increasing lengths $n(C_i) \to \infty$ is called *asymptotically good* if

$$\liminf_{i\to\infty} R(C_i) > 0 \quad \text{and} \quad \liminf_{i\to\infty} \delta(C_i) > 0. \tag{24}$$

The family $(C_i)_{i=1}^{\infty}$ is *dense* if there is a real constant $A > 0$ such that $n(C_{i+1}) \leq A \cdot n(C_i)$ holds for all $i \geq 1$. The aim of this note is to prove the following result.

Theorem A.14 *Assume that $q = \ell^2 > 4$ is a square. Then there exist asymptotically good dense families of transitive codes over $\mathbb{F}_q$.*

Theorem A.14 is a refinement of [8, Theorem 1.5] where the existence of asymptotically good families of transitive codes was shown (without the property of being dense). Our proof follows the same lines as the proof in [8]. The main ingredient is an appropriate tower of function fields over $\mathbb{F}_q$ (see Lemma A.17 below) and the construction of algebraic geometry codes from function fields.

For the theory of algebraic function fields, algebraic curves and algebraic geometry codes we refer to [7, 9]. In particular we need the following notions: for a function field $F/\mathbb{F}_q$, $g(F)$ denotes its genus and $N(F)$ denotes its number of rational places. A *tower of function fields* over $\mathbb{F}_q$ is a sequence $\mathcal{F} = (F_0 \subsetneqq F_1 \subsetneqq F_2 \subsetneqq \cdots)$ of function fields $F_i/\mathbb{F}_q$ with the following properties: (i) $\mathbb{F}_q$ is the full constant field of F_i, for all $i \geq 0$; (ii) all extensions F_{i+1}/F_i are separable; (iii) the genus $g(F_i)$ tends to ∞, for $i \to \infty$. It is well-known that the *limit* of a tower, $\lambda(\mathcal{F}) := \lim_{i\to\infty} N(F_i)/g(F_i)$ exists and is bounded by the Drinfeld-Vladuţ bound $\lambda(\mathcal{F}) \leq \sqrt{q} - 1$.

In order to prove Theorem A.14 we start with a specific tower over $\mathbb{F}_q$ (with $q = \ell^2$) as follows. Let $G_0 := \mathbb{F}_q(x_0)$ be the rational function field. For $i \geq 1$ we define recursively the fields $G_i := G_{i-1}(x_i)$ where x_i satisfies the equation

$$x_i^{\ell} - x_i = x_{i-1}^{\ell}/(1 - x_{i-1}^{\ell-1}). \tag{25}$$

Consider the subfields

$$\mathbb{F}_q(u) \subseteq \mathbb{F}_q(t) \subseteq \mathbb{F}_q(x_0) \quad \text{where} \quad t := x_0^{\ell} - x_0 \quad \text{and} \quad u := t^{\ell-1} + 1. \tag{26}$$

Let H_i be the Galois closure of G_i over $\mathbb{F}_q(u)$. The sequence

$$\mathcal{H} = (\mathbb{F}_q(u) \subseteq \mathbb{F}_q(t) \subseteq H_0 \subseteq H_1 \subseteq H_2 \subseteq \ldots) \tag{27}$$

is then a tower over $\mathbb{F}_q$; some of its properties are listed in Lemma A.15 below. We denote by p the characteristic of $\mathbb{F}_q$; i.e., $q = p^{2\mu}$ for some integer $\mu \geq 1$.

Lemma A.15 *(i) All extensions $H_i/\mathbb{F}_q(u)$ are Galois.*

(ii) The extension $\mathbb{F}_q(t)/\mathbb{F}_q(u)$ is Galois of degree $[\mathbb{F}_q(t) : \mathbb{F}_q(u)] = \ell - 1$.

(iii) The extension $H_i/\mathbb{F}_q(t)$ is a Galois p-extension of degree $[H_i : \mathbb{F}_q(t)] \geq \ell^{i+1}$.

(iv) The place $(u = 0)$ of $\mathbb{F}_q(u)$ splits completely in H_i; hence there are exactly $[H_i : \mathbb{F}_q(u)]$ distinct places of H_i which are zeros of u. All of these places are rational.

(v) The place $(u = \infty)$ of $\mathbb{F}_q(u)$ has exactly one extension in $\mathbb{F}_q(t)$, namely the place $(t = \infty)$, with ramification index $\ell - 1$. The ramification index $e_\infty^{(i)}$ of $(t = \infty)$ in $H_i/\mathbb{F}_q(t)$ is $\geq \ell^{i+1}$, its different exponent in $H_i/\mathbb{F}_q(t)$ is $d_\infty^{(i)} = 2(e_\infty^{(i)} - 1)$.

(vi) The place $(u = 1)$ of $\mathbb{F}_q(u)$ has exactly one extension in $\mathbb{F}_q(t)$, namely the place $(t = 0)$, with ramification index $\ell - 1$. The ramification index $e_0^{(i)}$ of $(t = 0)$ in $H_i/\mathbb{F}_q(t)$ is $\geq \ell^i$, its different exponent in $H_i/\mathbb{F}_q(t)$ is $d_0^{(i)} = 2(e_0^{(i)} - 1)$.

(vii) All places of $\mathbb{F}_q(u)$ except $(u = \infty)$ and $(u = 1)$ are unramified in H_i.

Proof: See [7, Section 7.4] ■

The next step in the proof of Theorem A.14 is to refine the tower $\mathcal{H}$ in order to get a *dense* tower. We need a lemma from Galois theory:

Lemma A.16 *Consider finite separable field extensions $H \subseteq L \subseteq M \subseteq N \subseteq \tilde{N} \subseteq S$ and assume that (a)-(c) hold:*

(a) $[L : H] = n$ and $[N : M] = m$.

(b) The extensions M/H, S/H and N/L are Galois.

(c) $\tilde{N}/H$ is the Galois closure of N/H.

Then $[\tilde{N} : M] \leq m^n$.

Proof: Let Γ be the Galois group of S/H and fix $\lambda_1, \ldots, \lambda_n \in \Gamma$ whose restrictions $\lambda_j|_L$ are pairwise distinct. For any $\sigma \in \Gamma$ there is a unique $j \in \{1, \ldots, n\}$ such that $\sigma|_L = \lambda_j|_L$. It follows that $\lambda_j^{-1} \circ \sigma$ is an automorphism of S/L, and hence $(\lambda_j^{-1} \circ \sigma)(N) = N$ (since N/L is Galois). Therefore $\sigma(N) = \lambda_j(N) =: N_j$. The Galois closure $\tilde{N}$ of N/H is the composite field of the fields $\sigma(N)$, over all $\sigma \in \Gamma$, so

$$\tilde{N} = \prod_{j=1}^{n} N_j.$$

As M/H is Galois, $M \subseteq N_j$ and $[N_j : M] = [N : M] = m$. This implies that $[\tilde{N} : M] \leq m^n$. ■

Let $H_i \subseteq H_{i+1}$ be two consecutive fields in the tower $\mathcal{H} = (\mathbb{F}_q(u) \subseteq \mathbb{F}_q(t) \subseteq H_0 \subseteq H_1 \subseteq H_2 \subseteq \ldots)$. We can break the extension H_{i+1}/H_i into steps of degree $\leq p^{\ell-1}$ as follows. Set $H_{i,0} := H_i$. The extension $H_{i+1}/\mathbb{F}_q(t)$ is a Galois p-extension, we denote its Galois group by Δ. The Galois group Δ_0 of H_{i+1}/H_i is a normal subgroup of Δ, as $H_i/\mathbb{F}_q(t)$ is Galois. By a well-known result from group theory, there exists a subgroup $\Delta_1 \subseteq \Delta_0$ of index $(\Delta_0 : \Delta_1) = p$ which is normal in Δ, see [2, Kap. III, Satz 7.2 d)]. Denote by T the fixed field of Δ_1. Now we apply Lemma A.16 (setting $H := \mathbb{F}_q(u)$, $L := \mathbb{F}_q(t)$, $M := H_i$, $N := T$ and $S := H_{i+1}$), and we obtain a field $\tilde{N} =: H_{i,1}$ such that $H_{i,0} \subseteq H_{i,1} \subseteq H_{i+1}$, $[H_{i,1} : H_{i,0}] \leq p^{\ell-1}$, and the extension $H_{i,1}/\mathbb{F}_q(u)$ is Galois. Repeating this process, we construct intermediate fields $H_i = H_{i,0} \subseteq H_{i,1} \subseteq \cdots \subseteq H_{i,s_i} = H_{i+1}$ where all fields $H_{i,j}$ are Galois over $\mathbb{F}_q(u)$ and each step $H_{i,j+1}/H_{i,j}$ is of degree $\leq p^{\ell-1}$.

The extensions $H_{i,j}/\mathbb{F}_q(t)$ are Galois p-extensions, and the two places $(t = 0)$ and $(t = \infty)$ of $\mathbb{F}_q(t)$ are the only ramified places in $H_{i,j}$; call their ramification indices $\varepsilon_0^{(i,j)}$ and $\varepsilon_\infty^{(i,j)}$, resp. By Lemma A.15, the corresponding different exponents are $2(\varepsilon_0^{(i,j)} - 1)$ and $2(\varepsilon_\infty^{(i,j)} - 1)$. The Hurwitz genus formula, applied to the extension $H_{i,j}/\mathbb{F}_q(t)$, gives then

$$g(H_{i,j}) = 1 + [H_{i,j} : \mathbb{F}_q(t)]\big(1 - \frac{1}{\varepsilon_0^{(i,j)}} - \frac{1}{\varepsilon_\infty^{(i,j)}}\big) \leq [H_{i,j} : \mathbb{F}_q(t)].$$

In order to avoid double indices i, j we rename the fields in the 'refined' tower. In the next lemma we put together the properties of this tower that will be used in the proof of Theorem A.14.

Lemma A.17 *There exists a tower $\mathcal{T} = (\mathbb{F}_q(u) \subseteq \mathbb{F}_q(t) \subseteq T_0 \subseteq T_1 \subseteq T_2 \subseteq \ldots)$ over $\mathbb{F}_q$ (with $q = \ell^2$) having the following properties:*

(i) All extensions $T_i/\mathbb{F}_q(u)$ are Galois; the extension $\mathbb{F}_q(t)/\mathbb{F}_q(u)$ is Galois of degree $\ell - 1$, and the extensions $T_i/\mathbb{F}_q(t)$ are Galois p-extensions.

(ii) The ramification indices $e^{(i)}$ of the place $(u = \infty)$ in the extensions $T_i/\mathbb{F}_q(u)$ tend to ∞ as $i \to \infty$.

(iii) $g(T_i) \leq [T_i : \mathbb{F}_q(t)]$, for all $i \geq 0$.

(iv) The element u has exactly $(\ell - 1)[T_i : \mathbb{F}_q(t)]$ zeros in T_i; all of them are rational places of T_i.

(v) For all $i \geq 0$ we have $[T_{i+1} : T_i] \leq p^{\ell-1}$; i.e., the tower $\mathcal{T}$ is 'dense'.

Proof: Items (i),(iii),(v) follow from the discussion above. Item (ii) follows from Lemma A.15(v). Item (iv) follows from Lemma A.15(iv). ■

Proof of Theorem A.14. Let δ be a real number with $0 < \delta < 1 - (\ell - 1)^{-1}$. Define R by the equation

$$R + \delta = 1 - \frac{1}{(\ell - 1)}, \tag{28}$$

then both δ and R are > 0. We will construct a dense sequence of transitive codes $(C_i)_{i=0}^\infty$ over $\mathbb{F}_q$ such that

$$\liminf_{i\to\infty} R(C_i) \geq R \quad \text{and} \quad \liminf_{i\to\infty} \delta(C_i) \geq \delta. \tag{29}$$

In what follows, we assume the reader to be familiar with the definition and basic properties of algebraic geometry (AG) codes (see [7, Chapter 2] or [9]). Consider the tower $\mathcal{T}$ as in Lemma A.17. For $i \geq 0$ let $n_i := [T_i : \mathbb{F}_q(u)]$ and define the divisor

$$D_i := \sum_{j=1}^{n_i} P_j^{(i)} \tag{30}$$

where the places $P_j^{(i)}$ are all zeros of the element $u - 1$ in T_i. Let $e^{(i)}$ be the ramification index of the place $(u = \infty)$ in the extension $T_i/\mathbb{F}_q(u)$ and set

$$A_i := \sum Q, \quad \text{where } \ Q \ \text{ runs over all poles of } u \text{ in } T_i. \tag{31}$$

Then $n_i = e^{(i)} \cdot \deg A_i$, and it follows from Lemma A.17(ii) that $\deg(A_i)/n_i \to 0$ as $i \to \infty$. Define the integers r_i by

$$1 - \delta - \frac{\deg A_i}{n_i} < r_i \cdot \frac{\deg A_i}{n_i} \leq 1 - \delta. \tag{32}$$

Define C_i as the AG code corresponding to the divisors D_i an $r_i A_i$; i.e.,

$$C_i := C_{\mathcal{L}}(D_i, r_i A_i). \tag{33}$$

The Galois group Γ_i of the field extension $T_i/\mathbb{F}_q(u)$ acts transitively on the places $P_1^{(i)}, \dots, P_{n_i}^{(i)}$ (since these are all places of T_i lying above the place $(u = 1)$), and Γ_i fixes the divisor A_i. Therefore the codes C_i are transitive codes, see [7, Section 8.2]. By Lemma A.17(v) we have

$$\frac{n(C_{i+1})}{n(C_i)} = \frac{[T_{i+1} : \mathbb{F}_q(u)]}{[T_i : \mathbb{F}_q]} \leq p^{\ell-1}, \tag{34}$$

hence the family $(C_i)_{i=0}^{\infty}$ is dense. It remains to verify the conditions (29). Given $\varepsilon > 0$, for all sufficiently large i holds by (32)

$$1 - \delta - \varepsilon < \frac{\deg r_i A_i}{n_i} \leq 1 - \delta. \tag{35}$$

The standard estimates for the parameters of an AG code (see [7, Section 2.2]) yield the inequalities

$$k(C_i) \geq r_i \deg A_i + 1 - g(T_i) \quad \text{and} \quad d(C_i) \geq n_i - r_i \deg A_i. \tag{36}$$

We divide by n_i and obtain (using Lemma A.17(iv),(v) and (32)) for the relative parameters of C_i the estimates

$$R(C_i) \geq \frac{r_i \deg A_i}{n_i} - \frac{g(T_i)}{n_i} > 1 - \delta - \varepsilon - \frac{1}{\ell - 1} = R - \varepsilon \tag{37}$$

and

$$\delta(C_i) \geq 1 - \frac{r_i \deg A_i}{n_i} \geq 1 - (1 - \delta) = \delta. \tag{38}$$

This finishes the proof of Theorem A.14. $\square$

Remark A.18 *In Theorem A.14 we have constructed dense families of transitive codes with asymptotic parameters on or above the Tsfasman-Vlăduţ-Zink bound $R + \delta = 1 - (\sqrt{q} - 1)^{-1}$, for every $\delta \in (0, 1 - (\sqrt{q} - 1)^{-1})$.*

Remark A.19 *For an explicit description of the codes $C_i = C_{\mathcal{L}}(D_i, r_i A_i)$ one needs an explicit construction of a basis of the Riemann-Roch spaces $\mathcal{L}(r_i A_i)$. In [3, 5, 6] polynomial-time algorithms are given for some specific towers. We believe that it is possible to provide such a polynomial time algorithm also for the tower $\mathcal{T}$, and hence to obtain an explicit construction of the codes C_i in polynomial time.*

References

[1] Garcia, A. and Stichtenoth, H.: *On the asymptotic behavior of some towers of function fields over finite fields*, J. Number Theory **61**, (1996), 248-273.

[2] Huppert, B.: *Endliche Gruppen I*, Springer-Verlag, 1967. Grundlehren der mathematischen Wissenschaften No. **134**.

[3] Katsman, G.L., Tsfasman, M.A. and Vlăduţ, S.G.: *Modular curves and codes with a polynomial construction*, IEEE Trans. Inform. Theory **30**, no. 2, part 2, (1984), 353–355.

[4] van Lint, J.H.: *Introduction to coding theory*, 2nd Edition. Springer-Verlag, 1992. Graduate Texts in Mathematics No. **86**.

[5] Noseda, F., Oliveira, G. and Quoos, L.: *Bases for Riemann-Roch spaces of one-point divisors on an optimal tower of function fields*, IEEE Trans. Inform. Theory **58** (2012), 2589-2598.

[6] Shum, K.W., Aleshnikov, I., Kumar, V.P., Stichtenoth, H. and Deolalikar, V.: *A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound*, IEEE Trans. Inform. Theory **47**, (2001), 2225-2241.

[7] Stichtenoth, H.: *Algebraic function fields and codes*, 2nd Edition. Springer-Verlag, 2009. Graduate Texts in Mathematics No. **254**.

[8] Stichtenoth, H.: *Transitive and self-dual codes attaining the Tsfasman-Vlăduţ-Zink bound*, IEEE Trans. Inform. Theory, **52**, (2006), 2218-2224.

[9] Tsfasman, M.A. and Vlăduţ, S.G.: *Algebraic-geometric codes*, Kluwer, 1991.