Multi-variate Abstractions of Algebraic Geometry Codes, With Applications

Yohay Kaplan
Multi-variate Abstractions of Algebraic Geometry Codes, With Applications

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Yohay Kaplan

Submitted to the Senate of the Technion — Israel Institute of Technology
Tevet 5772 Haifa January 2016
This research was carried out under the supervision of Prof. Eli Ben-Sasson, in the Faculty of Computer Science.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author’s doctoral research period, the most up-to-date versions of which being:


Part of this work was done while visiting MIT, Cambridge, MA. This research was supported by a grant of the US-Israel Binational Science Foundation and has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258. The generous financial help of the Technion is gratefully acknowledged.
# Contents

Abstract 1

1 Introduction 3

2 Preliminaries 7

2.1 Error Correcting Codes 7

2.2 Algebraic Geometry (AG) codes 8

3 A Schwartz-Zippel lemma and locally correctable codes 11

3.1 Background and overview 11

3.1.1 Locally correctable codes (LCCs) 11

3.1.2 On local correction, code symmetries, and local views 12

3.1.3 The tensor-product lifting of codes 13

3.1.4 Degree-lifting of codes 14

3.1.5 Lifting of affine-invariant codes 16

3.1.6 AG codes and degree-lifted AG codes 16

3.1.7 Explicit constructions and parameters 19

3.2 Context and history 20

3.2.1 Previous work on locally correctable codes 20

3.2.2 Previous work studying code invariance with respect to “local” properties 21

3.2.3 Previous work on codes over algebraic surfaces 21

3.3 On the hermitian function field 21

3.4 Definition and fundamental coding parameters of degree lifted AG codes 23

3.4.1 Definition of degree-lifted AG codes 24

3.4.2 Dimension of degree-lifted codes 24

3.4.3 A Schwartz-Zippel Lemma for degree-lifted AG codes 25

3.5 Single-step correction of degree lifted AG codes 26

3.5.1 Single-step correction of Hermitian codes 28

3.5.2 Degree lifting of 2-transitive AG codes are LDCs 29

3.5.3 Increasing transitivity via redundancy 30

3.6 Fractal correction of degree lifted AG codes 31
3.7 Correction via high-degree samplers ........................................... 36

4 A combinatorial nullstellensatz and probabilistically checkable proofs 41

4.1 Overview ....................................................................................... 41
  4.1.1 Our techniques ......................................................................... 43
  4.1.2 AG arithmetization ................................................................. 45
  4.1.3 Open problems .......................................................................... 47
  4.1.4 The road ahead .......................................................................... 48

4.2 Formal Statement of Main Results .................................................. 48

4.3 Tensors of AG Codes and an AG Combinatorial Nullstellensatz ...... 50
  4.3.1 Tensor Codes ............................................................................ 50
  4.3.2 AG codes and the multiplication property .................................. 52
  4.3.3 The AG Combinatorial Nullstellensatz ....................................... 54

4.4 Proof of Main Theorem 4.3 ............................................................. 56
  4.4.1 From circuit-SAT to Hypercube-CSP ......................................... 56
  4.4.2 From Hypercube-CSP to aggregate-AGCSP .............................. 57
  4.4.3 A proof of Main Theorem 4.3 using AG combinatorial Nullstellensatz 60

4.5 Hypercube CSPs — Proof of Theorem 4.8 ....................................... 64

4.6 From Hypercube-CSP to aggregate-AGCSP ..................................... 68
  4.6.1 A non-aggregated algebraic CSP .............................................. 68
  4.6.2 Aggregating the constraint polynomials ..................................... 71

4.7 Combinatorial Nullstellensatz over algebraic curves ......................... 75
  4.7.1 Formal statements of Nullstellensatz ......................................... 75
  4.7.2 Instantiating parameters .......................................................... 77
  4.7.3 Proof strategy ............................................................................ 79
  4.7.4 Proof of the AG Combinatorial Nullstellensatz for splitting sets 80
  4.7.5 Local Derivatives and multiplicities .......................................... 82
  4.7.6 AG Combinatorial Nullstellensatz for splitting sets with multiplicity 85
  4.7.7 Proof of the AG Combinatorial Nullstellensatz for general sets 88
Abstract

Polynomials over finite fields, and specifically the error correcting codes they induce, have been used extensively in all areas of theoretical computer science. These consist of the evaluation of univariate or multivariate (known as Reed-Solomon(RS) or Reed-Muller(RM) codes, respectively) polynomials on the points of some finite field. One of the limits of using such codes has been that the size of the field limits the degree of polynomials that we can use, and therefore the rate of the codes we achieve.

A well-known generalization of RS can be found in Algebraic Geometry(AG) codes. In these, we evaluate a well-chosen space of functions on a well-chosen set of points to get behavior that is strikingly similar to RS codes, both in parameters achieved and in the various algebraic properties polynomials have, but without the field size limiting the code size, though with an added cost in the complexity of description and operations. This has allowed for advances in many areas of theory (\(\epsilon\)-biased sets, list decoding, secret sharing etc.).

In this work we show how Algebraic Geometric codes can generalize RM codes by showing multi-variate versions of them. These include:

1. We construct locally correctable codes of previously unknown parameters by considering the "Degree lifting" of AG codes. This is a generalization of total degree RM (where only the total degree of a monomial is relevant). We define and prove the needed Schwartz-Zippel type lemma to make these codes useful and show how to generalize the standard RM correction methods.

2. We construct probabilistically checkable proofs(PCPs) of linear lengths and polynomial query length for \texttt{circuit-SAT} using tensored AG codes, which are a generalization of individual degree RM (where only the degree of each variable matters). No non-trivial linear PCPs were previously known. As a part of this construction, an appropriate generalization of Alon’s combinatorial nullstellensatz is given.
Chapter 1

Introduction

Polynomials over finite fields are a ubiquitous tool in theoretical computer science. They are repeatedly used in constructions of interesting objects or as essential parts of proofs, and have themselves become an intense area of research, as furthering our understanding of polynomials can allow us to obtain better results in other fields. The usefulness of polynomials comes from the wealth of algebraic structure that they are known to have. For instance, the fundamental property of polynomials having only as many zeroes as their degree allows as to use them for error correcting codes [RS60], extending this property to multi-variate polynomials (the Schwartz-Zippel lemma) and adding the ability to interpolate polynomials allows us to build list decodable codes [Sud97], which can be improved by considering polynomial derivatives [GS98] and so forth.

A fundamental limit to using polynomials, is that the field size must be greater then the degree of polynomials being used. This often functions as a limit to the parameters achieved by constructions or to the realm of applicability of proofs. So it seems immediately useful to have at our disposal objects that have a similar behavior and wealth of algebraic structure but without the limit on the field size. The known candidate for such are the evaluations of function spaces over an algebraic curve.

Univariate polynomials and algebraic curves

We will hold off on properly defining algebraic curves and their properties. For now, let us think of algebraic curves as the zero set of an absolutely irreducible bivariate polynomial $f(x, y)$ over some finite field $\mathbb{F}_q$. We will then consider evaluations of polynomials in $\mathbb{F}_q[x, y]$ on the set of zeros in $\mathbb{F}_q^2$ (rational points of $f$). We are only interested in polynomials that evaluate differently on this set of points so we would actually consider the ring $\mathbb{F}_q[x, y] \mod f(x, y)$. Note that the number of rational points of $f$ isn’t bound by $q$, and while it is bound by $q^2$, the actual definition of a curve removes that limit is well.

Let us consider the properties of univariate polynomials which we find useful:

1By necessity, this is a partial list.
1. The set of polynomials of degree $d$ or less is a vector space of dimension $d + 1$.

2. The number of zeros of a polynomial $p(x)$ is $\deg p(x)$.

3. $\deg p(x) \cdot q(x) = \deg p(x) + \deg q(x)$

4. $\deg p(x) + q(x) \leq \text{MAX} \{\deg p(x), \deg q(x)\}$

5. Affine invariance - for all $a, b \in \mathbb{F}$ $\deg p(ax + b) \leq \deg p(x)$

6. Interpolation - given a set of $d + 1$ points, for each possible assignment of values to those points there exists a unique degree $d$ polynomial whose evaluation on those points is equal to this assignments. This polynomial can be efficiently computed.

Let us see how well do algebraic curves emulate these properties.

**Degree**

Let $R = \mathbb{F}_q[x, y] \mod f(x, y)$. We can impose a degree function (which we call *curve degree*) on the functions in $R$. An easy way to define it is to say it is the number of zeros a function has on the zeros of $f$ (including zeros in the algebraic closure and accounting for multiplicity). Note that this can be quite different then the standard definition of polynomial degree.

So the curve degree, by definition, gives us property 2. It isn’t immediate, but it turns out that properties 3 and 4 also hold. As a consequence we get that the set $R^{\leq d}$ of functions of degree $d$ or less is a vector space. It is completely non-obvious what its dimension would be, for this we need the famous Riemann-Roch theorem which says that it is almost always $d + 1 - g$ where $g$ is a constant called the *genus* of the curve. So we get something close to property 1 except for the loss of the genus. This dimension deficit is the significant loss when using AG curves instead of polynomials (other then the added complexity) and pops up in most applications of AG curves.

**Interpolation**

An immediate area where we can see the genus pop up as a loss is in interpolation (property 6), where we can interpolate but not as exactly as for polynomials. What is true is that for any set of $d$ points and any possible assignment to them, we can find a function of curve degree $d + 2g$ that evaluates to these assignments. For the zero assignment we can find a non-zero function of curve degree $d + g + 1$.

A significant consequence of this is that not every set of points is exactly the set of zeros of a function. This is related to $R^{\leq d}$ not being a unique factorization domain.

**Invariance**

The rings $R^{\leq d}$ are generally not affine invariant. However, they can have very rich groups of symmetries (called *automorphisms*). An important part of the results in
chapter 3 is the observation that these automorphisms can be used instead of the affine
transformations that are so useful when using polynomials.

Multi-variate abstractions of AG

We have already seen that AG can provide spaces of functions that behave much like
univariate polynomial rings. The main focus of this thesis is in using AG to create spaces
that function like multivariate polynomial rings. By showing how these AG spaces
have some of the algebraic properties that multi-variate polynomial rings have, we can
then consider how to replace multivariate polynomials by multivariate AG in various
constructions to obtain mathematical objects with parameters that were previously
unknown to be possible.

The main results of this work are briefly discussed next, each is greatly expanded
upon (including the relevant background) in the corresponding chapter. They are
preceded by a short chapter of preliminaries that discusses some terms in coding theory
and AG codes which are common to both chapters.

Locally correctable codes and a Schwartz-Zippel lemma

In chapter 3 we construct locally correctable codes with previously unknown parameters.
This is done by defining an AG equivalent of total-degree polynomial codes and proving
the Schwartz-Zippel type lemma required to use them. The correction procedures are
made possible by the observation that the automorphism group of a curve can replace
the affine transformations of polynomials. Several possible correction procedures are
then considered and analyzed with the concrete example of the hamiltonian curve. This
result previously appeared in [BGK+13].

Probebalisticaly checkable proofs (PCP) and a combinatorial nullstellsatz

In chapter 4 we show how to construct linear length, polynomial query PCPs for
CIRCUIT-SAT. No non-trivial linear length PCPs were previously known. This con-
struction uses tensored AG codes, which behave like individual degree bound multivariate
polynomials. We again use automorphisms to replace affine transformations and also
prove an AG version of the combinatorial nullstellsatz. This result previously appeared
in [BKK+13].
Chapter 2

Preliminaries

2.1 Error Correcting Codes

For any \( n \in \mathbb{N} \), we denote \([n] \equiv \{1, \ldots, n\}\). 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 \( F \) be a finite field, and let \( k, \ell \in \mathbb{N} \). A (linear) code \( C \) is a linear one-to-one function from \( F^k \) to \( F^\ell \), where \( k \) and \( \ell \) are called the code’s message length and block length, respectively. We will sometimes identify \( C \) with its image \( C(F^k) \).

Specifically, we will write \( c \in C \) to indicate the fact that there exists \( x \in F^k \) 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 \( k/\ell \). If \( d = \delta_C \cdot \ell \), we also say that \( C \) is a \( [\ell, k, d]_F \)-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 \( x \in F^k \) 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 \( F^\ell \) is a codeword of \( C \) in time that is polynomial in \( \ell \). Moreover, observe that the code \( C \) always encodes the all-zeros vector in \( F^k \) to the all-zeros vector in \( F^\ell \).

We say that \( C \) is systematic if the first \( k \) symbols of a codeword contain the encoded message, that is, if for every \( x \in F^k \) 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 thesis, 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 \( [\ell, k, d]_F \)-code \( C : F^k \to F^\ell \) 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 $\ell$. 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 $f|_H = h$.

### 2.2 Algebraic Geometry (AG) codes

This section will give 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 $K$. It has an associated projective, irreducible, non-singular algebraic curve $C$.

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

$\mathbb{P}F$ is the set of places of $F/K$. Each place corresponds to a set of $K$-equivalent points on the algebraic curve $C$. 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 $P$. The degree of the place $P$ is the degree of the extension $F_P/K$. A place of degree 1 is called rational.

$v_P$ is the discrete valuation associated with the place $P$. We note that $v_P(z) > 0$ means that $z$ is 0 at $P$ and we say that $z$ has a zero of multiplicity $v_P(z)$ at $P$. If $v_P(z) < 0$ then $1/z$ is zero at $P$ and we say that $z$ has a pole of multiplicity $-v_P(z)$ at $P$.

$f(P) \in F_P$ 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. $v_P(t_P) = 1$)

$D_F$ is the free abelian group generated by the places of $F/K$. A divisor is a member of this group.

We denote the coefficient of $P$ in $D$ by $v_P(D)$ and define the relation:

$$D_1 \geq D_2 \iff \forall P \in \mathcal{P}_F, v_P(D_1) \geq v_P(D_2)$$

A divisor of the form $G = r \cdot P$ for some rational place $P$ is called a one point divisor. The next definition makes sense because every member of $F$ has a finite number of zeroes and poles.

**Definition 2.2.1** (Divisors associated with a function field member). For each $z \in F$ we define:
1. a principal divisor:

\[(z) = \sum_{P \in P_F} v_P(z) P\]

2. a pole divisor:

\[(z)_\infty = \sum_{P \in P_F, v_P(z) < 0} -v_P(z) P\]

3. a zero divisor:

\[(z)_0 = \sum_{P \in P_F, v_P(z) > 0} v_P(z) P\]

The \textbf{degree} of a divisor D is: \(\deg D = \sum_{P \in P_F} n_P \cdot \deg P_i\)

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

\textbf{Definition 2.2.2} (Riemann-Roch spaces). The \textbf{Riemann-Roch Space} of a divisor D is defined as:

\[\mathcal{L}(D) = \{ z \in F | (z) \geq -D \} \cup \{0\}\]

This is a finite dimensional \(K\)-vector space, the \textbf{dimension} of a divisor \(D\) is the dimension of its associated Riemann-Roch space and is denoted \(l(D)\).

\textbf{Theorem 2.1} (Riemann-Roch). \textit{For every function field, there is a positive constant \(g\) called the \textbf{genus} of the function field, for which:}

\[\forall A \in D_F, \deg A - l(A) \leq g - 1\]

\[i f \ deg A \geq 2g - 1, \deg A - l(A) = g - 1\]

\textbf{Lemma 2.2.3} (Basic facts about divisors). Let \(G, G'\) be divisors over a curve \(C\). 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 \((f) \geq -G\) then \(f \in \mathcal{L}(G)\).

\textbf{Definition 2.2.4} (Function field automorphism). A field isomorphism \(\phi : F \to F\) is an \textbf{automorphism} of the function field \(F/K\) if \(\forall z \in K : \phi(z) = z\).

\textbf{Theorem 2.2} (Automorphisms permute places). [Sti93, Sec 8.1] Let \(P\) be a place of the function field \(F\) and \(\phi\) an automorphism of it. Then \(P^\ast := \{ \phi(z) | z \in P\}\) is a place of \(F\) and \(\deg P = \deg P^\ast\).
This allows us to extend the action of automorphisms to places in the natural way. Which, in turn, allows us to define the action of automorphisms on divisors:

Let $D = \sum_{P \in P_F} n_P P$, then $\phi(D) = \sum_{P \in P_F} n_P \phi(P)$.

**Reminder**: An $[n, k, d]_q$ linear code is a $k$-dimensional subspace of $\mathbb{F}_q^n$ such that the hamming distance between any two words in the code is at least $d$.

**Theorem 2.3** (AG codes). [Sti93, Theorem 2.2.2] Let $F$ be a function field with a base field $K = \mathbb{F}_q$. Let $N = P_1, P_2 \ldots P_n$ be a set of rational places, and $D$ be a divisor such that $\forall P_i \in N, v_{P_i}(D) = 0$ and $\deg D < n$. Then

$$C_L(N, D) := \{(z(P_i))_{i=1}^n | z \in \mathcal{L}(D)\}$$

is an $[n, k, d]_q$ code with $k = \ell(D), d \geq n - \deg D$.

**Definition 2.2.5** (One point codes). $C_L(G, D)$ is called a **one point AG code** if $G$ is a one point divisor.

**Definition 2.2.6.** A **code automorphism** is a permutation of the coordinates of a code, $\sigma$ such that $\sigma(C) = C$.

**Theorem 2.4** (Function field automorphisms are code automorphisms). [Sti93, Sec 8.2] Let $C_L(D, G)$ be an AG code, and $\sigma$ an automorphism of the function field. If $\sigma(G) = G$ and $\sigma(D) = D$ then $\sigma$ is an automorphism of $C_L(D, G)$.

$P'|P$ means that $P' \in P_{F'}$ lies over $P \in P_F$ in the function field extension $F'/F$. 

\[ \text{Technion - Computer Science Department - Ph.D. Thesis PHD-2016-12 - 2016} \]
Chapter 3

A Schwartz-Zippel lemma and locally correctable codes

3.1 Background and overview

3.1.1 Locally correctable codes (LCCs)

Error correcting codes as envisioned in the early days of information theory [Sha48, Sha53, Ham50] and detailed in section 2.1 were mostly intended to be produced and consumed in blocks. To access any one message symbol $m_i$, a decoding procedure is applied to all $n$ symbols of the received word $w'$. Maintaining data-integrity is done block-wise in a similar way, say, by first decoding the message $m$ from $w'$ and then re-encoding $m$ to recover $w$.

The development of new randomized and interactive proof systems in the 1980’s called for error-correcting codes of a different nature (see Section 3.2.1 and the survey [Yek11]). The message and its encoding are now assumed to be either secret, or prohibitively long, and hence decoding/correcting a full block is either forbidden (due to secrecy) or intractable (because of its large block-length). In such settings the notions of locally decodable and locally correctable codes were distilled, because often it suffices to find out the value of a single message-symbol $m_i$, or a single codeword symbol $w_i$. To better discuss LCCs we assume oracle access to entries of the (possibly corrupted) codeword $w'$ and accordingly henceforth view a code $C$ as a set of functions $C = \{w : D \rightarrow \Sigma\}$ mapping a domain $D$ of size $n$ to the alphabet $\Sigma$, and use $w(i)$ to denote the $i$th entry of $w$.

A locally-correctable code $C$ is associated with a local corrector. This is a randomized procedure that is given as input a pointer $i \in D$ and has oracle access to a corrupted codeword $w' : D \rightarrow \Sigma$ that is within “small” relative Hamming distance $\delta$ of a codeword $w$ (think of $\delta = 0.01$). The local corrector queries $w'$ in a small number $q$ of locations ($q$ is called the query complexity of the corrector) and outputs a conjectured value $\hat{w}(i)$ for the “true” value $w(i)$ of the $i$th entry of the uncorrupted word $w$. We say the corrector
has soundness error $\epsilon$ for distance parameter $\delta$ if $\Pr[\hat{w}(i) = w(i)] \geq 1 - \epsilon$ holds for all $i \in D$ and all functions $w'$ that are within normalized Hamming distance $\delta$ of $w$. (Probability in the previous equation is over the randomness of the local corrector.) A code $C$ that has such a local corrector is called a $(q, \epsilon, \delta)$-LCC, and when we want to highlight the query complexity we call $C$ a $q$-LCC, assuming $\epsilon$ and $\delta$ are known. (See Section 3.2.1 for a brief survey of LCC constructions).

### 3.1.2 On local correction, code symmetries, and local views

For a code $C$ to be a $q$-LCC, every index $i \in D$ should be associated with a smooth set of $q$-local reconstructors (cf. [KT00])\(^1\). A $q$-local reconstructor for $i$ is a reconstruction function that can be computed by making $q$ queries to a received word $w'$. More formally, fix $D' \subset D$ with $|D'| = q$ and a function $r : \Sigma^q \to \Sigma$. We say that $(D', r)$ is a $q$-local reconstructor for $i \in D$ if for all\(^2\) codewords $w \in C$, we have $r(w|_{D'}) = w(i)$ (where $w|_{D'}$ is the restriction of the function $w$ to the domain $D'$). Roughly speaking, a set of $q$-local reconstructors $\{(D'_1, r_1), \ldots, (D'_t, r_t)\}$ for $i \in D$ is said to be smooth if sampling a random $j \in [t]$ and then sampling a random $\ell \in D'_j$ gives a distribution that is close (in statistical distance) to uniform over $D \setminus \{i\}$.

Required a smooth set of $q$-local reconstructors for every $i \in D$ calls for codes with quite a lot of structure. The sheer number of constraints — $n/q$ per index, summing up to $n^2/q$ overall — implies that picking these constraints arbitrarily, or at random, results (whp) in a code with rate 0. Indeed, all known families of LCCs — Reed-Muller (RM), multiplicity codes [KSY11a], and affine-invariant codes [KS08a, GKS13] — can be explained by two structural properties they possess:

(i) a doubly-transitive automorphism group, and (ii) a large-distance local view. We explain these two concepts next.

A code $C$ induces a group of automorphisms $\text{Aut}(C)$. This is the group of permutations $\pi$ of $D$ that keep the code invariant, i.e., $(w \circ \pi) \in C$ for all $w \in C$ where $(w \circ \pi)$ is the function (or codeword) defined by $w(i) \triangleq w(\pi(i))$ for all $i \in D$. A group $G$ acting on $D$ is doubly-transitive if for any two pairs of distinct elements $(i, j), (i', j') \in D^2$ there exists $\pi \in G$ mapping $i$ to $i'$ and $j$ to $j'$.

Fix $w \in C$. A $q$-local view of $w$ is the restriction $w|_{D'}$ of $w$ to a domain $D' \subset D$ with $|D'| = q$. Similarly, a $q$-local view of $C$ is a set of the form $C|_{D'} \triangleq \{w|_{D'} \mid w \in C\}$ for some $D' \subset D$ with $|D'| = q$. That is, $C|_{D'}$ is the projection of all codewords of $C$ to a domain $D' \subset D$ of size $q$. We informally say the local-view has large distance if the relative distance of $C|_{D'}$ is large.

To take a concrete example, consider $\text{RM}[m, d]_q = \{f : \mathbb{F}_q^m \to \mathbb{F}_q \mid \deg(f) \leq d\}$ and suppose $d = q/2$. (i) This code is doubly-transitive because it is affine-invariant, i.e., it

---

\(^1\)The results of [KT00] formally apply to the case of locally decodable codes, but the proofs can be examined and seen to hold for the more general case of locally correctable codes.

\(^2\)One can relax this requirement to hold only for almost all codewords, but in the context of linear error correcting codes (most LCCs are such) the two notions coincide.
is invariant under any invertible affine transformation $A$ of $\mathbb{F}^m_q$, as $\deg(f) \leq d$ implies $\deg(f \circ A) \leq d$. (ii) The code also has a large-distance $q$-local view, namely, the view $D' = \{(\alpha, 0, \ldots, 0) \mid \alpha \in \mathbb{F}_q\}$ obtained by fixing all but the first variable to 0. Clearly $\text{RM}[m, d, m, q]$ has relative distance $d/q = 1/2$ because this view is nothing but the well-known Reed-Solomon (RS) code $\text{RS}[d, q]$ consisting of univariate degree-$d$ polynomials over $\mathbb{F}_q$. The reason we mention these two properties is because any code that is (i) doubly-transitive, and (ii) has a $q$-local view $D'$ with relative distance $\delta$, is, in fact, a $(q, 0.1, \delta/20)$-LCC [KV10]. In order to gain a better understanding of the fundamental notion of local correctability it is helpful to explore other constructions of LCCs, that do not possess these properties. As additional motivation we point out that doubly-transitive codes, and in particular affine-invariant ones, have inherent coding-related limitations, e.g., their rate is very small when alphabet size and query complexity are fixed [BS05]. We now proceed to describe our code constructions.

### 3.1.3 The tensor-product lifting of codes

One way to go about constructing LCCs is to start with a “small” code and “lift” it. A “small” code $C$ is one with small block-length $q$ but large rate $\rho$ and large relative distance $\delta$. Lifting $C$ means we apply a simple algebraic or combinatorial operation to $C$ and obtain as a result a code $C'$ with larger block-length $n \gg q$. We want to find such lifting operations with the property that certain $q$-local views of $C'$ will be equal to $C$. This will be useful for showing $C'$ ‘has some local structure’ and is a good LCC.

Perhaps the simplest conceivable lifting process is tensoring [MS78].

**Definition 3.1.1** (Axis Parallel Views and Code Tensors).

An axis-parallel subset $D' \subset D^m$ is a set of the form

$$D' = \{(c_1, \ldots, c_{i-1}) \times D \times (c_{i+1}, \ldots, c_m)\},$$

for some $i \in m$ and $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_m \in D$. Fix a function $w : D^m \to \Sigma$. An axis-parallel view of $w$ is a restriction $w|_{D'}$ where $D' \subset D^m$ is an axis-parallel subset.

Let $C = \{w : D \to \Sigma\}$ be a code. The $m$-wise tensor of $C$, denoted $C^\otimes m$, is the set of functions $w : D^m \to \Sigma$ whose every axis-parallel view is in $C$.

For example, it can be seen that $(\text{RS}[d, q])^\otimes m$ is the set of $m$-variate polynomials of individual degree at most $d$. Therefore, $(\text{RS}[d, q])^\otimes m$ is a strict subcode of $\text{RM}[m, md, q]$. It can be verified that if $C$ is a linear code of rate $\rho$ and relative distance $\delta$, then $C^\otimes m$ is a linear code of rate $\rho^m$ and relative distance $\delta^m$. The large-distance axis-parallel views of $C^\otimes m$ can be used to show that $C^\otimes m$ is a locally testable code (LTC), i.e., there exists a randomized “tester” that makes $|D|^{O(1)}$ queries to $f : D^m \to \Sigma$ and distinguishes with

---

3 The other known families of LCCs, namely, affine-invariant codes [KS08a, GKS13] and multiplicity codes [KSY11a], are also (i) affine-invariant and (ii) have a large-distance local view.
high probability between the case that \( f \in C^\otimes m \) and the case that \( f \) is very far from \( C^\otimes m \) \cite{BS05, Val05a, CR05a, DSW06a, BV09a, BV09c}.

Unfortunately, code-tensoring fails as a general “lifting” method for constructing LCCs. The problem is that the only large-distance local-views we can put our hands on in \( C^\otimes m \) are the axis-parallel views and these do not correspond to a smooth set of reconstructors. There are only two axis-parallel lines that pass through \((i, j)\) in \( C \otimes C \) and similarly, only \( m \) axis-parallel lines pass through \((i_1, \ldots, i_m)\) in \( C^\otimes m \). This is regrettable because if \( C \) has a rich automorphism group, then it seems reasonable to expect that \( C^\otimes m \) has many non-axis-parallel views that are large-distance local views, as explained next. We end by pointing out that in spite of the limitations of tensoring for local correctability, we shall return to it later on to show that in some cases, the \( m \)-wise tensor of a nonlinear base-code is locally correctable (see 3.5.3).

### 3.1.4 Degree-lifting of codes

Let \( D \) be some finite domain, and \( \mathbb{F} \) be a finite field. Suppose we have a degree function \( \text{deg} \) assigning a non-negative integer \( \text{deg}(f) \) to functions \( f : D \to \mathbb{F} \). For ease of notation, for the rest of the introduction we fix a positive integer \( d \) which will be implicit in some definitions. We denote by \( C_d \) the the set of functions of degree at most \( d \). That is,

\[
C_d = \{ f : D \to \mathbb{F} | \text{deg}(f) \leq d \}.
\]

The point we make in this section is that if \( \text{deg}(f) \) ‘behaves like the degree of univariate polynomials’, we can lift \( C_d \) to a code \( C' \) of larger block-length, that will be convenient to analyze as an LCC. For this purpose the following definition will be useful.

**Definition 3.1.2.** We say \( \text{deg} \) is a curve-degree on \( D \) if the following properties hold.

(i) \( \text{deg}(f \cdot g) = \text{deg}(f) + \text{deg}(g) \). (ii) If \( \text{deg}(f) = d \), \( f \) either vanishes on at most \( d \) points in \( D \), or else it vanishes on all of \( D \). (iii) For any \( a, b \in \mathbb{F} \) and any \( f, g : D \to \mathbb{F} \),

\[
\text{deg}(a \cdot f + b \cdot g) \leq \max\{\text{deg}(f), \text{deg}(g)\}.
\]

Thus, the set of functions \( f : D \to \mathbb{F} \) with \( \text{deg}(f) \leq d \) is an \( \mathbb{F} \)-linear subspace. (iv) If \( \pi \in \text{Aut}(C_d) \), then \( \text{deg}(f \circ \pi) \leq \text{deg}(f) \).

It can be seen that when \( D = \mathbb{F} \) and \( \text{deg} \) is the degree function of univariate polynomials, \( \text{deg} \) is a curve-degree on \( \mathbb{F} \). The only property that is not immediate is property 3.1.2: In this case \( \text{Aut}(C_d) \) consists of the affine transformations \( \pi(X) = a \cdot X + b \) for some \( a, b \in \mathbb{F} \) with \( a \neq 0 \). And indeed, if \( f(X) \) is an univariate polynomial of degree \( d \), so is \( f'(X) \triangleq f(a \cdot X + b) \). The fact that \( \text{deg} \) behaves like the degree function of univariate polynomials, suggests the idea of lifting \( C_d \) to a code \( C' \) consisting of multivariate functions of a certain ‘total degree’. Denote by \( B \) the set of \( m \)-variate ‘monomials’ of total degree at most \( d \). Namely,

\[
B \triangleq \{ g_1(X_1) \cdot \cdots \cdot g_m(X_m) | g_i \in C_d; \sum_{j=1}^m \text{deg} g_j \leq d \}.
\]
Definition 3.1.3 (Degree-lifted Code). Using the notation above, the \( m \)-variate degree-
variate lifted code \( C^m_d \) of \( C_d \) is
\[
C^m_d \triangleq \text{span}\{B\}.
\]

The properties of curve-degree suggest a way to use automorphisms of \( C_d \) to locally
correct \( C^m_d \). It relies on the following fundamental definition.

Definition 3.1.4 (\( C \)-permissible subsets and views). Fix a code \( C = \{w : D \to \Sigma\} \) and
integer \( m \). A \( C \)-permissible subset of \( D^m \) is a subset \( D' \subset D^m \) that is of one of the
following forms.

- \( D' \) is an axis-parallel subset as per Definition 3.1.1.
- \( D' = \{(\pi_1(i), \ldots, \pi_m(i)) \mid i \in D\} \), for some \( \pi_1, \ldots, \pi_m \in \text{Aut}(C) \).

For a code \( C' = \{w : D^m \to \Sigma\} \) a \( C \)-permissible view of \( C' \) is a view of the form \( C'|D' \)
for a \( C \)-permissible subset \( D' \subset D^m \).

The usefulness of this definition for local correction stems from the following claim.

Claim 3.1.5. Let \( \text{deg} \) be a curve-degree on \( D \). Then the \( C_d \)-permissible views of \( C^m_d \)
are subsets of \( C_d \).

Proof. Let \( D' \subset D^m \) be a \( C_d \)-permissible subset. Recall that \( C^m_d \) is the span of functions
\( w : D^m \to \mathbb{F} \) of the form \( w = g_1(X_1) \cdots g_m(X_m) \), with \( \sum_{j=1}^{m} \deg(g_j) \leq d \). Thus, it
suffices to show that for \( w \) of this form, \( w|_{D'} \in C_d \). Fix such \( w \), and denote \( w' = w|_{D'} \).
Suppose \( D' \) is an axis-parallel subset. Then,
\[
w'(X) = c_1 \cdots c_{j-1} \cdot g_j(X) \cdot c_{j+1} \cdots c_m = c \cdot g_j(X),
\]
for some \( c \in \mathbb{F} \). Thus, as a function from \( D \) to \( \mathbb{F} \), \( \deg(w') \leq d \) and \( w' \in C_d \). If \( D' \) is not
an axis-parallel subset then \( w' \) is of the form
\[
w'(X) = g_1(\pi_1(X_1)) \cdots g_m(\pi_m(X_1))
\]
for some \( \pi_1, \ldots, \pi_m \in \text{Aut}(C_d) \). From Property 3.1.2 of curve-degree, \( \deg(g_j \circ \pi_j) \leq \deg(g_j) \) for every \( j \in [m] \). It follows that \( \deg(w') \leq d \) and \( w' \in C_d \).

The above claim suggests the following reconstruction procedure for \( C^m_d \). Suppose
we need to locally correct \( i \in D^m \) and have oracle access to a corrupted function
\( w' : D^m \to \Sigma \). We pick a random \( C_d \)-permissible subset \( D' \) in \( D^m \) that contains \( i \),
correct \( w'|_{D'} \) to the closest word \( v \in C_d \) and output the value assigned to \( i \) by \( v \).

Loosely speaking, when the \( C_d \)-permissible subsets are ‘well distributed’ in \( D^m \) it
can be shown that this correction procedure succeeds with high probability. It can be
seen that this will be the case, for instance, when \( C_d \) is doubly-transitive. I.e., when
\( \text{Aut}(C_d) \) is a 2-transitive group (see Section 3.5.2).
3.1.5 Lifting of affine-invariant codes

A form of code lifting that is very similar to ours (but is subtly different) was introduced for affine-invariant codes in [BMSS10], and used there to prove that affine-invariant low-density-parity-check (LDPC) codes are not necessarily locally testable (cf. Definition 4.1 there). Recently, and independently of this work, this form of “affine lifting” was shown in [GKS13] to lead to new constructions of codes that are simultaneously locally testable and locally correctable, and have parameters that essentially match those of multiplicity codes [KSY11a].

In both kinds of lifting — “degree” and “affine” — one starts with a code \( C = \{ f : D \to F \} \) and ends with a code \( C' = \{ f : D^m \to F \} \) that has the property that every \( C \)-permissible view of \( C' \) is a subset of \( C \) but there are important differences between the two. Some of the differences are syntactic: “affine lifting” assumes an affine-invariant code, and is defined using the “degree-set characterization” of affine invariant properties (cf. [GKS13, Definition 2.1]) whereas “degree lifting” assumes an algebraic degree function. Certainly some codes that posses a curve-degree are not affine invariant (e.g., the Hermitian code described later), and in the other direction it is not clear that all affine invariant codes can be defined as a space of low curve-degree for some curve-degree function (cf. Definition 3.1.2).

The difference between the two notions of lifting runs deeper. For the case of affine invariant code \( C \), a function \( f : D^m \to F \) belongs to the affine lifting of \( C \) if and only if every \( C \)-permissible view of it — i.e., every restriction of it to a 1-dimensional affine space — is a codeword of \( C \) ([GKS13, Proposition 2.5]). In contrast, we know of algebraic geometry codes \( C \) for which there exist functions \( f : D^m \to F \) that do not belong to \( C' \) because their curve-degree is too large, yet every \( C \)-permissible view of them belongs to \( C \), and in some cases even the lifting of RS codes will exhibit this behavior[FS95].

One final and crucial difference that we point out is that in the case of degree lifted codes, we do not assume that \( C \) is doubly transitive, and this complicates the process of locally correcting \( C \)-lifted codes, as explained next.

3.1.6 AG codes and degree-lifted AG codes

The previous three subsections motivate the following questions: Besides \( D = F \) and \( \deg \) being the standard degree of univariate polynomials, are there domains \( D \) for which we can define a curve-degree? Are there such cases where the resulting code \( C_d \) will be doubly transitive? We have seen that a positive answer to these questions will give us new families of locally correctable codes.

We now address the first question. It turns out that algebraic geometry gives us a way to define a curve-degree on \( D \), when \( D \) is ‘the set of rational points of an irreducible curve’\(^4\). Indeed, for these degree functions the resulting codes \( C_d \) are known

\(^4\)See the next two sections for more precise definitions.
as algebraic-geometry (AG) codes\(^5\), and have been extensively studied (see Section 3.3 for details). In particular, there are arbitrarily long codes with the best known rates (above the Gilbert-Varshamov bound) over some constant sized field and with constant relative distance [GS96], for which we can define a curve degree and apply degree lifting.

We use the properties of curve-degree to analyze the basic parameters of \( C_d^m \). We show that if the rate of \( C_d \) is \( \rho \) then the rate of \( C_d^m \) is roughly \( \rho/m! \). Turning to relative distance, the properties of curve-degree lead to a generalization of the ubiquitous Schwartz-Zippel Lemma for degree-lifted AG codes. This generalization implies that the relative distance of \( C_d^m \) is at least the relative distance of \( C_d \) (Lemma 3.4.4). Given the pivotal role of the Schwartz-Zippel Lemma in the study of RM-codes, we hope this generalization will be used to find other applications of degree-lifted AG codes. We stress that both the rate and relative distance of degree-lifted codes depend only on the rate and distance of the base-code, hence can be applied to any AG code with an appropriate degree function.

We now address the second question. We are not aware of any doubly-transitive AG-code, except RS. However, certain AG-codes — for example, Hermitian, Suzuki, and Ree codes — have a rather rich automorphism group that is not quite doubly-transitive. In what follows we describe a number of methods for locally correcting degree-lifted AG codes that have a rich automorphism group, albeit one that is not doubly-transitive. The parameters obtained by them for degree-lifted Hermitian codes are listed in the table in the following section. Below, we denote by \( C \) a code \( C_d \) for some curve-degree function \( \deg \).

**Fractal correctors** To compensate for the code automorphism group not being exactly 2 transitive, we perform correction in several steps. On input \( i, w' \) as above, the fractal corrector first picks a random \( C \)-permissible subset \( D' \) that contains \( i \). Next, for each \( j \in D' \), pick a random \( C \)-permissible subset \( D'_j \) that passes through \( j \) and locally-correct \( j \) as explained above. Finally, use the corrected values of all \( j \in D' \) to locally correct \( i \) as above. While we only performed 2 steps in this example, and we only require 2 steps to locally correct degree lifted Hermitian codes, this procedure can be generalized to any number of steps. The name we chose — “fractal corrector” — is explained by the set of queries this corrector makes, which has the size of a \( c \)-dimensional surface in \( D^m \) but resembles more a collection of 1-dimensional curves. (Details appear in Section 3.6.)

**High-degree correctors for degree-lifted codes** For certain AG-base codes with a nearly-doubly transitive automorphism group, the query complexity of local-correction can be reduced from \( |D|^2 \) to \( |D| \). The main challenge here is that the \( C \)-permissible subsets containing a given \( i \in D^m \) do not form a smooth set. The key to our solution is that \( C^m \) contains views that, although not \( C \)-permissible, have large distance. For

\(^5\)Specifically, one-point AG codes
instance, in the case of RM-codes, consider the set of parameterized quadratic curves in $\mathbb{F}_q^m$ of the form

$$D' = \{(P_1(x), \ldots, P_m(x)) \mid x \in \mathbb{F}_q\} \quad \text{deg}(P_i) = 2.$$  

These views are clearly not RS$[d]_q$-permissible, but nevertheless the relative distance of the corresponding view of RS$[m, d]_q$ is still pretty good (it is $1 - 2\delta$ where $1 - \delta$ is the distance of the base-RS-code). It turns out that for certain AG-base codes, like the Hermitian code, a similar notion of high-degree local views makes sense. The restriction of $C^m$ to such a high-degree view has relative distance that is smaller than that of $C$, but is nevertheless sufficiently large to be of use in correction. And the main benefit of using high-degree views is that they can form, in certain cases, a smooth set and hence can be used to locally correct $C^m$ (details appear in Section 3.7).

**Locally correctable Tensors of AG-base codes** We now return to the most basic “lifting” operation — code-tensoring. In Section 3.5.3 we give a concrete example of a code based on a slight generalization of the standard definition of AG codes. This code is doubly-transitive and its $m$-wise tensor $C^\otimes m$ is locally correctable. To correct $i \in D^m$ in a corrupted word $w'$ we pick a view $D'$ as defined in (3.1.4) that passes through $i$. Given that $C$ is doubly-transitive we show that the set of views is smooth, and the only thing left to argue is that the view has good distance.
### Explicit constructions and parameters

We end with a number of concrete constructions. These are obtained by taking $C$ to be the Hermitian code defined in the next section. The first column in the table below refers to the local-correcting procedure, the second column gives the rate of the lifted code, the third is the query complexity of the local-corrector and the last is the block-length. Notice that the rate and query complexity in all constructions are very close to that of the RM-code, and the block-length is significantly larger. (A different way to phrase this comparison is to say that for a given block-length and rate, the new AG-based LCCs have a much smaller alphabet). In the last row we give the best parameters feasible for constant rate LCCs based on degree lifted AG codes. We explain how it might be possible to get these codes immediately below the table. All constructions are over a field of size $q^2$.

<table>
<thead>
<tr>
<th>construction</th>
<th>rate</th>
<th>queries</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reed-Muller</td>
<td>$\frac{1}{m} (1 - \delta)$</td>
<td>$\sqrt[n]{n}$</td>
<td>$q^{2m}$</td>
</tr>
<tr>
<td>Single step correction (Section 3.5.1)</td>
<td>$\frac{1}{m} \left( 1 - q^{m-1}\delta - \frac{1}{q} \right)$</td>
<td>$\sqrt[n]{n}$</td>
<td>$q^{3m}$</td>
</tr>
<tr>
<td>Fractal correction (Section 3.6)</td>
<td>$\frac{1}{m} \left( 1 - \delta - \frac{m}{q} \right)$</td>
<td>$\sqrt[n]{n^2}$</td>
<td>$q^{3m}$</td>
</tr>
<tr>
<td>High degree correctors</td>
<td>(\forall t &gt; 0) (Section 3.7)</td>
<td>$\frac{1}{m} \left( 1 - \delta - \frac{1}{q} - \frac{m}{t} \right)$</td>
<td>$\sqrt[n]{n}$</td>
</tr>
<tr>
<td>Locally correctable</td>
<td>tensored AG codes (Section 3.5.3)</td>
<td>$\frac{1}{m} \left( 1 - \delta - \frac{1}{q} \right)$</td>
<td>$\sqrt[n]{n}$</td>
</tr>
<tr>
<td>Tower of curves</td>
<td>(hypothetical construction, see below)</td>
<td>(1 - \delta - \frac{1}{q})</td>
<td>$\sqrt[n]{n}$</td>
</tr>
</tbody>
</table>

**Codes over constant sized alphabets via a tower of curves**

Tensored Hermitian codes, whose local correctability parameters are listed in rows 2–5 in the table above, allow for LDCs of rate half and block length which is $q^{3/2}$ over an alphabet of size $q$. Using Ree and Suzuki curves [HS90, Ped92] instead, the block length could be extended up to $q^3$. To be able to get LCCs of any length over a constant sized alphabet, we need curves with an arbitrarily large number of points and a good automorphism group over a constant sized alphabet. To achieve this it is necessary to look at families of curves in a large-dimensional space. The standard method for constructing such curves in a way that still allows us to analyse them is via *towers* of curves [Sti93, Section 7.2].

In a tower of curves, we start with a single plane curve and then apply it iteratively to get an $m$ dimensional curve, the properties of the $m$-dimensional curve will then depend on the properties of the original planar curve.

In order to get LCCs of arbitrarily large length over constant sized alphabet using degree lifted towered AG codes, as listed in the last row of the table above, it is necessary to find a tower of curves which has a large number of automorphisms. The currently well-known towers ([GS96], Hermitian towers and others) do not seem to have this
Concatenation. A standard method of reducing the alphabet size of LCCs is code concatenation, which allows us to take a regularly constructed LCC and concatenate it with a code over a small alphabet to get an LCC with an alphabet size of our choosing. Comparing degree lifted AG codes to RM codes after concatenation, the gains are that degree lifted AG codes reduce the query complexity by a multiplicative factor, and induce a distribution over bits read during correction that, depending on the method of correction used, is either pairwise independent, or significantly closer to it then codes gotten by concatenation, which may be important in future applications.

3.2 Context and history

3.2.1 Previous work on locally correctable codes

The LCCs introduced here are, in particular, also locally decodable codes (LDCs). (In fact, any linear LCC is an LDC.) While mentioned as early as [Rec54], LDCs became widely studied (though only implicitly) during the 90s, as part of the drive towards constructions of PCPs [BF90, Lip90, LFKN92, Sha92, BFLS91, BFL92, AS98]. Their explicit definition was given in [KT00], and they have remained an object of intense, explicit study in their own right, while also being used in other results in computational complexity [BFNW93, IW97, AS03, STV99, SU05] and cryptography ([CGKS98], for instance). See [Yek11] for a recent survey on locally decodable codes.

The most intense study has been in the area of constant query LDCs. For many years, it was widely believed that constant query LDCs required exponential length \( n = \Theta(\exp(k^\alpha)) \) for some constant \( \alpha \). A recent series of results [Yek08, Rag07, Efr09, DGY10, BAETS10, IS10, CFL+10] showed this belief to be false by constructing constant query LDCs which have \( n = O(\exp(\exp(\log^2 k))) \) for a constant \( \alpha \) (i.e. sub-exponential length).

Of equal interest is a lower bound on the length of such codes. This area has also seen considerable work [KdW04, GKST02, WdW05, Woo07, DJK+02, Oba02], and already in [KT00] it was shown that no constant query, constant rate LDCs exist. The best known lower bound on the length of constant query LDCs is \( \Omega(k^{1+\delta(r)}) \) (where \( r \) is the number of queries and \( \delta(r) < 1 \)). It is a major open problem to decrease the gap between these lower and upper bounds.

Flipping the question on its head, and looking only at constant rate LDCs, the only known lower bound on the query complexity is \( \Omega(\log k) \) [KT00], while all the actual constructions known use a polynomial (\( \Theta(k') \)) number of queries.

For many years the only known LDCs with constant rate were Reed-Muller (RM) codes, which were limited to rate \( \frac{1}{2} \). This limit was inherent in the fact that RM codes must only use polynomials whose degree was smaller than the field size, or the basic property of unique decoding of a codeword would be lost (over \( \mathbb{F}_q, x \equiv x^q \)). Recently,
[KSY11a] showed that by evaluating both a polynomial and some of its derivatives, one can start with polynomials of degree greater than field size, thereby increasing the rate while at the same time, keeping the property of local correctability using an algorithm similar to the RM correction algorithm. This enables codes of rate approaching 1, that are locally decodable with $O(k^\epsilon)$ queries in the presence of a constant $\delta$ (which is determined by the rate and by $\epsilon$) fraction of errors.

3.2.2 Previous work studying code invariance with respect to “local” properties

The study of the automorphism group of various error correcting codes is an important and well-established branch of classical coding theory, and this is particularly true for the study of AG-codes [Gop88, Sti90]. See also the recent works [KW10, KL12] which study LDPC codes with relatively high rate and a rich invariance group.

In the context of “local” codes, most of the work focused on understanding locally testable codes in terms of their automorphism group. Random low-density-parity-check (LDPC) codes are not locally testable [BHR05] and furthermore a rather large set of local views is required for local testability [BGK+05]. [BSS03] showed that cyclic-LTCs cannot have constant rate. [AKK+05] asked whether all doubly-transitive codes with a local constraint are LTCs. [KS08a] initiated a study of this question in the context of affine-invariant linear codes and a large body of work has accumulated around this question [GKS09, BS10, BMSS10, BGM+11, BRS12, GKS13] (see [Sud10] and references therein).

Turning to locally decodable and correctable codes, we have already pointed out that doubly-transitive codes with a local-view are locally correctable, hence also locally decodable. Works studying the connection between group representation theory and LDC constructions includes [RY07] and [Efr12].

3.2.3 Previous work on codes over algebraic surfaces

AG codes were introduced by Goppa [Gop82] and famously used to break the Gilbert-Varshamov bound [TVZ82]. Intuitively, AG codes involve the evaluation of appropriately chosen function spaces over the points of a 1-dimensional object in $m$-dimensional space. In the wake of Goppa’s work, there have been several works on codes over various high dimensional algebraic surfaces ([TS91, Lac93, Han01, rod03, LGS05] for example. See [Lit08] for a survey). To the best of our knowledge, this is the first application of such high-dimensional AG-codes to theoretical computer science.

3.3 On the hermitian function field

In this section we’ll build on section 2.2 by adding details about the curve degree of members of an algebraic function and presenting the important facts about the
Hermitian curve, in particular we’ll study with some thoroughness the structure of the automorphism group of Hermitian codes.

We start by defining an important notion equivalent to polynomial degree:

**Definition 3.3.1.** The curve degree of \( z \in F \) is: \( \deg_C z := \deg (z)_\infty = \deg (z)_0 \) (this equality is always true).

We note that this means that the curve degree of \( z \in F \) is the number of zeros (accounting for multiplicities, and points in the algebraic closure) it has on the curve. This matches with the degree of a polynomial, which is also equal to the number of zeroes it has (again, when accounting for multiplicities and points in the algebraic closure). Unfortunately, the degree of a polynomial can be calculated by looking at its highest power, while the curve degree of \( z \in F \) can have a much more mysterious behaviour. However, in the Hermitian function field, which is the concrete example of a function field that we will be working with in this part there is a simple way of calculating the curve degree of a polynomial.

This function field will be of particular importance in this part of our work.

**Definition 3.3.2 (Hermitian function field).** Let \( q \) be some prime power, \( K = \mathbb{F}_{q^2} \), then \( \mathbb{H} \), the Hermitian function field is the field created by taking \( k(\mathbb{F}_{q^2}) \left[y]\right] \) mod \( y^q + y = x^{q+1} \)

The next two theorems will state the important properties of this function field.

**Theorem 3.1 (Structure of the Hermitian function field).** [Sti93, theorem 2.3.2] The following properties hold for \( \mathbb{H} \):

- The genus of \( \mathbb{H} \) is \( \frac{q^2 - 2}{2} \).
- There are \( q^3 \) pairs \((\alpha, \beta) \in \mathbb{F}_{q^2}^2\) such that \( \beta^q + \beta = \alpha^{q+1} \).
- For any \((\alpha, \beta) \in \mathbb{F}_{q^2}^2\) such that \( \beta^q + \beta = \alpha^{q+1} \) there is a unique rational place \( P_{\alpha, \beta} \) such that \((x - \alpha)(P_{\alpha, \beta}) = (y - \beta)(P_{\alpha, \beta}) = 0 \).
- There is a rational place \( Q_\infty \), which is the only pole of both \( x \) and \( y \).
- The places described in 2,3 are the only rational places of \( \mathbb{H} \) (for a total of \( q^3 + 1 \))
- \((x)_\infty = q \cdot Q_\infty, \ (y)_\infty = (q + 1) \cdot Q_\infty \)
- For any \( r \), the set \( \{x^iy^j | i \cdot q + j \cdot (q + 1) \leq r, j < q \} \) is a basis for \( L(\mathbb{H} \cdot Q_\infty) \)

**Definition 3.3.3.** Let \( N_H \) be the set of all rational places of the Hermitian function field of the form \( P_{\alpha, \beta} \). (i.e. all the rational places of \( H \) except \( Q_\infty \))

The automorphisms of \( \mathbb{H} \) will also be of importance:

**Theorem 3.2 (Structure of the Hermitian automorphisms).** [Sti93, Ex 6.10]
1. For any place $P_{\alpha, \beta} \in NH$ there is an automorphism $\sigma_{\alpha, \beta}$ such that $\sigma_{\alpha, \beta}(x) = x + \alpha, \sigma_{\alpha, \beta}(y) = y + \alpha^q x + \beta$. These automorphisms form a group $V$ of size $q^3$.

Note that:

(i) $\sigma_{\alpha, \beta}(P_{0,0}) = P_{\alpha, \beta}$.

(ii) $\sigma_{\alpha, \beta}^{-1} = \sigma_{-\alpha, \alpha^q + 1 - \beta}$ (it can be verified that $(-\alpha, \alpha^q + 1 - \beta)$ is, indeed, a rational point).

2. For any $c \in \mathbb{F}_{q^2}^*$ there is an automorphism $\tau_c$ such that $\tau_c(x) = cx, \tau_c(y) = c^{q+1}y$. These automorphisms form a cyclic group $W$ of size $q^2 - 1$ which stabilizes $P_{0,0}$ (i.e. $\tau_c(P_{0,0}) = P_{0,0}$).

3. The group $U$ generated by $V$ and $W$ is of size $q^3(q^2 - 1)$ and stabilizes $Q_\infty$.

Corollary 3.3 (Representations of Hermitian automorphisms). We can conclude the following from Theorem 3.2:

1. For any $P_{\alpha, \beta} \in NH$ and $c \in \mathbb{F}_{q^2}^*$ there is an automorphism $\varphi_{\alpha, \beta, c}$ such that $\varphi_{\alpha, \beta, c}(x) = cx + \alpha$ and $\varphi_{\alpha, \beta, c}(y) = c^{q+1}y + \alpha^q cx + \beta$. This set of automorphisms is exactly the group $U$.

2. For any $P_{\alpha_1, \beta_1}, P_{\alpha_2, \beta_2} \in NH$ there are exactly $q^2 - 1$ automorphisms taking $P_{\alpha_1, \beta_1}$ to $P_{\alpha_2, \beta_2}$, these are $\sigma_{\alpha_2, \beta_2} \tau_c \sigma_{\alpha_1, \beta_1}^{-1}$ for any $c \in \mathbb{F}_{q^2}^*$. We can also write these explicitly as:

$$
\phi_c(x) = (x - \alpha_1) \cdot c + \alpha_2
$$

$$
\phi_c(y) = \left(y - \alpha_1^q x + \alpha_1^{q+1} - \beta_1\right) \cdot c^{q+1} + \alpha_2^q (x - \alpha_1) \cdot c + \beta_2
$$

and note that these automorphisms depend only on $c$.

Since for any $\sigma \in U, \sigma(Q_\infty) = Q_\infty$ and $\sigma$ permutes the other rational places of $N$, it is a subgroup of the group of automorphisms of the AG code $C_L(r \cdot Q_\infty, NH)$ (Theorem 2.4). In fact, for all interesting values of $r$, it is exactly the group of code automorphisms [Xin95].

3.4 Definition and fundamental coding parameters of degree lifted AG codes

In this section, we define the notion of degree-lifted AG codes and establish their fundamental coding parameters — dimension (related to code-rate) and relative distance. To prove a lower bound on distance we present a generalization of the Schwartz-Zippel Lemma (lemma 3.4.4), bounding the number of zeroes of polynomials on the rational points of a curve.

23
3.4.1 Definition of degree-lifted AG codes

Definition 3.4.1 (Degree-lifted AG codes). Let \( \mathcal{L}_C(G,D) \) be a one point AG code (see def 2.2.5). Define:

\[
\mathcal{L}^m(G) = \text{sp} \left\{ f_1(X_1) \cdot f_2(X_2) \cdots f_m(X_m) \mid \forall i : f_i \in \mathcal{L}(G), \sum (f_i) \geq -G \right\}
\]

Then we define the code \( \mathcal{C}_L^m(G,D) \) to be:

\[
\{ f(\mathcal{P}) \mid \mathcal{P} \in D^m, f \in \mathcal{L}^m(G) \}
\]

Where, if \( \mathcal{P} = (P_1, \ldots, P_m) \) and \( f = \sum_{\forall i : f_i \in \mathcal{L}(G), \sum(f_i) \geq -G} a_i f_{i_1}(X_1) \cdot f_{i_2}(X_2) \cdots f_{i_m}(X_m) \) (where the coefficients \( a \) are in the base field), Then \( f(\mathcal{P}) : = \sum a_i f_{i_1}(P_1) \cdot f_{i_2}(P_2) \cdots f_{i_m}(P_m) \).

Definition 3.4.2 (Canonical basis). Let \( G \) be a one point divisor, i.e. \( G = r \cdot P \) for some rational place \( P \). A canonical basis for \( \mathcal{L}(G) \) is constructed by considering the series of divisors \( 0, P, 2P \ldots r \cdot P \). Whenever \( l(i \cdot P) > l((i - 1) \cdot P) \), pick a function \( \varphi \in \mathcal{L}(i \cdot P) \varphi \notin \mathcal{L}((i - 1) \cdot P) \) and add it to the basis.

To study degree lifted codes we introduce the following definition which extends the definition of curve degree (Definition 3.3.1) to functions in \( \mathcal{L}^m(G) \):

Definition 3.4.3 (Curve degree of tensored functions). The curve degree of \( \varphi_{i_1}(X_1) \cdot \ldots \varphi_{i_m}(X_m) \) is the sum \( \sum_{j=1}^{m} \deg \mathcal{L} \varphi_{i_j} \). The curve degree of \( f \in \mathcal{L}^m(G) \) is the maximal curve degree amongst its monomials.

3.4.2 Dimension of degree-lifted codes

The dimension of \( \mathcal{C}_L^m(G,D) \) is equal to the number of \( m \)-tuples of basis functions for \( \mathcal{L}(G) \) such that \( \sum_{i=1}^{m} (\varphi_i) \leq -G \). So the exact dimension of such a code would depend upon the gap sequence of \( G \). We can, however establish some bounds:

Theorem 3.4 (Dimension of degree-lifted AG codes). Let \( 2g - 1 \leq \deg G \leq n \). Let \( k \) be the dimension of \( \mathcal{C}_L^m(G,D) \), then:

\[
\binom{\deg G - m(g-2)}{m} \leq k \leq \min \left\{ \binom{\deg G + 1 - g}{m}, \binom{\deg G + m}{m} \right\}
\]

Proof. The first upper bound is the dimension of the tensored AG code (i.e. the \( m \)th power of the dimension of the base code). The second upper bound is what would happen if \( l(G) = \deg G \). The lower bound is the number of \( m \)-tuples if the minimal curve degree of a basis function is \( g \) (the worst case scenario). \( \square \)
3.4.3 A Schwartz-Zippel Lemma for degree-lifted AG codes

To be able to claim that the codes defined above have good distance we prove one of the main results of this chapter, a Schwartz-Zippel type lemma for polynomials over algebraic curves:

**Lemma 3.4.4** (Schwartz-Zippel for degree-lifted AG codes). Let $G \geq 0$ be a one point divisor, $D$ a set of rational places disjoint from the support of $G$. Let $f \in \mathcal{L}^m(G) \setminus \{0\}$. Let $\mathcal{P} = \{P\}_{i=1}^m$ be an $m$-tuple of randomly selected places out of $D$. Then the probability of $f(\mathcal{P})$ being zero is at most $\frac{\deg_G f}{|D|}$.

**Proof.** Fix a canonical basis $\{\varphi_i\}_{i=1}^{l(G)}$ for $\mathcal{L}(G)$. We prove by induction on $m$. Let $m = 1$, then $f$ is a member of $\mathcal{L}(G)$ and therefore a member of the function field. It can only be zero at $\deg_G f$ places (cf. definition 3.3.1).

Now assume the proposition for $m-1$, and consider the case of $m$. Take $f = \sum_{i=1}^k \prod_{j=1}^m f_{i,j}(X_j)$ and represent each $f_{i,j}$ in the canonical basis. Note that any basis function with a non-zero coefficient in this representation cannot have a curve degree higher than $f_{i,j}$’s. So we get a representation of $f$ as a linear combination of products of basis functions. Where each product has a curve degree smaller than $\deg G$. Randomly select a tuple $\mathcal{P} \in D^m$, let $\mathcal{P}^*$ be the first $m-1$ places in $\mathcal{P}$. Denote $f(\mathcal{P}^*) = \sum_{i,j \in [l(G)]} a_{i_1,i_2...i_m} \varphi_{i_1} (P_1) \cdot \varphi_{i_2} (P_2) \cdot \ldots \cdot \varphi_{i_{m-1}} (P_{m-1}) \cdot \varphi_{i_m} (X_m)$ and let $j$ be the largest index of a basis function which appears in $f$ with $X_m$. Then (by induction assumption) with probability at least $1 - \frac{\deg_G f - \deg_G \varphi_j}{|D|}$, $\varphi_j$ has a non-zero coefficient in $f(\mathcal{P}^*)$. This means that $f(\mathcal{P}^*)$ is a linear combination of $\{\varphi_i | 1 \leq i \leq j\}$, with a non-zero coefficient at $\varphi_j$, so it can’t be identically zero (because the basis functions are independent) and, by definition 3.4.2, $\deg_G f(\mathcal{P}^*) \leq \deg_G \varphi_j$ so the probability of $f(\mathcal{P}^*)$ being zero at a random place of $D$ is at most $\frac{\deg_G \varphi_j}{|D|}$. In summary:

$$\Pr[f(\mathcal{P}) = 0] = \Pr[f(\mathcal{P}^*) = 0] + \Pr_{P \in D}[f(\mathcal{P}^*) (P) = 0 | f(\mathcal{P}^*) \neq 0] \leq \frac{\deg_G f - \deg_G \varphi_j}{|D|} + \frac{\deg_G \varphi_j}{|D|} = \frac{\deg_G f}{|D|}.$$

We conclude that the distance of $\mathcal{L}^m(G,D)$ when $G \geq 0$ and $D$ is rational is at least $|D|^m - |D|^{m-1} \cdot \deg G$.

It is natural to ask whether the requirement that $G$ be a 1-point divisor is necessary; unfortunately, it is. As a counter example, consider the rational function field and the Riemann-Roch space $\mathcal{L}(Q_\infty + P_0)$. This is the space spanned by $\frac{1}{x}$, $1$, $x$. Now consider $\mathcal{L}^2(Q_\infty + P_0)$ which is the space spanned by $\frac{1}{x}$, $1$, $x$, $y$, $\frac{1}{y}$, $\frac{x}{y}$. We can evaluate functions in this space on points in $(\mathbb{P}_y^2)^2$ and were the SZ lemma to apply here, we would expect at most $2q - 2$ zeros. Now consider the function $f = \frac{(x-1)(y-2)}{x} - \frac{(x-1)(y-2)}{y}$ which is in $\mathcal{L}^2(Q_\infty + P_0)$. It is zero whenever $y = 2, x = 1$ or $y = x$ for a total of $3q - 6$ zeros, disproving a more general application of the SZ lemma.
3.5 Single-step correction of degree lifted AG codes

In this section we examine the most basic correction algorithm for degree lifted AG codes, in which we correct a point by looking at the restriction of the code to a random automorphism passing through it. We show that the closer a code’s automorphism group is to being two transitive (in the sense of definition 3.5.1) the better an LCC it makes (Theorem 3.5). We will then examine how this test works for degree lifted Hermitian codes (proving corollary 3.7 in Section 3.5.1). In Section 3.5.2 we show that this test works well when the underlying automorphism group is 2-transitive, and in Section 3.5.3 we study a variant of Hermitian codes which is 2-transitive and has a locally decodable tensor.

Let $C_{L}^{2}(G,D)$ be a code, and $\text{Aut}(G,D)$ the group of function field automorphisms stabilizing $G$ and $D$. We study the following procedure:

**Procedure 1**

In order to correct the point $(P_{1},P_{2},\ldots,P_{m})$ in the received message $f'$, do the following:

1. Pick a set of random automorphisms $\sigma_{2},\ldots,\sigma_{m} \in \text{Aut}(G,D)$ such that $\sigma_{i}(P_{1}) = (P_{1})$.

2. Read the values of the message at the points $C = \{(P,\sigma_{2}(P),\ldots,\sigma_{m}(P)) | P \in D\}$.

3. Use a decoding algorithm for AG codes on these values (treating the value at $(P,\sigma_{2}(P),\ldots,\sigma_{m}(P))$ as the value at $P$) and get a function $g \in \mathcal{L}(G)$.

4. Return $g(P_{1})$.

Remark. This is exactly the test used to correct RM codes, where the base code is a RS code and its automorphism group is the affine group.

What do we need for this test to work? At the least, we need $\text{Aut}(G,D)$ to be transitive or there wouldn’t be any $\sigma$ to choose in step 1. In order to prove that this test works, we need that for every point $(P_{1},P_{2})$, the possible samplers for this point cover a significant part of the space nearly uniformly. Formally:

**Definition 3.5.1** ($(\epsilon,\alpha)$-doubly transitive groups). The group $H$ acting on the set $S$ is $(\epsilon,\alpha)$-**doubly transitive** if for every $P_{1} \in S$, there is a subset $S_{P_{1}} \subset S$ with $|S_{P_{1}}| \geq (1-\epsilon) \cdot |S|$. Such that for any $P_{2} \in S$ and $P_{3} \in S_{P_{1}}$ the following holds. Suppose $\sigma$ is chosen uniformly from elements of $H$ mapping $P_{1}$ to $P_{2}$. Then $\sigma(P_{3})$ is distributed uniformly on a subset of $S$ of size at least $(1-\alpha) \cdot |S|$.

**Theorem 3.5** (Single step correction of degree lifted AG codes). Let $C_{L}^{2}(G,D)$ be an $[n,k,d]_{q}$ code such that $\text{Aut}(G,D)$ is $(\epsilon,\alpha)$-doubly transitive. For any $f \in C_{L}^{2}(G,D)$, point of correction $(P_{1},P_{2},\ldots,P_{m}) \in D^{m}$ and any $\delta$-fraction of errors, procedure 1 succeeds with probability at least $1 - \frac{2|D|}{d} \left( \frac{\delta}{(1-\alpha)m-1} + \epsilon \right)$.
Proof. Fix the point \((P_1, \ldots, P_m) \in D^m\) we want to correct. Fix the received word \(f' : D^m \to \mathbb{F}\). Fix the set \(T \subset D^m\) with \(|T| \leq \delta \cdot |D|^m\) of 'errors' where \(f'\) differs from the original codeword \(f\). As above we assume for \(2 \leq i \leq m\), that \(\sigma_i\) is chosen uniformly from the elements of \(\text{Aut}(G, D)\) mapping \(P_1\) to \(P_i\). For \(P \in D\) we define the random variable \(X_P\) which is one if \((P, \sigma_2(P), \ldots, \sigma_m(P)) \in T\), and 0 otherwise. Let \(X \equiv \sum_{P \in D} X_P\) be the fraction of errors on the 'curve' \(C \equiv \{(P, \sigma_2(P), \ldots, \sigma_m(P)) | P \in D\}\). It is easy to see that when \(P\) is chosen uniformly in \(S_{P_1}\), \((P, \sigma_2(P), \ldots, \sigma_m(P))\) is uniformly distributed on a subset of size at least \((1 - \epsilon) \cdot (1 - \alpha)^{m-1} |D|^m\). (Since, after choosing \(P\) each co-ordinate is independent and uniform on a subset of size at least \((1 - \alpha) \cdot |D|\). So the probability that such a point would be in \(T\) is at most

\[
\frac{\delta \cdot |D|^m}{(1 - \epsilon) \cdot (1 - \alpha)^{m-1} |D|^m} \leq \frac{\delta}{((1 - \epsilon) \cdot (1 - \alpha)^{m-1})}.
\]

In other words

\[
\Pr_{P \in S_{P_1}}(X_P = 1) \leq \frac{\delta}{(1 - \epsilon) \cdot (1 - \alpha)^{m-1}}.
\]

Note that

\[
\Pr_{P \in S_{P_1}}(X_P = 1) = \frac{1}{|S_{P_1}|} \sum_{P \in S_{P_1}} \Pr(X_P = 1)
\]

and

\[
E(X) = \sum_{P \in D} \Pr(X_P = 1) = \sum_{P \in S_{P_1}} \Pr(X_P = 1) + \sum_{P \notin S_{P_1}} \Pr(X_P = 1) \leq \frac{|S_{P_1}| \cdot \delta}{(1 - \epsilon) \cdot (1 - \alpha)^{m-1}} + |D| \cdot \epsilon \leq |D| \cdot (\delta/(1 - \alpha)^{m-1} + \epsilon).
\]

Thus, by Markov the probability of having more than \(\frac{d}{2}\) errors on the \(C\) is at most \(\frac{2}{7} \cdot |D|/(\delta/(1 - \alpha)^{m-1} + \epsilon)\).

For any \(z\) in the function field and function field automorphism \(\varphi\), \((\varphi(z)) = \varphi((z))\) so for any fixed \(\sigma_2, \ldots, \sigma_m\) the restriction of a codeword \(f\) to \((P, \sigma_2(P) \ldots \sigma_m(P))\) is a codeword \(g\) of \(C_L (G, D)\). This codeword will be retrieved correctly in Step 3 of the procedure if there are at most \(\frac{d}{2}\) errors on \(C\), and we have bounded the probability of this not happening above.

In Section 3.5.1 we’ll show that this test works on Hermitian codes, by proving the following theorem:

**Theorem 3.6** (Closeness of Hermitian automorphisms to doubly transitive). *The automorphism group of Hermitian codes is \(\left(\frac{1}{q}, \left(1 - \frac{1}{q}\right)\right)\)-doubly transitive.*

From Theorems 3.5,3.6 we derive the following corollary:

**Corollary 3.7** (Single step correction of degree lifted Hermitian codes ). *Procedure 1 works on \(C_L^n (r \cdot Q_\infty, N_H)\) with probability at least \(\frac{2 \cdot q^3}{q^{n-r}} \left(q^{m-1} \delta + \frac{1}{r^2}\right)\).*
We can see that if \((\alpha > 0)\) \(\delta\) must decrease as \(m\) increases. This isn’t satisfactory and we’ll solve this problem in several ways in the next sections.

What happens if \(\alpha = 0\) (i.e. the base code is 2-transitive)? In that case we get local correction for any \(m\), which we’ll prove in Section 3.5.2.

By adding some redundancy to the base code we can increase \(\alpha\), we examine a method for this in Section 3.5.3 and get a version of Hermitian codes which is locally correctable for larger \(m\)’s.

### 3.5.1 Single-step correction of Hermitian codes

We want to prove Theorem 3.6:

**Theorem** (Closeness of Hermitian automorphisms to doubly transitive). The automorphism group of Hermitian codes is \(\left(\frac{1}{q^2}, 1 - \frac{1}{q}\right)\)-doubly transitive.

This will imply (corollary 3.7) that the degree lifting of Hermitian codes is locally correctable from some small fraction of errors.

We need to examine the automorphisms taking \((x_1, y_1)\) to \((x_2, y_2)\) and how they act on some place \((x_3, y_3)\). The next two lemmas establish the needed properties.

**Lemma 3.5.2** (Automorphisms from \((x_1, y_1)\) to \((x_2, y_2)\)). There are \(q^2 - 1\) automorphisms taking \(x_1, y_1\) to \(x_2, y_2\).

**Proof.** Let \(\sigma_{x,y}\) be the automorphism taking \((0, 0)\) to \((x, y)\) and \(\tau\) a generator for the automorphism group that stabilizes \((0, 0)\) then the set \(\{\sigma_{x_2,y_2} \tau^j \sigma_{x_1,y_1}^{-1} | 1 \leq j \leq q^2 - 1\}\) is a set of \(q^2 - 1\) automorphism taking \((x_1, y_1)\) to \((x_2, y_2)\) and since \(|\text{Aut}(rQ_\infty, NH)| = q^3 (q^2 - 1)\) and \(|D| = q^3\) these are the only such automorphisms.

**Lemma 3.5.3** (Automorphisms rarely intersect more than once). Let \(\varphi_1, \varphi_2\) be two different automorphisms taking \((x_1, y_1)\) to \((x_2, y_2)\), if \(\varphi_1\) \((x, y) = \varphi_2\) \((x, y)\) then \(x = x_1\).

**Proof.** Let \(\tau\) be a generator of the automorphism group \(W\) (see theorem 3.2), \(\varphi_1 = \sigma_{x_2,y_2} \tau^j \sigma_{x_1,y_1}^{-1}\) and \(\varphi_2 = \sigma_{x_2,y_2} \tau^j \sigma_{x_1,y_1}^{-1}\). Then \(\sigma_{x_2,y_2} \tau^j \sigma_{x_1,y_1}^{-1} (x, y) = \sigma_{x_2,y_2} \tau^j \sigma_{x_1,y_1}^{-1} (x, y)\).

If \(x \neq x_1\) this means that there is a point \((x', y'), x' \neq 0\) such that \(\tau^i (x', y') = \tau^j (x', y')\) w.l.g let \(j > i\) then \((x', y') = \tau^{j-i} (x', y')\) so \(\tau^{j-i}\) stabilizes a non-zero point. But then, if we denote \(\tau^{j-i} (x) = \tau_c (x)\) \(x' = cx'\) so \(c = 1\) (since \(x' \neq 0\)) and \(\tau^{j-i}\) is the identity mapping in contradiction to \(\tau\) being of order \(q^2 - 1\).

So we have that for all but \(q\) places (those having \(x_3 = x_1\)), the image of \((x_3, y_3)\) under automorphisms taking \((x_1, y_1)\) to \((x_2, y_2)\) is of size \(q^2 - 1\) (so uniformity is immediate). Taking \(S_{P_1} = \{(x, y) \in NH | x \neq x_1\}\) this proves Theorem 3.6. Explicitly, this gives us that for \(r = q^3 - c\alpha q^4\), procedure 1 fails on the code \(C_E^2 (\tau \cdot Q_\infty, NH)\) with probability at most \(\frac{2}{q^2} (\delta + \frac{1}{q^4})\).

So,
Corollary 3.8 (Degree lifted Hermitian codes are single step correctable).

\[ C^2_L \left( (q^3 - ceq^4) \cdot Q_\infty, D \right) \]

is a locally correctable code with rate

\[ \frac{k}{n} \approx \frac{(q^3 - ceq^4)^2 - 2(q^3 - ceq^4)q^2}{2q^6} \approx \frac{1}{2} - c\epsilon q - \frac{1}{q}, \]

query complexity \( \sqrt{n} \) and alphabet size of \( \sqrt[3]{n} \).

We note that the Reed-Muller code with the similar rate of \( \frac{1}{2} - \epsilon \) has an alphabet size of \( \sqrt{n} \) (the query complexity is the same).

3.5.2 Degree lifting of 2-transitive AG codes are LDCs

If our base code is 2-transitive then procedure 1 samples the whole space. What is left is to establish the uniformity of this sampling. In this section we show that if a group is \((\epsilon, 0)\)-doubly transitive (i.e. 2-transitive) then \( \epsilon = 0 \) which will imply that the degree lifting of doubly transitive AG codes is locally correctable (Theorem 3.9).

Remark. Technically, a 2-transitive group won’t sample the whole space since, for instance, when correcting \((P_1, P_2)\) we will never sample \((P_1, P_3)\). We can overcome this by also considering the parallel lines (i.e. \{\((P_1, P) \mid P \in D\) etc) as views.

In the next two lemmas, we show that uniformity is a property of any 2-transitive group.

Lemma 3.5.4 (Uniformity of transitive groups). Let \( A \) be a set with a group \( \Phi \) acting on it transitively. Let \( \Phi_{a,b} \subseteq \Phi \) be the set of automorphisms taking \( a \) to \( b \), then \( |\Phi_{a,b}| \) is independent of the choice of \( a, b \).

Proof. Assume by contradiction that \( |\Phi_{a,b}| > |\Phi_{a,c}| \). Let \( \sigma \in \Phi \) be such that \( \sigma(b) = c \) then \( \sigma(\Phi_{a,b}) \subseteq \Phi_{a,c} \) so there are two different \( \varphi_1, \varphi_2 \in \Phi_{a,b} \) such that \( \sigma \circ \varphi_1 = \sigma \circ \varphi_2 \Rightarrow \varphi_1 = \varphi_2 \), a contradiction.

Lemma 3.5.5 (Uniformity of doubly transitive groups). Let \( A \) be a set with a group \( \Phi \) acting on it 2-transitively. Let \( \Phi_{a,b,c,d} \subseteq \Phi, a \neq c, b \neq d \) be the set of automorphisms taking \( a \) to \( b \) and \( c \) to \( d \), then \( |\Phi_{a,b,c,d}| \) is independent of the choice of \( a, b, c, d \).

Proof. From lemma 3.5.4 we know that \( |\Phi_{a,b}| \) is independent of \( a, b \), so assume by contradiction that \( |\Phi_{a,b,c,d}| > |\Phi_{a,b,c,d'}| \). Let \( \sigma \in \Phi \) be such that \( \sigma(b) = b, \sigma(d) = d' \) then \( \sigma(\Phi_{a,b,c,d}) \subseteq \Phi_{a,b,c,d'} \) so there are two different \( \varphi_1, \varphi_2 \in \Phi_{a,b,c,d} \) such that \( \sigma \circ \varphi_1 = \sigma \circ \varphi_2 \Rightarrow \varphi_1 = \varphi_2 \), a contradiction.

We can now prove the main result of this section.
Theorem 3.9 (Doubly Transitive AG codes are locally correctable). Let $C_L(G, D)$ be an $[n, k, d, q]$ code with a 2 transitive group of function field automorphisms, then $C_L^m(G, D)$ is locally correctable from a $\delta < \frac{d}{2n}$ fraction of errors.

Proof. set $\alpha = 0$ and $\epsilon = 0$ in Theorem 3.5.\hfill \Box

We are unaware of any 2-transitive AG codes (other than Reed-Solomon codes). Some immediate places to look for them are the Hermitian, Suzuki and Ree curves, as all of them have 2-transitive groups acting on their rational points ([Sti93], [HS90] and [Ped92] respectively). However, for these to give rise to 2-transitive codes there must also be a non-rational divisor of sufficiently low degree (smaller than the number of rational points) which is stabilized by all these automorphisms.

Would this kind of test work for the tensor of any 2-transitive code? The uniformity of the sampling has nothing to do with the base code being an AG code, the only property of AG codes we use is that the restriction of the tensor to the sampler is a codeword of the base code. We do not know whether this is true for the tensor (or some appropriate generalization of the degree lifted subcode of it) of some general 2-transitive code.

3.5.3 Increasing transitivity via redundancy

To get a 2-transitive AG code, we need an automorphism group that acts 2-transitively on some set of places, while stabilizing another (that can’t be of too high a degree). While we have several examples with a 2-transitive action, we don’t know of any (except for RM codes) where it also stabilizes another divisor. In particular, the Hermitian function field has a 2-transitive action on its set of rational places (including infinity), but we don’t know of any low degree (degree lower than $q^3$) divisor that is stabilized by it.

In this section we show that a 2 transitive action is sufficient to show local correctability of the tensored code (though with a smaller rate than we would get from degree lifting), this gives us a construction of an LCC based on tensored Hermitian codes.

We first need to slightly extend our notion of AG codes.

Let $S = \{P\}_i^n$ be a set of places and Aut $\{S\}$ be a group of function field automorphisms acting on $S$ (implied here is that $S$ is closed under the action of function from Aut$\{S\}$). We define the evaluation of a function at a place it has a pole in as 0.

Definition 3.5.6. The extended evaluation of a function $z \in F/K$ is denoted $\bar{z}$ and defined

$$\bar{z}(P) = \begin{cases} z(P) & \text{if } v_P(Z) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

This allows us to consider AG codes $C_L(G, D)$ when $\text{SUPP}(G) \cap \text{SUPP}(D) \neq \emptyset$. 

30
Definition 3.5.7. The code \( \tilde{C}(N,D) := \{(\tilde{z}(P_i))_{i=1}^n | z \in L(D)\} \)

We can now prove the main result of this section:

Theorem 3.10 (Local correction of increased transitivity AG codes). Let \( S \) be a set of place such that \( \text{Aut}\{S\} \) acts 2-transitively on \( S \), pick \( Q \in S \). Then \( \tilde{C}^m(L,r:Q,S) \) is locally correctable from a \( \delta \leq \frac{n-r-m}{2n} \) fraction of errors.

Proof. Mark \( |S| = n \). Let \( \delta n^m \) be the number of errors allowed. Say we wish to correct the codeword \( w \) at the point \( (P_1, P_2, \ldots, P_m) \). We randomly pick automorphisms \( \varphi_2, \ldots, \varphi_m \) such that \( \varphi_i(P_i) = P_i \), and consider the values received at the points \( C = \{(P, \varphi_2(P) \ldots \varphi_m(P)) | P \in S\} \). For all \( z \) in the function field: \( (\varphi(z)) = \varphi((z)) \) so \( \deg_{C} z = \deg_{C} \varphi(z) \). Which means that the restriction of \( w \) to \( C \) is a codeword of \( \tilde{C}(r:Q+r \cdot \sum_{i=2}^m \varphi_i(Q), S) \), this code has distance of at least \( d = n - m - r \cdot m \) and we can use standard algorithms to decode it. For any point \( (P_1', \ldots, P_m') \), \( P_i' \neq P_i \) the number of automorphisms selections which would lead to it being in \( C \) is independent of the identity of the \( P' \)'s (lemma 3.5.5) so the expected fraction of errors in \( C \) is \( \delta \).

Let \( X_C \) be a random variable denoting the fraction of errors on \( C \). Using markov
\[
\Pr\left[ X_C > \frac{d}{2} \right] < \frac{2\delta n^m}{4},
\]
so if \( \delta \) is smaller than half the relative distance of
\[
\tilde{C}(r:Q+r \cdot \sum_{i=2}^m \varphi_i(Q), S)
\]
the correction succeeds with high probability. \( \square \)

The disadvantage of this method is the presence of \( r \) in the distance of the restrictions. This means that \( r < \frac{n}{m} \). We also gain no benefit from bounding the total curve degree of monomials in this construction. So we bound the individual curve degrees of members in monomials instead. i.e. we look at the full tensoerd code and not at a sub-code of it. This gives us a dimension of \( (r-g)^m \).

Corollary 3.11 (Locally correctable tensors of Hermitian codes). The \( m \)-th tensor of \( \tilde{C}(r:Q_{\infty}, N_H \cup Q_{\infty}) \) is locally correctable from a \( \delta \leq \frac{d^2-r \cdot m}{2r} \) fraction of errors. The rate of these codes will be \( \frac{1}{m^m} \left( 1 - c\delta - \frac{1}{q} \right) \).

Proof. The set of rational points of the Hermitian curve has a 2-transitive group of automorphisms acting on it [Sti93]. Now apply Theorem 3.10. \( \square \)

Remark. Theorem 3.10 can be extended for \( (\epsilon, \alpha) \)-doubly transitive groups.

3.6 Fractal correction of degree lifted AG codes

In this section we will examine a fractal correction algorithm, define the required properties for a degree lifted code to be locally correctable using this algorithm (Theorem 3.12), show that Hermitian codes posses these properties and conclude that fractal correction succeeds on degree lifted Hermitian codes (corollary 3.15). Let \( C^m(L,G,D) \) be a
lifted AG code, and let $U$ be a subgroup of $\text{Aut}(G,D)$. The \textit{fractal correction procedure} for $C_m^L(G,D)$ and $U$ is as follows.

\textbf{Procedure 2:}
In order to correct the point $(P_1, \ldots, P_m)$ do the following:

1. Choose $m - 1$ random automorphisms $\sigma_2, \ldots, \sigma_m \in U$, under the constraint that $\sigma_i(P_1) = (P_i)$ for every $2 \leq i \leq m$.

2. Let $C = \{(P, \sigma_2(P) \ldots \sigma_m(P)) \mid (P) \in D\}$ be the embedding of the curve generated by these automorphisms.

3. For every point in $C$, apply procedure 1.

4. Use the values returned from step 3, apply standard AG decoding to get the restriction of $f$ to $C$, and calculate it at the point $P_1$.

So in this procedure, we first correct each point on $C$ and then use the corrected values to correct the value at the original point we wanted to correct.

\textbf{Definition 3.6.1} ($(\alpha, \epsilon)$-closeness to 2-steps uniformity). The group $H$ acting on the set $S$ is $\alpha, \epsilon$-\textit{close to 2-steps uniform} if for every $P \in S$, there exists a subset $S_P \subset S$ with $|S_P| \geq (1 - \epsilon) \cdot |S|$, such that

- for any $Z \in S_P$, there exists a subset $S_{P,Z} \subset S$ (depending only on $P$ and $Z$) with $|S_{P,Z}| \geq (1 - \epsilon) \cdot |S|$ such that the following holds.
  - Fix any $P' \in S_P$.
  - Choose random $\sigma, \psi \in H$ under the constraints that $\sigma(P) = P'$, and $\psi(Z) = \sigma(Z)$.

Then for any $Z' \in S_{P,Z}$, $\phi(Z')$ is $\alpha$-close to the uniform distribution on $S$.

\textbf{Remark.} It is easy to see that the above procedure and definition can be generalized to ones in which we do a ‘depth-c recursion’. This could be useful in lifted codes where it would require such a depth of choosing automorphisms to generate a near-uniform point.

\textbf{Theorem 3.12} (Fractal correction of degree lifted AG codes). Let $C_L(G,D)$ be an $[n,k,d]_q$ code such that $\text{Aut}(G,D)$ is $\alpha, \epsilon$-close to 2-steps uniform. For any $f \in C_L^m(G,D)$, point of correction $(P_1, P_2 \ldots P_m) \in D^m$ and any $\delta$-fraction of errors. Procedure 2 succeeds with probability at least $1 - \left(\frac{2n}{q}\right)^2 (\delta + m \cdot \alpha + \epsilon)$

\textbf{Proof.} Fix the point $(P_1, \ldots, P_m)$ that is to be corrected. Let $T \subset D^m$ be the set of errors of size $|T| \leq \delta \cdot n^m$ where the received word differs from the original codeword $f$. As described in Procedure 2, choose random $\sigma_2, \ldots, \sigma_m$ such that $\sigma_i(P_1) = P_i$. Procedure 2 now runs Procedure 1 on each point $(Z, \sigma_2(Z), \ldots, \sigma_m(Z))$ for $Z \in D$. This corresponds to choosing random $\psi_2, \ldots, \psi_m \in U$ under the constraint that $\psi_i(Z) = \sigma_i(Z)$
for $2 \leq i \leq m$, and then using the hermitian code decoding algorithm on the points of the 'curve'

$$C_Z \triangleq \{(Z', \psi_2(Z'), \ldots, \psi_m(Z'))|Z' \in D\}.$$  

For $Z \in D$, define a random variable $X_Z$ that is 1 when $C_Z$ contains more than $d/2$ errors, i.e., $|C_Z \cap T| > d/2$, and 0 otherwise. In a similar argument to the proof of Theorem 3.5, it follows that when $X_Z = 0$ then Step 3 of the procedure will correctly retrieve $f(Z, \sigma_2(Z), \ldots, \sigma_m(Z))$. Let us fix $Z \in S_{P_1,Z}$. In this case, for $Z' \in S_{P_1,Z}$ the coordinates of the point $(Z', \psi_2(Z'), \ldots, \psi_m(Z'))$ are $\alpha$-close to uniform (here is where we use Definition 3.6.1), and therefore by 3.6.3 $(Z', \psi_2(Z'), \ldots, \psi_m(Z'))$ is $m \cdot \alpha$-close to uniform in $D^m$. Hence, for $Z' \in S_{P_1,Z} \cdot |T|$ is at most $\delta + m \cdot \alpha$. Since $|S_{P_1,Z}| \geq (1 - \epsilon) \cdot n$, we have that the expected number of errors on $C_Z$ is at most $\left(\delta + m \cdot \alpha + \epsilon\right)n$. So

$$\Pr(X_Z = 1) \leq \frac{2}{d} (\delta + m \cdot \alpha + \epsilon) n.$$

Similarly, as $|S_{P_1}| \geq (1 - \epsilon) \cdot n$, we now have that the expected number of errors on $C$ is at most

$$((1 - \epsilon)(\delta + m \cdot \alpha + \epsilon)n + \epsilon)n \leq (\delta + m \cdot \alpha + \epsilon)n^2.$$

Using Markov, we get the required bound on the error probability.

This does give us a way to locally correct codes with $m > 2$. We now return to the automorphisms of the Hermitian code, the remainder of this section will be concerned with proving the following:

**Theorem 3.13** (2-steps uniformity of Hermitian automorphisms).

$$\text{Aut}(r \cdot Q_\infty, N_H)$$

is $1/q^2 \cdot 1/q + 2/q^2$-close to 2-steps uniform.

**Corollary 3.14** (Fractal correction of degree lifted Hermitian codes). Consider

$$C_L^m \left(\left(q^3 - c\left(\delta + \frac{m}{q}\right) q^3\right) \cdot Q_\infty, D\right)$$

, for some constant $c$, if the fraction of errors in the received codeword is smaller than $\delta$ then procedure 2 succeeds with probability at least $1 - \frac{8}{\left(\delta + \frac{m}{q}\right)c^2}$.

**Corollary 3.15** (Fractal correction degree lifted Hermitian codes ).

$$C_L^m \left(\left(q^3 - c\left(\delta + \frac{m}{q}\right) q^3\right) \cdot Q_\infty, D\right)$$

33
is a locally correctable code with rate
\[
\frac{k}{n} \approx \frac{(q^3 - c \left( \delta + \frac{m}{q} \right) q^3 m - m(q^3 - c \left( \delta + \frac{m}{q} \right) q^3 )^{m-1} q^2 }{m! \cdot q^{3m}} \approx \frac{1}{m!} \left( 1 - c\delta - \frac{m}{q} \right)
\]
query complexity \( \sqrt{n^2} \) and alphabet size of \( \frac{1}{5n} \).

We note that the Reed-Muller code with the similar rate of \( \frac{1}{m} (1 - \delta) \) has a query complexity of \( \sqrt{n} \) and an alphabet size of \( \sqrt{n} \). This is an immediate consequence of Theorem 3.13. To prove this theorem we’ll first need some preparatory lemmas.

**Lemma 3.6.2** (Image of bounded polynomial is uniform). A degree \( q + 1 \) polynomial \( f : \mathbb{F}_{q^2} \to D \), \( |D| = q \) has a statistical difference of at most \( \frac{1}{q} - \frac{1}{q^2} \) from the uniform distribution on \( D \).

**Proof.** Let \( Q \) be the distribution induced by \( f \), assume that \( \delta (Q, U) > \frac{1}{q} - \frac{1}{q^2} \). Then there is a set \( A \subseteq D \) such that \( \left| \Pr_{x \in \mathbb{F}_{q^2}} [ f(x) \in A] - \frac{|A|}{q} \right| > \frac{1}{q} - \frac{1}{q^2} \). This means that one of the following 2 things must happen:

1. \( f \) takes more values to \( A \) then the uniform distribution does, in that case \( |f^{-1}(A)| \geq |A| q + q \) but \( |f^{-1}(x)| \leq q + 1 \) (because of its degree) so \( |A| = q \) which is ludicrous.

2. \( f \) takes less values to \( A \) then the uniform distribution does. Apply 1. to \( \bar{A} \) to get a contradiction. \( \square \)

**Lemma 3.6.3** (Statistical distance is additive). Let \( Q \) be a distribution on \( D \) and \( P \) a distribution on \( D' \) which is independent of \( Q \). \( P \) and \( Q \) are \( \mu \)-close to uniform. Then \( Q \times P \) is at least \( 2\mu \)-close to uniform.

**Proof.** Statistical distance obeys the triangle inequality so
\[
d(Q \times P, U_D \times U_{D'}) \leq d(Q \times P, Q \times U_{D'}) + d(Q \times U_{D'}, U_D \times U_{D'}) = 2\mu
\]

**Proof.** (of Theorem 3.13)

We think points \( P_1, P_2 \in N_H \). We denote \( P_1 = (x_1, y_1) \) and \( P_2 = (x_2, y_2) \). We define \( S_{P_1} \subseteq N_H \) to be the set of points in \( N_H \) whose first coordinate is different from \( P_1 \). That is,
\[
S_{P_1} \triangleq \{ Z = (z_1, z_2) \in N_H | z_1 \neq x_1 \}.
\]

Note that \( |S_1| = q^3 - q^2 = (1 - 1/q) \cdot |N_H| \) as required in Definition 3.6.1. We now fix a point \( Z = (z_1, z_2) \in S_{P_1} \). We now wish to choose random elements \( \sigma, \psi \in U \) under the constraint that

1. \( \sigma(P_1) = P_2 \).
2. \( \psi(Z) = \sigma(Z) \).

Using the representation of automorphisms described in Corollary 3.3, we denote \( \sigma = \varphi_{\alpha, \beta, c} \) and \( \psi = \varphi_{\alpha', \beta', c'} \). Recall this means that, for example for \( \sigma \), for any \( (x, y) \in N_H \),

\[
\sigma(x) = cx + \alpha, \quad \sigma(y) = cq^2y + a^2cx + \beta.
\]

Using this notation, and the second item of Corollary 3.3, the above constraints are equivalent to the following. For any fixed \( Z' = (x, y) \in N_H \):

\[
\begin{align*}
\sigma(x) &= (x - x_1) \cdot c + x_2 \\
\sigma(y) &= (y - x_1^q x + x_1^{q+1} - y_1) \cdot cq^3 + x_2^q (x - x_1) \cdot c + y_2.
\end{align*}
\]

\[
\begin{align*}
\psi(x) &= (x - z_1) \cdot c' + \sigma(z_1) \\
\psi(y) &= (y - z_1^q x + z_1^{q+1} - z_2) \cdot cq^{q+1} + \sigma(z_1)^q (x - z_1) \cdot c' + \sigma(z_2)
\end{align*}
\]

Combining them we get:

\[
\begin{align*}
\psi(x) &= (x - z_1) \cdot c' + \sigma(z_1) \\
\psi(y) &= (y - z_1^q x + z_1^{q+1} - z_2) \cdot cq^{q+1} + ((x - x_1) \cdot c + x_2)^q (x - z_1) \cdot c' \\
&\quad + (z_2 - x_1^q z_1 + x_1^{q+1} - y_1) \cdot c^q + x_2^q (z_1 - x_1) \cdot c + y_2
\end{align*}
\]

We emphasize that we are thinking now of \( x, y, x_1, y_1, x_2, y_2, z_1, z_2 \) as fixed, while \( c \) and \( c' \) are chosen uniformly\(^6\) in \( F_q ^2 \). As \( z_1 \neq x_1 \) this implies that \( \psi(x) \) is uniformly distributed. Now let us fix \( \psi(x) \) to a value \( u \). So, we have \( u = (x - z_1) \cdot c' + (z_1 - x_1) \cdot c + x_2 \). Rearranging terms, we express \( c' \) as

\[
c' = \frac{- (z_1 - x_1) \cdot c - x_2 + x_T}{(x - z_1)}.
\]

Now, we want to look at the distribution of \( \psi(y) \) given this fixing. Using this setting of \( c' \) we get that \( \psi(y) \) is a polynomial in \( c \) of degree (at most) \( q + 1 \). Furthermore, it is a polynomial whose image is in a set of size \( q \) - the set \( \{ b \in F_q ^2 \mid (u, b) \in N_H \} \). Hence, if it was a non-zero polynomial, Lemma 3.6.2 would imply that \( \phi(y) \) is \( \frac{1}{q} \)-close to the uniform distribution on this set. It would then follow that \( (\psi(x), \psi(y)) \) is \( \frac{1}{q} \)-close to the

\(^6\)Actually, as we defined \( U', c \) and \( c' \) vary uniformly in \( F_q ^2 \), but in our decoding algorithm we can allow also \( c = 0 \). This will add to our set of functions, for every \( (\alpha, \beta) \in N_H \), the constant function mapping all of \( N_H \) to \( (\alpha, \beta) \). Alternatively we could slightly increase the resulting \( \alpha \) and \( \epsilon \) in Theorem 3.13.
uniform distribution on $N_H$.

So, what is left to do is to show that for most fixings of $Z = (x, y)$, $\psi(y)$ is a non-zero polynomial in $c$. The coefficient of $c^{q+1}$ in this polynomial is $\left(z_2 - x_1^q z_1 + x_1^{q+1} - y_1\right) + \left(y - z_1^q x + z_1^{q+1} - z_2\right) \left(\frac{(x-z_1)}{(x-z_1)}\right)^{q+1}$.

For fixed $P = (x_1, y_1), Z = (z_1, z_2)$, when $x \neq z_1$, this will be zero exactly when

$$h(x, y) \triangleq (z_2 - x_1^q z_1 + x_1^{q+1} - y_1) \cdot (x - z_1)^{q+1} + \left(y - z_1^q x + z_1^{q+1} - z_2\right) (z_1 - x_1) = 0.$$  

$h(x, y)$ can have at most $q^2 + q$ zeros on the hermitian curve. So defining $S_{P, Z} \triangleq \{Z' = (x, y) \in N_H|x \neq z_1, h(x, y) \neq 0\}$, we are done. \hfill \square

3.7 Correction via high-degree samplers

In this section we examine correction via high-degree samplers. We define the required properties of a base AG code for its degree lifting to be locally correctable in this manner (Theorem 3.16) and construct an explicit set of high-degree samplers for degree lifted Hermitian codes which allows us to locally correct them (Theorem 3.18).

So far we have used automorphisms in our correcting, this had the benefit of having the restriction of the degree lifted code be a word in the base code. The downside of this is that the number of automorphisms isn’t as large as we would like it to be. In the section we replace the automorphisms with a larger class of functions that will allow us to easily sample the whole space but with some increase in the curve degree of the restrictions of a code-word to a sampler.

Definition 3.7.1. A generalized automorphism of a function field is a function $f : F/K \rightarrow F/K$ such that for all $P \in \mathcal{P}_F$ the set $f(P)$ is contained in only one place of $F/K$. We define $f(P)$ to be that unique place.

Definition 3.7.2 $((l, t)$-samplers). The set of generalized automorphisms $\Phi$ is an $(l, t)$-sampler for the code $C_L(G, D)$ if there is a subset $A \subseteq L(G)$ of dimension $\frac{l(G)}{l}$ such that for all $f \in \Phi$ and $g \in A$, $\deg_G g + l \geq \deg_C f(g)$ and $\Phi$ has a $(0, 0)$-doubly transitive action on $D$.

Theorem 3.16 (High degree correction of degree lifted AG codes). If $C_L(G, D)$ is an $[n, k, d]_q$ code which has an $(l, t)$-sampler $ \Phi$ then for any $f \in A^m$, point of correction $(P_1, P_2 \ldots P_m) \in D^m$ and any $\delta$-fraction of errors. Procedure 1 (when choosing functions from $\Phi$) succeeds with probability at least $1 - \frac{25}{d-(m-1)l}$.

Proof. For any $z \in A$ and function $\varphi \in \Phi$, $\deg_G \varphi(z) \leq \deg_G z + l$ so the restriction of a codeword $f$ to $(P, \sigma_1(P), \ldots \sigma_m(P))$ is a codeword of $C_L(G', D)$ for some $G'$ such that $\deg G' \leq (m-1)l + \deg G$. $C_L(G', D)$ has distance $d - (m-1)l$ and so we can handle $\frac{d-(m-1)l}{2}$ errors. By Markov, the probability of having more than $\frac{d-(m-1)l}{2}$ errors is at most $\frac{25}{d-(m-1)l}$. \hfill \square
We will spend the remainder of this section proving the following theorem:

**Theorem 3.17** (Samplers for Hermitian codes). *Hermitian codes have a $\left(\frac{q^3}{t^2}, t\right)$ sampler for any $0 < t \leq 1$*

The corollary of which will be:

**Theorem 3.18** (High degree correction of degree lifted Hermitian codes).

\[
C^m_L \left( \left( \left( 1 - \frac{m}{t} \right) q^3 - c\delta q^3 \right) \cdot Q_\infty, D \right)
\]

has a locally correctable subcode of rate:

\[
k_n \approx \left( \left( 1 - \frac{m}{t} \right) q^3 - c\delta q^3 \right)^m - m \left( \left( 1 - \frac{m}{t} \right) q^3 - c\delta q^3 \right)^{m-1} q^2 \frac{1}{tm!} \left( 1 - \frac{m}{t} - c\delta - \frac{1}{q} \right)
\]

, query complexity $\sqrt{n}$ and alphabet size of $\frac{1}{\sqrt{m}}\sqrt{n}$.

Our set of functions will be:

\[
\varphi(x) = ax + b
\]

\[
\varphi(y) = a^{q+1}y + b^qax + c + T^*(\alpha x + \beta y)
\]

Where $a \in \mathbb{F}_{q^2}^*$, $(b, c) \in N_H$, $\alpha, \beta \in \mathbb{F}_{q^2}$ and $T^*(z) = z^q - z$.

We note that $Tr(T^*(z)) = 0$ so these are indeed generalized automorphisms.

We define $C^m_{L} (r \cdot Q_\infty, N_H)$, to be the sub-code of $C^m_{L} (r \cdot Q_\infty, N_H)$ in which the y-degrees of the evaluated functions are bound by $\frac{q}{t}$. This bounds the potential increase in the curve degree of the functions by an additive factor of $\frac{q^m}{t^m}$, and decreases the dimension by a multiplicative factor of at most $\frac{1}{t}$. All that remains is to show that these functions cover the whole space uniformly. This will be proven in lemma 3.7.6 but we need to do a little work beforehand.

First we show that these function are actually different from one another.

**Lemma 3.7.3** (High degree testers are different). *If

\[
\forall (x, y) \in N_H : \varphi_1(x, y) = \varphi_2(x, y)
\]

then $\varphi_1 \equiv \varphi_2$*

**Proof.** we can immediately derive that $a_1 = a_2$ and $b_1 = b_2$, we are left with the equation $T^*((\alpha_1 - \alpha_2)x + (\beta_1 - \beta_2)y) + c_1 - c_2 = 0$, pick $y_0 \in \mathbb{F}_{q^2}$ s.t. $y_0^q + y = 1$ and set $y = y_0$. This equation then become a degree $q$ polynomial in $x$, but there are $q + 1$ $x$ values for which $(x, y_0)$ is a Hermitian rational point. So this polynomial must have $q + 1$ zeroes, so we get that $\alpha_1 = \alpha_2$ and $c_1 = c_2$. We can then conclude that $\beta_1 = \beta_2$ and $\varphi_1 \equiv \varphi_2$.\qed
Now we show that to correct a particular point we have many possible tests.

**Lemma 3.7.4** (Equal number of functions through each point). Let 
\((x_1, y_1), (x_2, y_2) \in N_H, \) the number of functions for which \(\varphi(x_1, y_1) = (x_2, y_2)\) \(q^4 \cdot (q^2 - 1)\).

**Proof.** Pick any \(a \) \((q^2 - 1 \text{ options})\), there is a single solution to \(ax_1 + b = x_2\) so \(b\) is set.

We need \(y_2 = b'^ax_1 + a'^y_1 + c + T^*(\alpha x + \beta y)\) pick any \(c\) such that \(b'^a = c^a + c\) \((q \text{ options})\), and let \(b'^ax_1 + a'^y_1 + c = \lambda\) then we need \(y_2 - \lambda = T^*(\alpha x_1 + \beta y_1)\), note that \(Tr(\lambda) = N(x_2) = Tr(y_2)\) so \(y_2 - \lambda\) is a trace zero element, denote it by \(\lambda'\).

There are \(q\) elements \(z\) in \(F_{q^2}\) such that \(z^q - z = \lambda'\) \((T^*\text{is an additive homomorphism from } F_{q^2} \text{ onto the trace zero elements})\). So for any choice of \(\alpha\) there are \(q\) options of \(\beta\) such that \(T^*(\alpha x_1 + \beta y_1) = \lambda'\). (the only issue is if \(x_1, y_1 = (0, 0)\). However, in that case, \(\varphi(0, 0) = (b,c)\) regardless of \(a, \alpha, \beta\), so we just set \(b,c = (x_2, y_2)\) and still get the same number of \(\varphi\)’s ) So the number of \(\varphi\)’s going through \((x_1, y_1, x_2, y_2)\) is \((q^2 - 1) \cdot q \cdot q^2 \cdot q\).

So when looking to correct a particular point, we can choose between \(q^4 \cdot (q^2 - 1)\) different testers.

And now we can show that these tests cover nearly the whole space.

**Lemma 3.7.5** (Equal number of functions through each 2 points). Let 
\((x_1, y_1, x'_1, y'_1), (x_2, y_2, x'_2, y'_2)\) be two points on the Hermitian plane such that \(x_1 \neq x_2, x'_1 \neq x'_2\) then there are \(q^3\) functions such that \(\varphi(x_1, y_1) = (x'_1, y'_1), \varphi(x_2, y_2) = (x'_2, y'_2)\)

**Proof.** The conditions on the \(x\)’s yield the equations \(ax_1 + b = x'_1, ax_2 + b = x'_2\) which have a single solution when \(x_1 \neq x_2\).

Pick any \(c\) such that \(Tr(c) = N(b) \) \((q \text{ options})\) and we now get the equations:

\[
b'^a ax_1 + a'^y_1 + c + T^*(\alpha x_1 + \beta y_1) = y'_1
\]

\[
b'^a ax_2 + a'^y_2 + c + T^*(\alpha x_2 + \beta y_2) = y'_2
\]

let \(b'^a ax_1 + a'^y_1 + c = \lambda_1, b'^a ax_2 + a'^y_2 + c = \lambda_2\) and note that \(Tr(\lambda_i) = Tr(y'_i)\) so \(y'_1 - \lambda_1 = \lambda'_1\) and \(y'_2 - \lambda_2 = \lambda'_2\) are both trace zero elements. Pick \(z_1, z_2\) such that \(T^*(z) = \lambda'_1 \) \((q^2 \text{ options})\). The set of equations \(\alpha x + \beta y = z\) has a solution unless \(\exists \gamma : (x_1, y_1) = (\gamma x_2, \gamma y_2)\) this however implies that \(Tr(\gamma y_2) = N(\gamma x_2) \implies (\gamma q^4 - \gamma) = 0\) and so can only happen if \(\gamma \in \{0, 1\}\), \(\gamma = 1\) would mean that both points are equal which is is not an option. If \(\gamma = 0\) then \(x_1 = y_1 = 0\). In that case, instead of picking a random \(c\) we pick one where \(c = y'_1\). We get that \(\varphi(0, 0) = x'_1, y'_1\) for any values of \(\alpha, \beta\). But we still need to get \(T^*(\alpha x_2 + \beta y_2) = \lambda'_2\), for every choice of \(\alpha\) there are \(q\) options for \(\beta\) for a total of \(q^3\) total options.

38
Lemma 3.7.6 (Uniform coverage by high degree functions). The set of functions used to correct any point cover nearly the whole space uniformly.

Proof. For any point \((x_1, y_1, x'_1, y'_1)\) there are \(q^4 (q^2 - 1)\) functions passing through it, pick \((x_2, y_2)\) such that \(x_2 \neq x_1\). For any choice of \(x'_2, y'_2\) such that \(x'_2 \neq x'_1\) (there are \((q^2 - 1)q\) such points), there are \(q^3\) functions passing through \((x_1, y_1, x'_1, y'_1)\) and \((x_2, y_2, x'_2, y'_2)\) so going through all the options for \(x'_2, y'_2\) we cycle through all the functions passing through \((x_1, y_1, x'_1, y'_1)\), which shows that the functions passing through a single point cover almost all of the space in a uniform manner.

The fraction of points unsampled by the functions through a particular point is so small as to not matter. We can then conclude that

\[ C_m \left( \left( 1 - \frac{m}{t} \right) q^3 - c\delta q^3 \right) \cdot Q_\infty, D \] is the subcode for which Theorem 3.18 applies.
Chapter 4

A combinatorial nullstellensatz and probabilistically checkable proofs

4.1 Overview

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 $\text{poly}(t(n))$ 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 $t(n) \cdot \text{poly} \log t(n)$ [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 $t(n) \cdot \text{poly} \log t(n)$ 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 [KSY11b], 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 chapter, we show that for the special case of circuit-SAT, there exists a (non-uniform) PCP verifier that verifies the satisfiability of a circuit of size $n$ by making $n^\varepsilon$ queries to a proof of length $O_{\varepsilon}(n)$ for all $\varepsilon > 0$. Using the efficient reduction of NP to circuit-SAT of [PF79], this implies the existence of a PCP verifier that makes $O(t^\varepsilon)$ queries to a proof of length $O_{\varepsilon}(t \log t)$ for every language $L \in \text{NTIME}(t)$.

**Theorem 4.1 (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 $\text{poly}(n)$ that when given as input the description of a circuit $\phi$ of size $n$, acts as follows:

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

**Remark.** The theorem above can be restated with circuit-SAT replaced by any family of constraint satisfaction problems (CSP) with constant constraint-arity and constant alphabet. This is because there exists a polynomial-time, linear-size, reduction between circuit-SAT and such a CSP.

Again, by combining Theorem 4.1 with the reduction of [PF79], we obtain the following result.

**Corollary 4.2.** For every $\varepsilon > 0$ and time-constructible function $t : \mathbb{N} \rightarrow \mathbb{N}$ the following holds for every language $L \in \text{NTIME}(t)$. For every input length $n$, there exists a randomized oracle circuit $C_n$ of size $\text{poly}(t)$ 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 $\pi_x$ of length $O_{\varepsilon}(t(n) \log t(n))$ such that $C_{n^\varepsilon}(x)$ accepts with probability 1.
3. If $x \notin L$, then for every string $\pi$, it holds that $C_{n^\varepsilon}(x)$ rejects with probability at least $\frac{1}{2}$.
Remark. 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.

4.1.1 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 $\phi$ of circuit-SAT of size $n$:

1. (Graph problem) Reducing $\phi$ to a constraint satisfaction problem (CSP) over an affine Schreier graph $G$ of size $m_1$. An affine Schreier graph has as its vertex set a vector space $V$ and its edge set is generated by a set $S$ of affine transformations, i.e., the set of neighbors of a vertex $v \in V$ is $\{A(v) \mid A \in S\}$. 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 $F$ 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 $H \subset F$. 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 $R \overset{\text{def}}{=} Q \circ P$ 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\(^\ddagger\). Thus, in order to verify that the ACSP instance is satisfiable, we need to be able to verify that a function is close to a low-degree polynomial. This is done in [BS08] by constructing a PCP of proximity (PCPP) [BGH\(^+\)06, DR06] for testing that a polynomial is of low degree. This step uses \(O(\sqrt{n})\) 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 polylogarithmic 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 Schreier graphs with larger degree and larger generating set \(S\), specifically, \(|S| = n^\epsilon\), in which case the routing loses only a constant factor, i.e. \(m'_1 = O_\epsilon(n)\). 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 replace the affine Schreier graphs with ones that are applicable to AG-codes. The reason that earlier constructions used affine Schreier graphs is because the affine group is the automorphism group of the RS- and RM-codes used in those PCP constructions. So we generalize this by working with Schreier graphs generated by a set \(S\) of automorphisms of the corresponding AG-code that is used in the next step.

- 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 solve the zero testing for AG codes by following the ideas of [BS08] and 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 4.7. We discuss this point in some more detail in the next sub-section.

\(^\ddagger\)Actually, those procedures also rely on the promise that some additional auxiliary functions are close to low-degree polynomials.
An alternative method is presented in [BKK+13]. This approach 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 [Mei13]. 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 $n^\varepsilon$. To this end, we use tensor products of the AG codes rather than the AG codes themselves. We then use the fact that tensor codes are locally testable\(^2\) as an analog of low-degree testing.

We 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 [Mei13]), 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 $n^\varepsilon$ while keeping the rate constant. This seems like a very interesting question.

### 4.1.2 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\(^3\) 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 $E = (\lfloor \infty, \ldots, \lfloor 1), \lfloor \in\mathbb{F}_q(\lfloor \infty, \ldots, \lfloor \infty) \mid k \text{ carefully chosen rational equations over } \mathbb{F}_q$

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

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.

\(^2\)The study of local testability of tensor codes was initiated by [BS06] and further studied in [Val05b, CR05b, DSW06b, BV09b, BSV09, GM12, Mei12b, Vid12]. We use the state-of-the-art testability results of [Vid12] (cf. Theorem 4.4)

\(^3\)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.
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 $f, g : \mathbb{F} \to \mathbb{F}$ 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 [Mei12a, Mei13], this “distance of multiplication code” property is sufficient for a PCP-style arithmetization. AG codes are a natural generalization of “low-degree” codes (under the proper definition of “degree”) and, in particular, have the “distance of multiplication code” property needed for PCPs.

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

Remark. We note that error correcting codes with multiplication properties, and in particular AG codes, have been used in many prior works. See further discussion in Section 4.3.2.

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 an affine Schreier graph 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 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 [KS08b] (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].

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, 

$$\frac{n_{i+1}}{n_i} \to \infty.$$
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. An asymptotically good family of transitive AG codes that is “dense”, i.e., \( n_{i+1}/n_i \) is at most an absolute constant \( c \) was presented by Stichtenoth in the Appendix to [BKK+13]. 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 a 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 PCP construction, on the other hand, relies crucially on our codes being AG codes, and in particular on the AG Combinatorial Nullstellensatz to be discussed next. This construction is more randomness-efficient then the sum-check based construction, 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. (4.3)) can be used to solve the zero-testing problem multi-variate polynomials.

Our 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 4.14 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 4.7 for more details.

4.1.3 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 $n \cdot \text{poly log } n$, and our construction gives query complexity of $n^\varepsilon$ 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 $n \cdot \text{poly log}_q n$?

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 4.2 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.

4.1.4 The road ahead

In the next section we formally state our main results. Section 4.3 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 4.4 gives the proof of Main Theorem 4.1. It does so in a succinct manner, leaving the proofs of various reductions to later sections (Sections 4.5–4.7). Of particular importance is Section 4.7 where the AG combinatorial Nullstellensatz is proved.

4.2 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 $\varphi$, denoted $|\varphi|$, is defined to be the number of wires of $\varphi$. We say that a circuit $\varphi$ is satisfiable if there is an input $x \in \{0, 1\}^*$ for $\varphi$ such that $\varphi(x) = 1$.

Definition 4.2.1. The circuit satisfiability problem, circuit-SAT, is the problem of deciding whether a circuit is satisfiable. Formally,

$$\text{circuit-SAT} \overset{\text{def}}{=} \{ \varphi : \varphi \text{ is a satisfiable circuit} \}.$$
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 $\pi$. It is common to define a PCP verifier as an oracle machine that is given oracle access to the proof $\pi$ 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., \cite{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 4.2.2** (Non-uniform PCP verifier, following \cite{BGH+06}). Let $L \subseteq \{0,1\}^*$ be a language, and let $r, q, \ell, d, v, \rho : \mathbb{N} \rightarrow \mathbb{N}$, $\rho : \mathbb{N} \rightarrow (0,1)$. A (non-uniform) PCP verifier $V = \{V_n\}_{n=1}^\infty$ for $L$ with query complexity $q$, proof length $\ell$, rejection probability $\rho$, 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 $|I| \leq q(n)$, and a circuit $\psi : \{0,1\}^I \rightarrow \{0,1\}$ of size at most $d(n)$. For $\pi \in \{0,1\}^{\ell(n)}$ we denote by $\pi|_I$ the restriction of $\pi$ 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

$$\Pr[\psi(\pi|_I) = 1] = 1,$$

where the probability is over $\psi$ 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

$$\Pr[\psi(\pi|_I) = 0] \geq \rho(n),$$

49
where the probability is over $\psi$ and $I$ generated by the verifier $V$ on input $w$.

We can now state our main result.

**Theorem 4.3** (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 $\text{circuit-SAT}_n$ 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 4.3 goes by the following schematic sequence of reductions, explained next.

\[
\text{circuit-SAT} \xrightarrow{(i)} \text{Theorem 4.8} \quad \text{Hypercube-CSP} \xrightarrow{(ii)} \text{Theorem 4.9} \quad \text{aggregate-AGCSP} \xrightarrow{(iii)} \text{Section 4.4.3} \quad \text{Nullstellensatz.} \quad (4.1)
\]

(i) The first reduction maps an instance $\varphi$ of circuit-SAT to a graph constraint satisfaction problem over a sub-graph of the hypercube. This reduction is stated in Section 4.4.1 and proved in Section 4.5. (ii) The next reduction maps the hypercube constraint satisfaction problem into an Algebraic Geometry Constraint Satisfaction Problem (AGCSP). This part is stated in Section 4.4.2 and proved in Section 4.6. We complete the proof of Theorem 4.3 by using the AG combinatorial Nullstellensatz (Theorem 4.7) which is proven in Section 4.4.3.

### 4.3 Tensors of AG Codes and an AG Combinatorial Nullstellensatz

This section starts by reviewing the basic properties of tensor product codes as well as their local testability and concludes by describing the AG codes that we use and our AG Combinatorial Nullstellensatz which pertains to tensors of AG codes.

#### 4.3.1 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 4.3.1.** Let $R : \mathbb{F}^{kn} \rightarrow \mathbb{F}^{\ell_R}$, $C : \mathbb{F}^{kC} \rightarrow \mathbb{F}^{\ell_C}$ 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 4.3.2. 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 $\delta_R$ and $\delta_C$ 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:

Iterated tensor code 4.3.1. Let $C : \mathbb{F}^k C \to \mathbb{F}^\ell C$ be a code. For every $m \in \mathbb{N}$ we denote by $C^\otimes m : \mathbb{F}^{km} \to \mathbb{F}^{\ell m}$ 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 $g : D^m \to \mathbb{F}$.

Axis-parallel lines 4.3.2. For $i \in [m]$ and $\bar{v} = (v_1, \ldots, v_m) \in D^m$ the set

$$D^m|_{i, \bar{v}} = \{(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_m) \mid x \in D\}$$

is called the $i$-axis-parallel line passing through $\bar{v}$. Similarly, the restriction of $g$ to this axis-parallel line, denoted $g|_{i, \bar{v}}$, is the function with range $D$ defined by

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

Using Fact 4.3.2, one can prove by induction the following.

Fact 4.3.3. Let $C$ be a linear code whose codewords are identified with functions $f : D \to \mathbb{F}$. Then, a function $g : D^m \to \mathbb{F}$ is a codeword of $C^\otimes m$ if and only if for every $1 \leq i \leq m$ and $\bar{v} \in D^m$ it holds that the function $g|_{i, \bar{v}}$ is a codeword of $C$.

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

Fact 4.3.4. 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 [Mei13].

Claim 4.3.5 ([Mei13, 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 $m \in \mathbb{N}$. 51
Then, for every coordinate \( x \in D^m \) there exist scalars \( \alpha_{t,z} \in F \) (for every \( 1 \leq t \leq m \) and \( z \in H \)) such that for every codeword \( g \in C^{\otimes m} \) it holds that

\[
g(x) = \sum_{z_1 \in H} \alpha_{1,z_1} \cdot \sum_{z_2 \in H} \alpha_{2,z_2} \cdot \cdots \cdot \sum_{z_m \in H} \alpha_{m,z_m} \cdot g(z_1,\ldots,z_m).
\]

Furthermore, the scalars \( \alpha_{t,z} \) can be computed in polynomial time given \( x \) 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. The “moreover” part in Claim 4.3.5 does not appear in [Mei13], but is implicit in the proof there.

Claim 4.3.6 ([Mei13, Claim 3.8]). Let \( C = \{f : D \to F\} \) be a linear evaluation code, let \( m \in \mathbb{N} \), and let \( g \in C^{\otimes m} \). Then, for every sequence of scalars \( \alpha_{t,z} \) (for every \( 2 \leq t \leq m \) and \( z \in D \)) it holds that the function \( f : D \to F \) defined by

\[
f(z_1) = \sum_{z_2 \in D} \alpha_{2,z_2} \cdot \sum_{z_3 \in D} \alpha_{3,z_3} \cdot \cdots \cdot \sum_{z_m \in D} \alpha_{m,z_m} \cdot c(z_1,\ldots,z_m)
\]

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 4.4 (Testing of tensor codes). There exists a randomized polynomial-time tester that satisfies the following requirements:

- **Completeness:** If \( w \in C^{\otimes m} \), then the tester accepts with probability 1.
- **Size:** The tester uses at most \( \log(\ell^m) + O(\log m) \) random bits and \( \ell^2 \) queries, and performs \( \text{poly}(\ell) \) arithmetic operations.
- **Soundness:** If \( w \notin C^{\otimes m} \), then the tester rejects with probability at least \( \gamma_m \cdot \delta(w,C^{\otimes m}) \), where \( \gamma_m = \delta_3^m / \text{poly}(m) \).

4.3.2 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 4.3. We do so here using a limited amount of algebraic geometry and hence postpone the definitions and proofs to Section 4.7. 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.

52

Technion - Computer Science Department - Ph.D. Thesis PHD-2016-12 - 2016
Definition 4.3.7 (Multiplication codes). Let $C, C'$ be two evaluation codes with the same domain $D$. Define their multiplication $C \cdot C' = \text{span}\{ f \cdot f' \mid f \in C, f' \in C' \}$ 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 $\tilde{C} = (C_1, \ldots, C_{d_{\text{mult}}})$ with the same domain $D$ is called a multiplication code family of multiplication degree $d_{\text{mult}}$ if for all $1 \leq i, j \leq d_{\text{mult}}$ with $i + j \leq d_{\text{mult}}$, we have $C_i \cdot C_j \subseteq C_{i+j}$.

Definition 4.3.8 (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

$$\text{Aut}(C) = \{ \pi : D \to D \mid \{ f \circ \pi \mid f \in L \} = C \}$$

A code $C$ is called transitive if its automorphism group is transitive, i.e., for every $x, y \in D$ there exists $\pi \in \text{Aut}(C)$ such that $\pi(x) = y$.

An asymptotically good family of transitive AG codes was presented in [Sti06]. This family was “sparse”: the ratio between blocklengths of consecutive members in this family was super-constant. Applied to our framework, this family would only have led to a result saying that infinitely often circuit-SAT$_n$ has constant-rate PCPs with sublinear query complexity. The main result presented in the appendix to [BKK+13] 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 4.5. (Asymptotically good transitive AG-codes for every message length) 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 $\rho$ and $\delta$ respectively and a multiplication degree parameter $d_{\text{mult}}$ which satisfy

$$c_q \cdot d_{\text{mult}} \cdot \rho + \delta < 1 - \frac{d_{\text{mult}}}{\sqrt{q-1}} \quad (4.2)$$

Then for every sufficiently large message length $k$ there exists a code $C$, and a multiplication code family $\tilde{C} = (C_1, C_2, \ldots, C_{d_{\text{mult}}})$ 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 $k$, relative distance at least $\delta$, and with length $|D| \leq \frac{1}{\rho} \cdot k$. Succinctly, $C$ is an $[\lfloor D \rfloor, \geq k, \geq \delta |D|_q]$-code.

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

53
Distance of Multiplication codes. For each \( j \leq d_{\text{mult}} \), the code \( C_j \) has relative distance at least \( \delta \).

Remark. [BKK+13] states the above result a bit differently. More specifically, the codes there 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, it states a better trade-off between \( \rho, \delta, d \), for those message lengths, namely,

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

The parameters in Theorem 4.5 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 \( k \).

Previous works on multiplication codes. Codes with various multiplication properties, and AG codes in particular, have been studied and used in many works in the past:

- AG codes with multiplication properties are used in the study of bilinear multiplication algorithms, starting from the work of Chudnovsky and Chudnovsky [CC88].

- Kötter [Köt92] and Pelikaan [Pel92] have shown that the Berlekamp-Welch algorithm can be generalized to codes that exhibit multiplication properties. See also Sudan [Sud01].

- Similar ideas were used for constructing secure multi-party computation protocols, starting from Cramer et al. [CDM00]. AG codes were introduced to this line of work by Chen and Cramer [CC06]. See also [CCG+07, CChHC08, CCCX09, CCX11].

- As mentioned above, in [Mei12a, Mei13], it was shown that codes with multiplication properties could be used to construct interactive proofs and PCPs.

4.3.3 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 4.6** (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
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

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

(4.3)

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

In Section 4.7 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 4.3. Comparing it to the previous theorem one main difference is that we require two auxiliary functions $\xi,\xi'$ as opposed to only one above. The full-generality result is stated in Theorem 4.14.

**Theorem 4.7** (Special case of AG Combinatorial Nullstellensatz). Let $C = \{ f : D \to \mathbb{F} \}$ and $\mathring{C} = (C_1,\ldots,C_{d_{\text{mult}}})$ be the codes from Theorem 4.5 with multiplication degree $d_{\text{mult}} = 6d$, rate parameter $\rho$, and distance parameter $\delta$ satisfying

$$c_q \cdot d_{\text{mult}} \cdot \rho + \delta < \frac{1}{2} - \frac{d_{\text{mult}}}{\sqrt{q} - 1}. \quad (4.4)$$

Then for every $H \subset D$ with

$$|H| < \left( \frac{1}{12} - \frac{\delta}{6} - \frac{2}{\sqrt{q} - 1} \right) \cdot |D|, \quad (4.5)$$

there exist $\xi(X) = \xi_H(X) \in C_{2d}$ and $\xi'(X) = \xi'H(X) \in C_{3d}$ satisfying the following.

Suppose $f(X_1,\ldots,X_m) \in (C_d)^{\otimes m}$. Then $f$ vanishes on $H^m$ if and only if there exist $f'_1,\ldots,f'_m : D^m \to \mathbb{F}_q$, with $f'_i \in (C_{4d})^{\otimes m}$ for each $i \in [m]$, such that:

$$f(X_1,\ldots,X_m) \cdot \prod_{i=1}^{m} \xi'_i(X_i) = \sum_{i=1}^{m} f'_i(X_1,\ldots,X_m) \cdot \xi(X_i). \quad (4.6)$$

In Section 4.7, we show how to instantiate parameters in Stichtenoth’s code construction and in our AG Combinatorial Nullstellensatz to get the precise statements of Theorem 4.5 and Theorem 4.7.
4.4 Proof of Main Theorem 4.3

4.4.1 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 4.8). The proof of this theorem is deferred to Section 4.5. 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 4.4.1 (Constraint graph). A constraint graph $G$ is a graph $(V, E)$ coupled with a finite alphabet $\Sigma$, 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 $\sigma : V \rightarrow \Sigma$. We say that an assignment $\sigma$ satisfies an edge $(u, v) \in E$ if $(\sigma(u), \sigma(v)) \in c_{u,v}$, and otherwise we say that $\sigma$ violates $(u, v)$.

We say that $\sigma$ 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 4.4.2 (Graph CSP). The graph constraint satisfaction problem, graph-CSP, is the problem of deciding whether a constraint graph $G$ is satisfiable. Formally,

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

Definition 4.4.3 (The hypercube graph). The $m$-dimensional $k$-ary hypercube, denoted $H_{k,m}$, 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 4.4.4 (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,

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

We now state the first step in our reduction, its proof appears in Section 4.5.
Theorem 4.8 (From circuit-SAT to Hypercube-CSP). There exists a polynomial time procedure that maps every circuit $\varphi$ of size $n$ and integer $m \in \mathbb{N}$ to a constraint graph $G_{\varphi,m}$ over an alphabet $\Sigma$ of size $4$ that is satisfiable if and only if $\varphi$ 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 $k \leq 4((4 \cdot n)^{1/m} + 1)$.

4.4.2 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 4.8. 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.

**Axis-parallel composition with automorphism 4.4.1.** Let $C = \{f : D \rightarrow \mathbb{F}\}$, let $\pi$ be an automorphism of $C$, and let $g : D^m \rightarrow \mathbb{F}$ be a codeword of the tensor code $C \otimes m$. Then, for each $i \in [m]$, we define the function $g^{\pi,i} : D^m \rightarrow \mathbb{F}$ 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 $g^{(\pi_1, \ldots, \pi_t)} : D^m \rightarrow \mathbb{F}^{1+t \cdot m}$ to be the function obtained by aggregating the $1 + t \cdot m$ functions $g, g^{\pi_1,1}, \ldots, g^{\pi_t,m}$. Formally,

$$
g^{(\pi_1, \ldots, \pi_t)}(\overline{x}) = (g(\overline{x}), g^{\pi_1,1}(\overline{x}), \ldots, g^{\pi_t,m}(\overline{x})).
$$

**Definition 4.4.5 (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 $h : H \rightarrow \mathbb{F}$ to codewords $f : D \rightarrow \mathbb{F}$.

• $\pi_1, \ldots, \pi_t$ are automorphisms of $C_j$ for every $j \in [d]$.

• $Q^{(\psi)} : D^m \times \mathbb{F}^{1+t\cdot m} \rightarrow \mathbb{F}$ 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 $\psi$ is a function $g : D^m \rightarrow \mathbb{F}$. Denote by $f^{(\psi, g)}$ the function

\[
f^{(\psi, g)} : D^m \rightarrow \mathbb{F}, \quad f^{(\psi, g)}(\overline{x}) \overset{\text{def}}{=} Q^{(\psi)}(\overline{x}, g(\pi_1, \ldots, \pi_t)(\overline{x})).
\]

(4.7)

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.

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 4.6 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 4.8 to an instance of aggregate-AGCSP. The proof appears in Section 4.6.

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

• **Input:**
  
  - A number $m \in \mathbb{N}$.
  - An alphabet $\Sigma$.
  - A constraint graph $G$ over $\Sigma$ whose underlying graph is a 4-regular subgraph of $H_{k,m}$.
  - A finite field $\mathbb{F}$.
  - Bases for all the codes in a multiplication code family $\vec{C} = (C_1, \ldots, C_{d_{\text{mult}}})$ of transitive evaluation codes $C_j = \{f : D \rightarrow \mathbb{F}\}$, where $C \overset{\text{def}}{=} C_1$ has message length at least $2 \cdot k$, and $d_{\text{mult}} \geq |\Sigma|$. 

58
For each $\alpha, \beta \in D$, a permutation $\pi$ of $D$ that (1) maps $\alpha$ to $\beta$, and (2) is an automorphism of $C_j$ for each $j \in [d]$.

- Output: An instance of $\text{aggregate-AGCSP}$

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

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

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

Remark. Note that the fact that the procedure in Theorem 4.9 runs in polynomial time implies in particular that the length of $\psi$ 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 section provide a proof of our main theorem based on the AG combinatorial Nullstellensatz. An alternative proof, based on the sum-check protocol, is presented in [BKK+13]. Before going into the proof, we first prove the following lemma. Intuitively, the lemma says: Suppose we have an assignment $g$ to $\psi$ 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 $\psi$, $C$, $m$, and $Q^{(\psi)}$ be as in the definition of $\text{aggregate-AGCSP}$, and let $\delta_C$ be the relative distance of $C$. Let $g : D^m \to F$ be an assignment to $\psi$ and let $f^{(\psi, g)}$ be as in the definition of $\text{aggregate-AGCSP}$. We say that a point $\bar{x} \in D^m$ is locally legal for $g$ if for every $i \in [m]$, it holds that $g|_{i, \bar{x}} \in C$ (i.e. $g$ restricted to the axis-parallel line that goes through $\bar{x}$ in direction $i$ is a codeword of $C$). We have the following result.

**Lemma 4.4.6.** Let $\hat{g} : D^m \to F$ be a codeword of $C^m$ that is $\tau$-close to $g$ for some $0 < \tau < 1$ (i.e., $g$ and $\hat{g}$ disagree on at most $\tau$ fraction of the points in $D^m$). Let $\bar{x}$ be a uniformly distributed point in $D^m$. Then

$$\Pr_{\bar{x} \in D^m} \left[ \bar{x} \text{ is locally legal for } g \text{ and } f^{(\psi, g)}(\bar{x}) \neq f^{(\psi, \hat{g})}(\bar{x}) \right] \leq m \cdot \tau / \delta_C. \quad (4.8)$$

*Proof.* We begin by fixing some point $\bar{x}$ that is both locally legal and satisfies $f^{(\psi, g)}(\bar{x}) \neq f^{(\psi, \hat{g})}(\bar{x})$. Recall that $f^{(\psi, g)}$ is computed by evaluating $g$ on all the axis-parallel lines that pass through $\bar{x}$, and the same goes for $f^{(\psi, \hat{g})}$ and $\hat{g}$. Thus, the assumption that $f^{(\psi, g)}(\bar{x}) \neq f^{(\psi, \hat{g})}(\bar{x})$ implies that there exists some $i \in [m]$ such that $g|_{i, \bar{x}} \neq \hat{g}|_{i, \bar{x}}$. 


Moreover, since we assume that $g$ is locally legal, we get that $g|_{i,x}$ is a legal codeword. Note that also $\hat{g}|_{i,x}$ is a legal codeword, due to the assumption that $\hat{g}$ is a codeword of $C^m$ and to Fact 4.3.3. We conclude that if $\pi$ is both locally legal and satisfies $f(\psi,g)(\pi) \neq f(\psi,\hat{g})(\pi)$, 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 4.8, 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 $\tau/\delta_C$, 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 $\tau$-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. 

4.4.3 A proof of Main Theorem 4.3 using AG combinatorial Nullstellensatz

Applying the pair of reductions described in the previous sections (cf. (4.1)) converts a circuit $\phi$ of size $n$ to an instance $\psi$ of aggregate-AGCSP whose assignments are of length $O(n)$. Using the notation of Theorem 4.9 and Definition 4.4.5, we see that $\phi$ is satisfiable if and only if there exists $g \in C^m \otimes m$ for which $f(\psi,g)$ defined in (4.7) vanishes on $H^m$. The AG combinatorial Nullstellensatz (Theorem 4.7) 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 4.4. Details follow.

Proof. of Theorem 4.3 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 $\phi$ 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 (4.1), i.e., the reduction of Theorem 4.8 followed by the reduction of Theorem 4.9. The first reduction maps $\phi$ to an instance $G$ of Hypercube-CSP over a subgraph of $H_{k,m}$ where $k \leq 4((4 \cdot n)^{1/m} + 1)$.

For the second reduction let $d = |\Sigma| = 4$ where $\Sigma$ is the alphabet stated in Theorem 4.8. We give one possible way of fixing parameters. Set $q = 2^{16}$, and note that
cq = 2^{255}. Set \(d_{\text{mult}} = 6d = 24\). Set \(\delta = \frac{1}{100}\) and \(\rho = \frac{1}{100cq}\), and note that Equation (4.4) is satisfied.

Thus we may take a transitive AG code \(C = \{f : D \rightarrow \mathbb{F}_q\}\) and multiplication code family \(\vec{C} = (C_1, \ldots, C_{d_{\text{mult}}} )\) as in Theorem 4.5, such that \(C\) has message length \(200k\), blocklength \(|D| \in [200k, \frac{200k}{\rho}]\), and each \(C_j\) has relative distance \(\delta\). We will be using Theorem 4.7 on this code. Note that every set \(H \subseteq D\) of size \(\leq 2k\) satisfies Equation (4.5), because

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

Now \(V\) applies Theorem 4.9 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 \(C_j, j \in [d_{\text{mult}}]\).
- For every \(\alpha, \beta \in D\), a permutation \(\pi\) of \(D\) that (1) is an automorphism of each \(C_j, j \in [d_{\text{mult}}]\), and (2) maps \(\alpha\) to \(\beta\).
- The pair of functions \(\xi = \xi_H, \xi' = \xi'_H : D \rightarrow \mathbb{F}_q\) defined in Theorem 4.7 where \(H \subset D, |H| \leq 2k\) is part of the aggregate-AGCSP instance \(\psi\) and defined in Theorem 4.9. (Note that our earlier discussion showed that Equation (4.5) is satisfied and thus Theorem 4.7 does apply here.)

Denote by \(\psi\) 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 \(\psi\), and \(f'_1, \ldots, f'_m\) should “prove” that \(f^{(\psi, g)}\) (cf. (4.7)) vanishes on \(H^m\) as per the AG combinatorial Nullstellensatz Theorem 4.7. The verifier performs the following checks while recycling randomness.

1. **Tensor test**: Invoke the local tester of Theorem 4.4 to test that \(g \in C^\otimes m\) and \(f'_\ell \in (C_d^d)^\otimes m\) for each \(\ell \in [m]\), using the same randomness for all the invocations.

2. **Zero test**: Choose a uniformly distributed point \(\pi = (x_1, \ldots, x_m) \in D^m\) and check that

\[
(i)\ f^{(\psi, g)}(\pi) \cdot \prod_{\ell=1}^{m} \xi'_\ell(x_\ell) = \sum_{j=1}^{m} f'_j(\pi) \cdot \xi(x_j),\text{ where } f^{(\psi, g)}(\pi) \text{ is computed by making } 1 + m \cdot t \text{ queries to } g.\text{ (ii) } \pi \text{ is a locally legal for } g. \text{ That is, for every direction } t \in [m], \text{ the axis-parallel line } g|_{t,\pi} \text{ is a codeword of } C.
\]

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 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

$$(m + 2) \cdot \log_2 q \cdot |D|^m < \left(\frac{2}{\varepsilon} + 3\right) \cdot 16 \cdot 2^{540/\varepsilon + 270} \cdot n \leq c_\varepsilon \cdot n \quad (4.9)$$

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

Completeness. Suppose $\varphi$ is satisfiable. Then by the completeness of Theorem 4.8 and Theorem 4.9 we know that $\psi$ is satisfiable. This means there exists $g \in C^{\otimes m}$ such that the function $f^{(\psi, g)}$ defined in (4.7) belongs to $(C_d)^{\otimes m}$ and vanishes on $H^m$. Hence, by the AG-Nullstellensatz Theorem 4.7 there exist $f'_1, \ldots, f'_m \in (C_{d4})^{\otimes m}$ such that (4.12) 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 $\varphi$ is not satisfiable. Then by the soundness of Theorem 4.8 and Theorem 4.9 we conclude $\psi$ is not satisfiable, i.e., there does not exist $g \in C^{\otimes m}$ such that $f^{(\psi, g)}$, as defined in (4.7), vanishes on $H^m$. Suppose the verifier is given the proof $g, f'_1, \ldots, f'_m$. We show that the verifier rejects with probability $\Omega_m(1)$.

Let $\delta$ be the minimum of the relative distances of $C_d$ and $C_{d4}$ and note that $\delta$ does not depend on $n$. If $g$ is $(\delta^{m+1}/2m)$-far from $(C_d)^{\otimes m}$ or any of $f'_1, \ldots, f'_m$ is $(\delta^{m+1}/2m)$-far from $(C_{d4})^{\otimes m}$, the tensor test rejects with probability at least $\gamma_m \cdot \delta^{m+1}/2m = \Omega_m(1)$ (where $\gamma_m$ is as defined in Theorem 4.4). Thus, we may focus on the case in which $g$ is $(\delta^{m+1}/2m)$-close to $(C_d)^{\otimes m}$ and all of $f'_1, \ldots, f'_m$ are $(\delta^{m+1}/2m)$-close to $(C_{d4})^{\otimes m}$. In this case, $g, f'_1, \ldots, f'_m$ are close to unique codewords $\hat{g}, \hat{f}'_1, \ldots, \hat{f}'_m$ of $(C_d)^{\otimes m}$ and $(C_{d4})^{\otimes m}$ respectively.

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

Let $\hat{f}' = \sum_{i=1}^m \hat{f}'_i(\pi) \cdot \xi(x_i)$ and $\hat{f}''(\pi) = f^{(\psi, \hat{g})}(\pi) \cdot \prod_{i=1}^m \xi(x_i)$, noticing $\hat{f}', \hat{f}'' \in (C_{d4})^{\otimes m}$. Since $f^{(\psi, \hat{g})}$ does not vanish on $H^m$ and by the AG-Nullstellensatz Theorem 4.7 we know that $\hat{f}' \neq \hat{f}''$. So by the distance property of $(C_{d4})^{\otimes m} = (C_{dmax})^{\otimes m}$, we conclude $\hat{f}'$ and $\hat{f}''$ differ on $\pi$ with probability at least $\delta_m$. Now, by Lemma 4.4.6, we get that with probability at least

$$1 - m \cdot \delta^{m+1}/2m \cdot \delta \geq 1 - \delta^m/2$$

62
one of the following cases occur:

1. $\varpi$ is not locally legal for $g$, so the zero test rejects.

2. $f^{(\psi, g)}(\varpi) = f^{(\hat{\psi}, g)}(\varpi)$, so

$$f^{(\hat{\psi}, g)}(\varpi) \cdot \prod_{r=1}^{m} \xi'(x_{r}) = \hat{f}''(\varpi) \neq \sum_{i=1}^{m} \hat{f}'_{i}(\varpi) \cdot \xi(x_{i}).$$

Since we assumed that all of $f_{1}', \ldots, f_{m}'$ agree with $\hat{f}_{1}', \ldots, \hat{f}_{m}'$ on $\varpi$, we get that

$$f^{(\hat{\psi}, g)}(\varpi) \cdot \prod_{r=1}^{m} \xi'(x_{r}) \neq \sum_{i=1}^{m} f_{i}'(\varpi) \cdot \xi(x_{i}),$$

and therefore the zero test rejects.

We conclude that the test rejects if $\hat{f}'$ and $\hat{f}''$ differ on $\varpi$, all of $f_{1}', \ldots, f_{m}'$ agree with $\hat{f}_{1}', \ldots, \hat{f}_{m}'$ on $\varpi$, and either $\varpi$ is not locally legal for $g$ or $f^{(\hat{\psi}, g)}(\varpi) = f^{(\hat{\psi}, g)}(\varpi)$. 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 $O(1/\delta^{m})$ 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 4.4 requires $\log_{2}(|D|^{m}) + O(\log m)$ random bits, and choosing $\varpi$ 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 $\log_{2} n + O(\log m + \frac{1}{\delta^{m}}) = \log_{2} n + 2^{O(\frac{1}{\delta^{m}})}$.

**Query complexity.** Each invocation of the local tester of Theorem 4.4 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 $f^{(\psi, g)}(\varpi)$, another $m$ queries to $f_{1}', \ldots, f_{m}'$, and another $m \cdot |D|$ queries to verify that $\varpi$ 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 $O(\text{poly}(\frac{1}{\delta^{m}}) \cdot n^{\epsilon})$.

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

**Verifier complexity.** The verifier clearly runs in time $O_{\epsilon}(1) \cdot \text{poly}(n)$.
Decision complexity. The decision predicate of the verifier consists of:

1. poly(|D|) arithmetic operations on the field elements it queries.

2. Verifying membership in the code C, which can also be done using poly(|D|) arithmetic operations since C is linear.

3. Invoking the local tester of Theorem 4.4, which uses 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.

4.5 Hypercube CSPs — Proof of Theorem 4.8

In this section we prove the first step in the reduction (4.1) — Theorem 4.8. 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 4.5.1, we reduce the circuit-SAT problem to graph-CSP (see Definition 4.4.2). This reduction yields a constraint graph whose underlying graph resembles the topology of the original circuit-SAT instance. Then, in Lemma 4.5.2, 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 4.5.1** (Reduction from circuit-SAT to graph-CSP). There exists a polynomial time procedure that maps every circuit \( \varphi \) of size n to a constraint graph \( G_\varphi \) of size n over alphabet \( \Sigma = \{0, 1\}^2 \) such that \( G_\varphi \) is satisfiable if and only if \( \varphi \) is satisfiable.

**Proof.** Fix a circuit \( \varphi \). We describe the construction of the corresponding constraint graph \( G_\varphi \). For each gate \( g \) of \( \varphi \), the graph \( G_\varphi \) has a corresponding vertex \( v_g \), and for each wire \((g_1, g_2)\) of \( \varphi \) the graph \( G_\varphi \) has a corresponding edge \((v_{g_1}, v_{g_2})\). The alphabet of \( G_\varphi \) is \( \Sigma = \{0, 1\}^2 \). Intuitively, an assignment \( \sigma : V \to \Sigma \) should label a vertex \( v_g \) with a symbol \( b_1b_2 \in \{0, 1\}^2 \) 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 \( G_\varphi \), 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 \((a_1a_2, b_1b_2) \in (\{0, 1\}^2)^2 \) 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 \( G_\varphi \) is satisfiable if and only if \( \varphi \) is satisfiable. \( \square \)

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 4.5.2.** There exists a polynomial time procedure when given as input
- A constraint graph \( G = (V, E) \) of size \( n \) and alphabet \( \Sigma \); and
- An integer \( m \in \mathbb{N} \),

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

By combining Proposition 4.5.1 and Lemma 4.5.2, Theorem 4.8 follows immediately:

**Theorem 4.10** (4.8, restated). There exists a polynomial time procedure that maps every circuit \( \varphi \) of size \( n \) and integer \( m \in \mathbb{N} \) to a constraint graph \( G_{\varphi,m} \) over an alphabet \( \Sigma \) of size 4 that is satisfiable if and only if \( \varphi \) 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 4.5.2 is based on the following proposition, whose proof is deferred to end of this section.

**Proposition 4.5.3.** There exists a polynomial time procedure that takes as input a permutation \( \pi \) on \([k]^m\) and outputs a collection \( P \) 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 2\( k \)-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 \( P \) and the last vertex of another path in \( P \), but other than that we require the paths to be vertex disjoint.

**Remark.** Proposition 4.5.3 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 4.5.2.

**Proof.** of Lemma 4.5.2 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. 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 \( M_1, \ldots, M_4 \) in polynomial time (see, e.g., [Cam98, Proposition 18.1.2]).

Let \( k = \lceil |V|^{1/m} \rceil \leq (4n)^{1/m} + 1 \), 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 \( H \equiv H_{4k,m} \). More specifically, observe that the graph \( H \) contains four copies of the hypercube \( H_{2k,m} \): 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 \( H_1, \ldots, H_4 \) respectively. The procedure will embed the edges of the matchings \( M_1, \ldots, M_4 \) on vertex-disjoint paths in \( H_1, \ldots, H_4 \) respectively by applying Proposition 4.5.3 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 \( H \). 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 \( \pi \) on \([k]^m\) (recall that we identify the vertices of \( G \) with \([k]^m\)). The procedure then applies Proposition 4.5.3 to the permutation \( \pi \) and the graph \( H_j \), thus obtaining a collection \( \mathcal{P} \) of vertex-disjoint paths in \( H_j \) that connect every pair of vertices in \([k]^m\) that are matched by \( \pi \). 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

\[
2m(4(4n)^{1/m})^m \leq 2m4^{m+2}n.
\]

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 4.5.3.

**Proof.** of Proposition 4.5.3 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 \([k]^{m-1}\), and in which two vertices \((i_1, \ldots, i_{m-1}), (j_1, \ldots, j_{m-1}) \in [k]^{m-1}\) are connected by an edge if and only if \( i_1 = j_1 \) and \( \cdots \) and \( i_{m-1} = j_{m-1} \). This is a 3-regular constraint graph of size \( 4k^{m-1} \), and it is easy to see that the procedure runs in polynomial time (see, e.g., [Cam98, Proposition 18.1.2]).
by an edge if and only if there exist $i_m, j_m \in [k]$ such that $\pi(i_1, \ldots, i_m) = (j_1, \ldots, j_m)$. 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 $\pi_1, \ldots, \pi_k$ in polynomial time.

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 $P_1, \ldots, P_k$ in $H_{2k,m−1}$. Finally, in order to construct the output collection $P$, the procedure constructs a path for each vertex $(i_1, \ldots, i_m) \in [k]^m$ 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 $(i_1, \ldots, i_m)$ and $(j_1, \ldots, j_m)$, and suppose that this edge belongs to the matching $\pi_t$. Let $p = (v_1, \ldots, v_t)$ be the path in $P_t$ that connects $(i_1, \ldots, i_m)$ to $(j_1, \ldots, j_m)$ in $H_{2k,m−1}$ where $v_1 = (i_1, \ldots, i_m)$ and $v_l = (j_1, \ldots, j_m)$. Now, we choose the path $p'$ that connects $(i_1, \ldots, i_m)$ to $(j_1, \ldots, j_m)$ in $H_{2k,m}$ to be $(v'_0, \ldots, v'_{t+1})$ where $v'_0 = (i_1, \ldots, i_m)$, $v'_{t+1} = (j_1, \ldots, j_m')$, and for each $i \in [t]$, 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 $i \in [t]$, 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 $P$ 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 $H_{2k−1,m}$ from which $p'_1$ and $p'_2$ were obtained. If $p_1$ and $p_2$ belong to two distinct collections $P_{t_1}$ and $P_{t_2}$, then it is clear that $p'_1$ and $p'_2$ must be vertex-disjoint, since the last coordinate of all the vertices in $p'_i$ (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 $P_t$, 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 $c_1$. 

\[\square\]
4.6 From Hypercube-CSP to aggregate-AGCSP

In this section we prove the second step in Reduction (4.1) — 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 4.6.2, we apply an aggregation step to reach the language aggregate-AGCSP that is the end-point of the reduction stated in Theorem 4.9.

4.6.1 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.

Let $C = \{ f : D \rightarrow \mathbb{F} \}$, let $\pi$ be an automorphism of $C$, and let $g : D^m \rightarrow \mathbb{F}$ be a codeword of the tensor code $C^\otimes m$. Then, for each $i \in [m]$, we define the function $g^{\pi,i} : D^m \rightarrow \mathbb{F}$ 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 $g^{(\pi_1, \ldots, \pi_t)} : D^m \rightarrow \mathbb{F}^{1+t \cdot m}$ to be the function obtained by aggregating the $1 + t \cdot m$ functions $g, g^{\pi_1,1}, \ldots, g^{\pi_t,m}$. Formally,

$$g^{(\pi_1, \ldots, \pi_t)}(\overline{x}) = (g(\overline{x}), g^{\pi_1,1}(\overline{x}), \ldots, g^{\pi_t,m}(\overline{x})).$$

**Definition 4.6.1 (AGCSP).** An instance of the AGCSP problem is a tuple

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

where

- $m, d, t$ are integers
- $C$ is a systematic linear code that encodes messages $h : H \rightarrow \mathbb{F}$ to codewords $f : D \rightarrow \mathbb{F}$.
- $\pi_1, \ldots, \pi_t$ are automorphisms of $C$.
- For each $\overline{v} \in H^m$, the function $Q_\pi$ is a polynomial of degree $< d$ over $1 + t \cdot m$ variables. The polynomials $Q_\pi$ will serve as the constraints.

An assignment is a function $g : D^m \rightarrow \mathbb{F}$. 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

$$Q_\pi \left( g^{(\pi_1, \ldots, \pi_t)}(\overline{v}) \right) = 0.$$
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 4.4.4 (cf. Section 4.5) to our algebraic CSP.

**Theorem 4.11.** There exists a polynomial time procedure with the following input-output behavior:

- **Input:**
  - A number \( m \in \mathbb{N} \).
  - An alphabet \( \Sigma \).
  - A constraint graph \( G \) over \( \Sigma \) whose underlying graph is a 4-regular subgraph of \( H_{k,m} \).
  - 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 \( \pi \) of \( D \) that (1) maps \( \alpha \) to \( \beta \), and (2) is an automorphism of \( C \).

- **Output:** An instance of AGCSP \( \psi = (m, d, t, \mathbb{F}, C, H, \pi_1, \ldots, \pi_t, \{ Q_v | v \in H^m \}) \) 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 \( C \) such that for every \( \alpha, \beta \in H \) there exists some automorphism \( \pi_j \) such that \( \pi_j(\alpha) = \beta \) - those automorphisms are just a subset of the automorphisms given in the input.

We define the instance \( \psi \) of AGCSP that is the output of the procedure. We set

\[
\psi = (m, d, t, \mathbb{F}, C, H, \pi_1, \ldots, \pi_t, \{ Q_v | v \in H^m \})
\]

where \( d \overset{\text{def}}{=} |\Sigma| \), \( t = O(k^2) \), and where the polynomials \( \{ Q_v | v \in H^m \} \) 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 \( H_{k,m} \)). Let us identify \( \Sigma \) with some subset of \( \mathbb{F} \). We will have two kinds of polynomials \( Q_v \):  

1. Those \( Q_v \)'s for which \( v \in (H - H_0) \times H_0^{m-1} \). Those polynomials will check that the assignment \( g \) assigns to \( H_0^m \) values that are in \( \Sigma \).
2. Those $Q_\tau$'s for which $\tau \in H_0^m$. 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 $Q_\tau$ will be equal to the zero polynomial.

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

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

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 $Q_\tau$ by $z$ and $\{y_{i,j}\}_{i=1,j=1}^{l,m}$, and define $Q_\tau$ to be the unique polynomial that satisfies the following two requirements:

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

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

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

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

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

The polynomial $Q_\tau$ verifies that the assignment $g$ assigns to $\overline{v}$ a value in $\Sigma$, and is constructed as follows. Let $\pi_{l_0}$ be an automorphism of $C$ that maps $v_1$ to $\eta(v_1)$. Again, we denote variables of the polynomial $Q_\tau$ by $z$ and $\{y_{i,j}\}_{i=1,j=1}^{l,m}$. Now, we define $Q_\tau$ to be the unique polynomial that involves only the variable $y_{l_0,1}$, and whose roots are exactly the values in $\Sigma$. Note that such a polynomial $Q_\tau$ 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 $H_0^m$ to $\mathbb{F}$. We claim that $g$ is satisfying assignment of $\psi$. To this end, let us consider the two kinds of polynomials $Q_\tau$ separately.
Let $\mathbf{v} \in H_0^m$, and let $u_1, u_2, u_3, l_1, l_2, l_3, j_1, j_2, j_3$ be defined as before. We need to show that $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v})) = 0$. Observe that in the expression $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v}))$, the variables $z$, $y_1, j_1$, $y_{l_2}, j_2$, $y_{l_3}, j_3$ are assigned $g(\mathbf{v})$, $g_{\pi_1,j_1}(\mathbf{v})$, $g_{\pi_{l_2},j_2}(\mathbf{v})$, $g_{\pi_{l_3},j_3}(\mathbf{v})$ respectively, that $g(\mathbf{v}) = \sigma(\mathbf{v})$, and that for each $s \in [3]$ it holds that $g_{\pi_{l_s},j_s}(\mathbf{v}) = g(\pi_s)$. Now, since $\sigma$ satisfies the constraints of $\mathbf{v}$, we get that $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v})) = 0$ by the definition of $Q_{\mathbf{v}}$.

• Let $\mathbf{v} \in (H - H_0) \times H_0^{m-1}$, and let $\mathbf{v} \in H_0^m$ be defined as before. We need to show that $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v})) = 0$. Observe that in the expression $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v}))$, the variable $y_{l_0,1}$ is assigned $g_{\pi_{l_0,1}}(\mathbf{v}) = g(\mathbf{v}) = \sigma(\mathbf{v})$. Since $\sigma(\mathbf{v}) \in \Sigma$, we get that $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v})) = 0$ by the definition of $Q_{\mathbf{v}}$.

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

Next, to show that $\sigma$ is indeed a satisfying assignment for $\psi$, we show that for each $\mathbf{v} \in H_0^m$, the assignment $\sigma$ satisfies the constraints of $\mathbf{v}$ in $G$. Fix $\mathbf{v}$ and let $\pi_1, \pi_2, \pi_3, l_1, l_2, l_3, j_1, j_2, j_3$ be defined as before. Then, in the expression $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v}))$, the variables $z$, $y_1, j_1$, $y_{l_2}, j_2$, $y_{l_3}, j_3$ are assigned $\sigma(\mathbf{v})$, $\sigma(\pi_1)$, $\sigma(\pi_2)$, and $\sigma(\pi_3)$. Since $Q_{\mathbf{v}}(g(\pi_1, \ldots, \pi_t)(\mathbf{v})) = 0$, we get by the definition of $Q_{\mathbf{v}}$ that $\sigma$ satisfies the constraints of $\mathbf{v}$ (as $Q_{\mathbf{v}}$ is of the second type).

### 4.6.2 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 $\tilde{C}$. In this section we show how to reduce AGCSP to aggregate-AGCSP, thus proving Theorem 4.9.

**Definition 4.6.2** (4.4.5, aggregate-AGCSP, restated). An instance of the aggregate-AGCSP problem is a tuple

$$\psi = (m, d, t, \mathbb{F}, \tilde{C}, H, \pi_1, \ldots, \pi_t, Q(\psi))$$

where

• $m, d, t$ are integers

• $\tilde{C} = (C_1, \ldots, C_d)$ 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 $Q^{(\psi)}(\overline{\pi}, g^{(\pi_1, \ldots, \pi_t)}(\overline{\pi}))$ is a codeword of $(C_d)_\otimes m$.

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

$$f^{(\psi, g)} : D^m \to \mathbb{F}, \quad f^{(\psi, g)}(\overline{\pi}) \overset{\text{def}}{=} Q^{(\psi)}(\overline{\pi}, g^{(\pi_1, \ldots, \pi_t)}(\overline{\pi})) \quad (4.10)$$

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., $f^{(\psi, g)}(\overline{\pi}) = 0$ for all $\overline{\pi} \in H^m$.

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

**Theorem 4.12** (4.9, restated). There exists a polynomial-time procedure with the following input-output behavior:

• **Input:**
  
  - A number $m \in \mathbb{N}$.
  
  - An alphabet $\Sigma$.
  
  - A constraint graph $G$ over $\Sigma$ whose underlying graph is a 4-regular subgraph of $H_{k,m}$.
  
  - A finite field $\mathbb{F}$.
  
  - Bases for all the codes in a multiplication code family $\widetilde{C} = (C_1, \ldots, C_{d_{\text{mult}}})$ 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 $d_{\text{mult}} \geq |\Sigma|$.
  
  - For each $\alpha, \beta \in D$, a permutation $\pi$ of $D$ that (1) maps $\alpha$ to $\beta$, and (2) is an automorphism of $C_j$ for each $j \in [d]$.

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

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

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 4.3.3).
Proposition 4.6.3. If \((C_1, \ldots, C_d)\) 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 4.3.7) is contained in \((C_d)^\otimes m\).

Proposition 4.6.4. For every codeword \(g \in C^\otimes m\), an automorphism \(\pi\) 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

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

Proof. of Theorem 4.9 The procedure of Theorem 4.9 works as follows. First, it invokes the procedure of Theorem 4.11, giving it the code \(C = C_1\) as input. This results in an instance

\[
\psi_0 = (m, d = |\Sigma|, t, \mathbb{F}, C, H, \pi_1, \ldots, \pi_t, \{Q_{\psi} | \psi \in H^m\})
\]

of AGCSP. Then, the procedure constructs its output \(\psi\) from \(\psi_0\) by replacing the polynomials \(Q_{\psi}\) with a corresponding function \(Q^{(\psi)}\) (and also includes the entire multiplication code family \(\widetilde{C}\) in \(\psi\)). Roughly, we construct \(Q^{(\psi)}\) by summing all the terms of the form \(1_{\psi} \cdot Q_{\psi}\), where \(1_{\psi}\) is the indicator function of \(\psi\). Details follow.

We will need some notation. For every \(\overline{\psi} \in H^m\), let \(1_{\overline{\psi}} : H^m \to \mathbb{F}\) be the indicator function of \(\overline{\psi}\), i.e., the function that takes the value 1 on \(\overline{\psi}\) 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 \(\overline{1}_{\psi} : D^m \to \mathbb{F}\) be any codeword of \(C^\otimes m\) such that \(\overline{1}_{\psi}|_{H^m} = 1_{\psi}\) (i.e., its restriction to \(H^m\) equals \(1_{\psi}\)), and we let \(\overline{1}_{H^m} : D^m \to \mathbb{F}\) be any codeword of \(C^\otimes m\) such that \(\overline{1}_{H^m}|_{H^m} = 1_{H^m}\) (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 \(Q_{\psi}\) such that they are homogeneous, i.e., such that all their monomials are of degree exactly \(d - 1\). Fix \(\overline{\psi} \in H^m\), and consider the polynomial \(Q_{\overline{\psi}} : \mathbb{F}^{1+t \cdot m} \to \mathbb{F}\). Let us denote the variables of this polynomial by \(z\) and \(\{y_{l,j}\}_{l,j=1}^{t,m}\) as before. To make \(Q_{\overline{\psi}}\) homogeneous, we add a new variable \(s\), and multiply each monomial of \(Q_{\overline{\psi}}\) by the appropriate power of \(s\) to increase its degree to \(d - 1\). Let us denote by \(Q_{\overline{\psi}}'(z, (y_{l,j})_{l,j=1}^{t,m}, s)\) the resulting polynomial.

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

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

Observe that for every \(\overline{\psi} \in H^m\), it holds that \(P_{\overline{\psi}}(\overline{\psi}, z, (y_{l,j})_{l,j=1}^{t,m}) = Q_{\overline{\psi}}(z, (y_{l,j})_{l,j=1}^{t,m})\), since \(\overline{1}_{H^m}(\overline{\psi}) = 1\) for every such \(\overline{\psi}\).
Finally, we define the function \( Q(\psi) : D^m \times \mathbb{F}^{1+t \cdot m} \to \mathbb{F} \) by
\[
Q(\psi)(x, z, (y_{l,j})_{l=1,j=1}^{t,m}) \overset{\text{def}}{=} \sum_{v \in H^m} \tilde{1}_v(x) \cdot P_\pi(x, z, (y_{l,j})_{l=1,j=1}^{t,m}).
\]

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 \( f(\psi, g)(\overline{v}) = Q_\pi(g^{(\pi_1, \ldots, \pi_t)}(\overline{v})) \). Note that this in particular implies that \( g \) satisfies \( \psi \) if and only if it satisfies \( \psi_0 \). It holds that
\[
f(\psi, g)(\overline{v}) = \sum_{\pi \in H^m} \tilde{1}_\pi(\overline{v}) \cdot P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v}))
\]
(Since \( \tilde{1}_\pi(\overline{v}) = 0 \) for all \( \overline{u} \in H^m - \{\overline{v}\} \))
\[
= P_\pi(\overline{v}, g^{(\pi_1, \ldots, \pi_t)}(\overline{v})).
\]
(Since \( P_\pi = Q_\pi \) on all \( \overline{x} \in H^m \))
\[
= Q_\pi(g^{(\pi_1, \ldots, \pi_t)}(\overline{v})),
\]
as required.

We turn to show that \( Q(\psi) \) satisfy the requirements of Definition 4.4.5. 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 4.6.3, it suffices to prove that \( f(\psi, g) \) is a codeword of \( (C^{\otimes m})^d \). Furthermore, it suffices to prove that each \( P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v})) \) is a codeword of \( (C^{\otimes m})^{d-1} \), since this would imply that each
\[
\tilde{1}_\pi(\overline{v}) \cdot P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v}))
\]
is a codeword of \( (C^{\otimes m})^d \) (being the coordinate-wise multiplication of codewords of \( C^{\otimes m} \) and \( (C^{\otimes m})^{d-1} \), respectively). This, in turn, would imply that \( f(\psi, g) \) is a codeword of \( (C_d)^{\otimes m} \) (being the sum of codewords of \( (C^{\otimes m})^d \)).

We thus focus on proving that each \( P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v})) \) is a codeword of \( (C^{\otimes m})^{d-1} \). It suffices to observe that \( P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v})) \) is a sum of coordinate-wise multiplications of \((d-1)\) codewords of \( C^{\otimes m} \). To see this, observe that \( P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v})) \) is obtained from \( Q_\pi(\overline{v}) \) 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 4.6.4). Now, each monomial of \( Q_\pi(\overline{v}) \) contributes to \( P_\pi \) a coordinate-wise multiplication of exactly \( d-1 \) of those functions, so each such monomial contributes a codeword of \( (C^{\otimes m})^{d-1} \). Finally, \( P_\pi \) is the sum of all those monomials, and therefore \( P_\pi(x, g^{(\pi_1, \ldots, \pi_t)}(\overline{v})) \) is a codeword of \( (C^{\otimes m})^{d-1} \), as required.

It remains to show that \( Q(\psi) \) can be computed in polynomial time given the polynomials \( Q_\psi \), and this will imply that \( Q(\psi) \) can be represented by a small circuit. This follows from the fact that the codewords \( \tilde{1}_\pi \) 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).
4.7 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 4.7) is used to solve the “zero-testing” problem over tensors of AG codes.

4.7.1 Formal statements of Nullstellensatz

We start with an AG code defined by a projective nonsingular curve $C$ 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 $C = C_L(C, D, G)$ consists of evaluations of functions in the Riemann-Roch space $\mathcal{L}(G)$ at the set of points $D$. A model case to keep in mind is the evaluation code of polynomials over $\mathbb{F}_q$ with degree $\leq k$ (Reed-Solomon codes), where $C$ is the projective line $\mathbb{P}^1(\mathbb{F}_q)$ (which is of genus 0), $D$ is a set of $\mathbb{F}_q$ rational points on $C$ not including $\infty$ (the point at $\infty$), and $G$ equals $k \cdot (\infty)$.

The $m$-dimensional tensor code of $C$ is naturally viewed as a set of “multivariate” algebraic functions defined on the $m$-dimensional variety $C^m$, 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 $C^m$. Thus for a function $f \in \mathcal{L}(G)$, the function $f(X_i)$ represents a rational function on $C^m$ which “ignores” all coordinates except the $i$th. For divisors $G_1, \ldots, G_m$ on $C$, define the space $\mathcal{L}(G_1, \ldots, G_m)$ of multivariate algebraic functions on $C^m$ as follows:

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

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 4.7.1** (Tensored AG codes). Let $C_L(i) = C_L(C_i, D_i, G_i), i = 1, \ldots, m$ be $m$ AG codes over $\mathbb{F}_q$. The codewords of the tensor product code $\bigotimes_{i=1}^{m} C_L(i)$ are the evaluations of functions in $\mathcal{L}(G_1, \ldots, G_m)$ over $\prod_{i} D_i$:

$$\bigotimes_{i=1}^{m} C_L(i) = \left\{ \langle f(x_1, \ldots, x_m) \rangle \mid f \in \mathcal{L}(G_1, \ldots, G_m) \right\}.$$

**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

$$\langle f(x_1, \ldots, x_m) \rangle_{x_i \in D_i \text{ for } i = 1, \ldots, m-1} = \bigotimes_{i=1}^{m-1} C_L(i).$$
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 \( m \)th axis belongs to \( \mathcal{L}_C(m) \) and this completes the proof. \( \square \)

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 \( H \subseteq D \). In the case of polynomial codes (coming from curves of genus 0), one such characterization can be achieved via Alon’s Combinatorial Nullstellensatz (Theorem 4.6). We seek a generalization of the Combinatorial Nullstellensatz to arbitrary algebraic curves. However, even formulating a generalization requires some care, because of some fundamentally different features of curves with positive genus. In the case of a genus 0 curve \( C \), for every subset \( H \subseteq C \) there exists a function \( \xi_H(X) \) whose degree equals \( |H| \) and which vanishes precisely on the points of \( H \) with multiplicity 1. This property was crucial to even state the standard Combinatorial Nullstellensatz. On a general curve the existence of such a \( \xi_H \) holds only for rather special sets \( H \) which we call splitting sets. For splitting sets we obtain the following analog of Theorem 4.6, which will be proved in section 4.7.4.

**Theorem 4.13** (AG Combinatorial Nullstellensatz for splitting sets). Let \( C \) be an algebraic curve over a field \( K \), \( G > 0 \) be a divisor on it, and \( H \subseteq C \) be a set of rational points, disjoint from \( \text{Support}(G) \), satisfying \( \deg G \geq |H| + 2g(C) - 1 \). 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) \( \deg (\xi(X))_0 = |H| \). (Thus \( H = \text{Support}((\xi)_0) \), and \( \xi \) has only zeroes 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 \( f_i'(X_1, \ldots, X_m) \in \mathcal{L}^m(G) \) such that

\[
f(X_1, \ldots, X_m) = \sum_{i=0}^{m} f_i'(X_1, \ldots, X_m)\xi(X_i). 
\]

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 \( C \) but requires two auxiliary functions \( \xi_H \) and \( \xi'_H \).

**Theorem 4.14** (AG Combinatorial Nullstellensatz for arbitrary sets). Let \( C \) be an algebraic curve over a perfect field\(^6\) \( K \), \( G > 0 \) be a divisor on it, and \( H \subseteq C \) be a set of rational points, disjoint from \( \text{Support}(G) \), satisfying \( \deg G \geq |H| + 2g(C) - 1 \). 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 \( f(X_1, \ldots, X_m) \in \mathcal{L}^m(G) \). Then \( f \) vanishes on \( H^m \) if and only if there exist

---

\(^6\)While the definition of perfect fields is not important for the statement, we note that all finite fields and all fields of characteristic 0 are perfect.
\[ f'_1, \ldots, f'_m \in L^m(AG) \text{ such that} \]
\[ f(X_1, \ldots, X_m) \cdot \prod_{i=1}^{m} \xi'(X_i) = \sum_{i=1}^{m} f'_i(X_1, \ldots, X_m) \cdot \xi(X_i). \quad (4.12) \]

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

This means that Theorem 4.13 is sufficient for our purposes. However, Theorem 4.14 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 4.18) to be of independent interest.

### 4.7.2 Instantiating parameters

We now show how to instantiate parameters into the construction of dense optimal transitive towers to get Theorem 4.5, and then how to use Theorem 4.13 to get Theorem 4.7. In order to do this, we first quote the main theorem from the appendix to [BKK+13].

**Theorem 4.15.** Assume that \( q = \ell^2 > 4 \) is a square. Then there exist asymptotically good dense families of transitive codes over \( \mathbb{F}_q \).

**Proof.** of Theorem 4.5 Let \( \epsilon > 0 \) be a constant (independent of \( k \)) such that:

\[ c_q \cdot d_{\text{mult}} \cdot \rho + \delta < 1 - \epsilon - \frac{d_{\text{mult}}}{\sqrt{q} - 1}. \]

Theorem 4.15 gives 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:

\[ n' = \frac{d_{\text{mult}} \cdot k}{1 - \epsilon - \delta - \frac{d_{\text{mult}}}{\sqrt{q} - 1}}, \]

and consider the AG curve given as above. From Theorem 4.15, we have the following properties:

1. There is a group \( \Gamma \) of automorphisms of \( \mathcal{C} \) which acts transitively on \( D \).
2. \( n \overset{\text{def}}{=} |D| \leq n' \cdot c_q < k \cdot \frac{1}{\rho} \).
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 \( \Gamma \), with:
5. For $j \in [d_{\text{mult}}]$, we define $C_j$ to be the AG code $C_L(C,D,jG^*)$. Then $\bar{C} = (C_1, \ldots, C_{d_{\text{mult}}})$ is a multiplication code family. Furthermore, each $C_j$ is invariant under the action of $\Gamma$.

6. For $j \in [d_{\text{mult}}]$, $C_j$ has distance at least

$$1 - \frac{\deg(jG^*)}{n} \geq \delta.$$ 

7. By the Riemann-Roch theorem,

$$\dim(C) \geq \deg(G^*) - \frac{n}{\sqrt{q} - 1} \geq \left(1 - \frac{\epsilon - \delta}{d_{\text{mult}}} - \frac{1}{\sqrt{q} - 1}\right) \cdot n \geq \left(1 - \frac{\epsilon - \delta}{d_{\text{mult}}} - \frac{d_{\text{mult}}}{\sqrt{q} - 1}\right) \cdot \frac{d_{\text{mult}} \cdot k}{1 - \epsilon - \delta - \frac{d_{\text{mult}}}{\sqrt{q} - 1}} \geq k.$$

This concludes the proof of Theorem 4.5.

Now we show that these codes also satisfy the conclusion of Theorem 4.7.

**Proof.** of Theorem 4.7 We keep the notation of the proof of Theorem 4.5. We take the code given by Theorem 4.5 with the parameter $\epsilon = \frac{1}{2}$. We simply have to check that the hypotheses of Theorem 4.14 are satisfied.

Put $G = dG^*$. We have $C = C_L(C,D,G^*)$, and $C_d = C_L(C,D,G)$. The hypothesis of Theorem 4.7 has a set $H \subseteq D$ with

$$|H| \leq \left(\frac{1}{6} \left(\frac{1}{2} - \delta\right) - \frac{2}{\sqrt{q} - 1}\right) \cdot n.$$

This implies that

$$|H| \leq \frac{d}{d_{\text{mult}}} \left(\frac{1}{2} - \delta\right) n - \frac{2n}{\sqrt{q} - 1} = \frac{d}{d_{\text{mult}}} \left(1 - \epsilon - \delta\right) n - \frac{2n}{\sqrt{q} - 1} \leq \deg(G) - 2g(C).$$

Thus we may apply Theorem 4.7. 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 4.14, $f$ vanishes on $H^m$ if and only if there exists $f'_1, \ldots, f'_m \in \mathcal{L}^m(G)$. This completes the proof of Theorem 4.7.
\(L^m(4G)\) (which we view as elements of \((C_4d)^\otimes m\)) such that:

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

This implies the desired result. \qed

### 4.7.3 Proof strategy

We demonstrate our proof strategy by using it to prove the original Combinatorial Nullstellensatz [Alo99] for the special case of bivariate polynomials \((m = 2)\).

**Theorem 4.16.** For \(H \subset \mathbb{F}_q\) let \(\xi_H(Z) = \prod_{\alpha \in H}(Z - \alpha)\) be the nonzero monic polynomial of degree \(|H|\) that vanishes on \(H\). Let \(f(X,Y)\) be a polynomial over \(\mathbb{F}_q\) of individual degree at most \(d\). Then \(f(X,Y)\) vanishes on \(H \times H\) if and only if there exist polynomials \(f'_1, f'_2 \in \mathbb{F}_q[X,Y]\) of individual degree at most \(d\) such that

\[
f(X,Y) = f'_1(X,Y) \cdot \xi_H(X) + f'_2(X,Y) \cdot \xi_H(Y). \quad (4.13)
\]

**Proof.** In the easy direction, if (4.13) holds then the right hands side is obviously zero on all of \(H \times H\) and therefore so is \(f\).

For the other direction we start by noticing that the analogous statement for univariate polynomials \((m = 1)\) holds trivially. Namely,

**Fact 4.7.2.** A degree \(d\) univariate polynomial \(g(X)\) vanishes on \(H\) if and only if there exists a polynomial \(g'(X)\) such that \(g(X) = g'(X) \cdot \xi_H(X)\); furthermore \(\deg(g') = \deg(g) - |H| \leq d\).

Using this observation about univariate polynomials we proceed to prove the theorem for bivariate ones.

For each \(y \in H\) define \(f_y(X) \overset{\text{def}}{=} f(X,y)\) to be the univariate polynomial obtained by fixing \(Y\) to \(y\). By Fact 4.7.2 we have

\[
f_y(X) = \xi_H(X) \cdot f'_y(X) \quad (4.14)
\]

Define \(\delta_y(Z)\) to be the degree \(|H|\) Lagrange polynomial satisfying

\[
\delta_y(z) = \begin{cases} 1 & z = y \\ 0 & z \in H \setminus \{y\} \end{cases}
\]

Now define

\[
h(X,Y) = f(X,Y) - \sum_{y \in H} \delta_y(Y)f_y(X) = f(X,Y) - \xi_H(X) \sum_{y \in H} \delta_y(Y)f'_y(X) \quad (4.15)
\]

The last equality uses (4.14).
Notice \( h \) has individual degree at most \( d \) because it is the difference of two polynomials of individual degree at most \( d \). For every \( y \in H \) the univariate polynomial \( h(X, y) \) is the zero polynomial, by construction. In other words, if we rewrite \( h(X, Y) = \sum_{i=0}^{d} h_i(Y)X^i \), then for every \( i \) and \( y \in H \) we have \( h_i(y) = 0 \). So by Fact 4.7.2 \( h_i(Y) = \xi_H(Y)h'_i(Y) \) and \( h(X, Y) = \xi_H(Y) \sum_{i=0}^{d} h'_i(Y)X^i \). Plugging this into (4.15) gives

\[
\xi_H(Y) \sum_{i=0}^{d} h'_i(Y)X^i = f(X, Y) - \xi_H(X) \sum_{y \in H} \delta_y(Y)f'_y(X) \tag{4.16}
\]

and to complete the proof let \( f'_1(X, Y) \) be the last sum above and \( f'_2(X, Y) \) be the first one.

\[\square\]

**Remark.** Note that the proof above can be used to get better bounds on the (total and individual) degree of \( f'_1, f'_2 \) and these better bounds are indeed stated in [Alo99].

### 4.7.4 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 4.7.5, but the proof in this case is simpler, so we do include it.

**Lemma 4.7.3 (Interpolation).** Let \( C \) be a curve over \( K \) and let \( G \) be a divisor on \( C \). For every set of rational points \( H \subseteq C \) disjoint from \( \text{Support}(G) \) and function \( a : H \rightarrow K \), if \( \deg(G) \geq |H| + 2g - 1 \), there exists \( f(X) \in \mathcal{L}(G) \) with \( f(x) = a(x) \) for each \( x \in H \).

**Proof.** Consider the linear mapping \( T : \mathcal{L}(G) \rightarrow K^{|H|} \) that sends each function in \( \mathcal{L}(G) \) to its evaluation on the points of \( H \). The kernel of \( T \) is \( \mathcal{L}(G - H) \) (we overload notation here to think of \( H \) as a divisor). By the Riemann-Roch theorem, the dimension of \( \mathcal{L}(G - H) \) is \( t(G - H) = \deg G - |H| - g + 1 \) (since \( \deg(G - H) \geq 2g - 1 \)), and the dimension of \( \mathcal{L}(G) \) is \( \deg G - g + 1 \). Thus the dimension of the image of \( T \) is \( |H| \) and the mapping is surjective. This implies the result. \[\square\]

We now prove the AG Combinatorial Nullstellensatz for splitting sets.

**Proof.** of Theorem 4.13 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 \( f \) 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 \( f'_1(X_1) = \frac{f(X_1)}{\xi(X_1)} \) (this is a rational function on \( C \)). Clearly (4.11) holds, and so we only have to prove that \( f'_1 \in \mathcal{L}(G) \). In other words, we have to show that

\[
(()) f' + G \geq 0.
\]

- For \( x_1 \in H \), we have \( v_{x_1}(f') = v_{x_1}(f) - v_{x_1}(\xi) = v_{x_1}(f) - 1 \geq 0 \), since \( f(x_1) = 0 \).

  We also have \( v_{x_1}(G) = 0 \), since \( x_1 \notin \text{Supp}(G) \). Thus \( v_{x_1}(f') + v_{x_1}(\xi) \geq 0 \).
• For \( x_1 \notin H \), we have \( v_{x_1}(f') = v_{x_1}(f) - v_{x_1}(\xi) \geq v_{x_1}(f) \), since \( \xi \) does not vanish outside \( H \). Thus \( v_{x_1}(f') + v_{x_1}(G) \geq v_{x_1}(f) + v_{x_1}(G) \geq 0 \), since \( f \in \mathcal{L}(G) \).

This completes the \( m = 1 \) case.

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,

\[
f_x(X_1, \ldots, X_{m-1}) \overset{\text{def}}{=} f(X_1, \ldots, X_{m-1}, x) = \sum_{i=1}^{m-1} f_{x,i}(X_1, \ldots, X_{m-1}) \xi(X_i),
\]

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

Since \( \deg G \geq |H| + 2g(C) - 1 \), Lemma 4.7.3 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

\[
\forall x \in H, \quad h(X_1, \ldots, X_{m-1}, x) = 0.
\]  

(4.17)

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

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

(4.18)

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

By (4.17) and the linear independence of the \( \beta_i \), we get that \( a_i(x) = 0 \) for each \( i \in [\ell], x \in H \). By the \( m = 1 \) case, we have that The \( \beta_i \)'s are linearly independent so their only linear combination which is identically zero is the trivial one, combining with (4.17) we get that \( a_i(x) = 0 \) for each \( i \in [\ell], x \in H \). By the \( m = 1 \) case that we proved above, we have that

\[
a_i(X_m) = b_i(X_m) \cdot \xi(X_m), \quad \text{for some } b_i(X_m) \in \mathcal{L}(G)
\]

(4.19)

Define

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

Then (4.18) and (4.19) imply that \( h(X_1, \ldots, X_m) = f''_m(X_1, \ldots, X_m) \cdot \xi(X_m) \).
Thus

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

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.

4.7.5 Local Derivatives and multiplicities

We have so far shown an AG Combinatorial Nullstellensatz for splitting sets. However a general set of points \( H \) might not be a splitting set. A natural starting point for showing a Nullstellensatz-like theorem for such sets would be to find some function \( \xi_H \) which is zero on \( H \), and has only a few other zeroes (i.e. is of low degree). Such a \( \xi_H \) might have zeroes on points outside of \( H \), and it might also have zeroes of high multiplicity (\( v_x(\xi_H) > 1 \)).

In this section we develop the technical tools for handling such difficulties, we will then use these tools in the following sections to prove a multiplicity version of the splitting sets Nullstellensatz (Section 4.7.6) and then the full AG Combinatorial Nullstellensatz (Section 4.7.7).

From now on (until Section 4.7.7), we assume that the ground field \( K \) is algebraically closed. This will allow us to work with power series. Later we will reduce the case of general perfect fields to the case of algebraically closed fields.

Let \( x \) be a rational point on \( C \) and let \( t_x \) be a local parameter for \( x \). Every rational function \( f \) on \( C \) which is regular at \( x \) can be written as \( f = \sum_{i=0}^{\infty} a_i t_x^i \), with\(^6\) each \( a_i \in K \). 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 \( \hat{O}_x \), the completion of the local ring \( O_x \)).

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 \( i^{th} \) 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.

\(^6\)Such power series representations exist at every degree 1 place of an algebraic curve over an arbitrary field (see [St93, Theorem 4.2.6]). Since \( K \) is algebraically closed, every point \( x \) on \( C \) corresponds to a degree 1 place.
Remark. 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:

\[
f(X_1, \ldots, X_m) = \sum_{l \in \mathbb{Z}^m, l \geq 0} a_l \prod_{j=1}^m t_{x,j}^l (X_j),
\]

(4.20)

with \( a_l \in K \) (again this equality is in the completion \( \hat{\mathcal{O}}_{(x_1, \ldots, x_m)} \) of the local ring \( \mathcal{O}_{(x_1, \ldots, x_m)} \)). Then we define, for \( i \in \mathbb{Z}^m \), the \( i \)’th 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 \( a_l \) in the above representation, there exists some \( j \in [m] \) for which \( i_j \geq e_j \).

We next show that the statement “individual multiplicity \( \geq (e_1, \ldots, e_m) \)” is independent of choice of local parameters.

**Theorem 4.17** (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.** We first fix some notation. We will use \( t \) and \( s \) to represent the vectors of local parameters \( t_{x,1}(X_1), t_{x,2}(X_2), \ldots, t_{x,m}(X_m) \) and \( s_{x,1}(X_1), s_{x,2}(X_2), \ldots, s_{x,m}(X_m) \) respectively. We will use \( j, l \) to denote vectors in \( \mathbb{Z}^m \). Finally, we use \( t^{j} \) to denote the monomial \( \prod_{i=1}^m t_{x,i}^{j_i}(X_i) \).

We will first show that the monomial \( t^{j} \), when represented as a power series in \( s \), only has non-zero coefficients for the monomials \( s^l \) for those \( l \) satisfying \( j_i \leq l_i \) for each \( i \in [m] \).

To do this we utilize the fact that \( t_{x,i}(X_i) = s_{x,i}(X_i) \cdot u_i(X_i) \) for some \( u_i(X_i) \) with \( v_{x,i}(u_i) = 0 \) (as both \( t_{x,i} \) and \( s_{x,i} \) are local parameters). Thus \( t^{j} = s^{l} \prod_{i=1}^m u_i(X_i)^{j_i} \). We then represent each \( u_i(X_i) \) as power series in \( s_i(X_i) \). Since \( v_{x,i}(u_i) = 0 \), each \( u_i(X_i) \), when represented as a power series in \( s_i(X_i) \), has no terms with negative powers. So we

---

**Important remark:** Even though we wrote (4.20) as a power series in the \( t_{x,i}(X_i) \), using the formal variables \( X_1, \ldots, X_m \), it does not make sense to substitute values for the \( X_j \). The only reason for using the variables \( X_1, \ldots, X_m \) in this expression is to distinguish between the \( m \) coordinates.

83
get that each monomial $s^l$ appearing in the resulting power series will have $j_i \leq l_i$, for each $i \in [m]$.

Now suppose $f$ has individual multiplicity $\geq (e_1, \ldots, e_m)$ at $(x_1, \ldots, x_m)$ when represented in $t$. Then for each monomial $a^l$ appearing in the power series representation of $f$ with a nonzero coefficient, we know that there is an $i$ such that $j_i \geq e_i$. When shifting to a power series in $s$, in every monomial that was induced by $a^l$ we get that the power of $s$ is at least $j_i$, and hence at least $e_i$. Thus the power series representation of $f$ in $s$ also has the property of having individual multiplicity $\geq (e_1, \ldots, e_m)$ at $(x_1, \ldots, x_m)$.

Thus the notion of individual multiplicity does not depend on the choice of local parameters. □

Here are some important remarks about individual multiplicity.

- 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 vacuously satisfied (provided $f$ is regular at the point).

- 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 $\geq (e_1, \ldots, e_m)$”.

- Suppose $x_1, \ldots, x_m \in \mathcal{C}$ and $f_1(X), f_2(X), \ldots, f_m(X)$ are functions on $\mathcal{C}$. Let $e'_i = v_{x_i}(f_i)$. Then the function $f(X_1, \ldots, X_m)$ on $\mathcal{C}^m$ given by

$$f(X_1, \ldots, X_m) = \prod_{i=1}^{m} f_i(X_i)$$

vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity $\geq (e_1, \ldots, e_m)$ if and only if $e'_i \geq e_i$ for some $i \in [m]$.

- If $f(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity $\geq (e_1, \ldots, e_m)$ and $f'(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity $\geq (e'_1, \ldots, e'_m)$, then $f \cdot f'(X_1, \ldots, X_m)$ vanishes at $(x_1, \ldots, x_m)$ with individual multiplicity $\geq (e_1 + e'_1, \ldots, e_m + e'_m)$.

Recall from Remark 17 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 $(X_{m'+1}, \ldots, X_m)$. Evaluating this function at some point will give us a power series in $t_{x_1}(X_1), \ldots, t_{x_{m'}}(X_{m'})$ (i.e. an object in the completion ring $\hat{O}_{(x_1, \ldots, x_{m'})}$).

84
Suppose \( x_1, \ldots, x_{m'} \in \mathcal{C} \). Write:

\[
f(X_1, \ldots, X_m) = \sum_{i \in \mathbb{Z}^{m'}, i \geq 0} a_i(X_{m'+1}, \ldots, X_m) \prod_{j=1}^{m'} t_{ij}(X_j).
\]

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

### 4.7.6 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 \( \xi \) may have zeroes of multiplicity \( > 1 \). The equivalent result for polynomials was shown in [KR12].

**Theorem 4.18** (AG Combinatorial Nullstellensatz for splitting sets with multiplicity). Let \( \mathcal{C} \) be an algebraic curve over a field \( K \), \( G > 0 \) be a rational divisor on it, and \( D > 0 \) 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 \( D = (\xi)_0(X) \). Let \( f(X_1, \ldots, X_m) \in \mathcal{L}^m(G) \).

---

8 Again, the variables \( X_j \) are present in the right hand side of this formula only to distinguish the coordinates. Substituting values for the \( X_j \) in the right hand side does not make sense.
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 \((x_1, \ldots, x_m) \in \text{Support}(D)^m\) 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 4.13, we will need a lemma on interpolation, this time with multiplicity.

**Lemma 4.7.5 (Interpolation with multiplicity).** Let \( G, D \) be divisors such that \( \deg G \geq \deg D + 2g(C) - 1, \ D \geq 0 \) and \( \text{Support}(D) \cap \text{Support}(G) = \emptyset \). Suppose we are given, for each \( x \in \text{Support}(D) \) 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 \( x \in \text{Support}(D) \) and each \( j < v_x(D) \), we have \( f^{(j)}(x) = c_{x,j} \).

**Proof.** Consider the mapping \( \varphi : \mathcal{L}(G) \rightarrow \prod_{x \in \text{Support}(D)} K^{v_x(D)} \) defined by

\[ \varphi(f) = \left( \left( f^{(i)} \right)_{x \in \text{Support}(D)} \right) \]

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

The dimension of \( \mathcal{L}(G) \) equals \( \deg(G) + 1 - g \), by the Riemann-Roch theorem (since \( \deg(G) \geq 2g - 1 \)).

The mapping \( \varphi \) is linear and has kernel \( \mathcal{L}(G - D) \), so by the Riemann-Roch theorem, the dimension of the kernel is \( \deg(G - D) + 1 - g \) (since \( \deg(G - D) \geq 2g - 1 \)). So the dimension of the image of \( \varphi \) is exactly \( \deg D \), which equals the dimension (as a vector space over \( K \)) of the space \( \prod_{x \in \text{Support}(D)} K^{v_x(D)} \).

Thus \( \varphi \) is surjective, and the result follows. \( \square \)

We now prove Theorem 4.18. The proof closely follows the proof in the case of splitting sets without multiplicity (Lemma 4.13).

**Proof.** (of Theorem 4.18) We start with the easy direction. 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 \((x_1, \ldots, x_m) \in \text{Support}(D)^m\) 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.

We prove the other direction by induction. For \( m = 1 \), suppose that for each \( x_1 \in SUPP(D), f(X_1) \) vanishes with multiplicity at least \( v_{x_1}(D) = v_{x_1}(\xi(X_1)) \). We
take $f'_1(X_1) = \frac{f(X_1)}{\xi(X_1)}$. Clearly (4.22) holds, and so we only have to prove that $f'_1 \in L(G)$.

In other words, we have to show that $f'_1 \in L(G)$. Then we can write

$$h(X_1, ..., X_m) = \sum_{i=1}^{\ell} \beta_i(X_1, ..., X_m) a_i(X_m)$$

(4.25)

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

This completes the $m = 1$ case.

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

For every $x \in C$ and every $j < v_x(D)$, we deduce from Proposition 4.7.4 that $f^{(0,0,...,0,j)}(X_1, ..., X_{m-1}, x)$ vanishes at each $(x_1, ..., x_{m-1}) \in \mathcal{C}^{m-1}$ with individual multiplicity at least $(v_{x_1}(D), ..., v_{x_{m-1}}(D))$. So by induction, there exist $f'_{x,j,v}(X_1, ..., X_{m-1}) \in L^{m-1}(G)$ such that:

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

(4.23)

For each $x \in \text{Support}(D)$ and each $j < v_x(D)$, let $\delta_{x,j}(X_m) \in L(G)$ be such that for each $y \in \text{Support}(D)$ 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 4.7.5, since $\deg G \geq \deg D + 2g(C) - 1$. Define

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

(4.24)

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

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

(4.25)

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

87
Then for each \( x \in \text{Support}(D) \), \( j < v_x(D) \), we have:

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

Therefore, using the fact that the \( \beta_i \) form a basis for \( \mathcal{L}^{m-1} \), 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

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

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

Thus from (4.23) and (4.24) we get

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

and using (4.25) and (4.26) we conclude

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

as desired. \( \square \)

Remark. A virtually identical proof would work for the case where we have \( m \) different \( \xi \)'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: \( f_i \xi(X_i) \in \mathcal{L}^m(G) \).

4.7.7 Proof of the AG Combinatorial Nullstellensatz for general sets

In this section, we will prove the AG Combinatorial Nullstellensatz for general sets, Theorem 4.14. Our strategy is as follows. Recall that we want to show the existence of functions \( \xi_1, \xi' \in \mathcal{L}(G) \) 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 \( \xi \) (of degree not much more than \( |H| \)) that vanishes on \( H \). For this sketch, we
further assume that all the zeroes of $\xi$ are of multiplicity 1. Let $J$ be the zeroes of $\xi$ not in $H$ (thus the zero set of $\xi$ equals $J \cup H$). Then we take $\xi'$ 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 $H^m$ if the function $f(X_1, \ldots , X_m) \cdot \prod_{i=1}^{m} \xi'(X_i)$ vanishes on $(H \cup J)^m$. Since $H \cup J$ is a splitting set (it is the set of zeroes of $\xi$, by assumption), we may apply the AG Combinatorial Nullstellensatz for splitting sets (Theorem 4.13) and complete the proof. Even if the zeroes of $\xi$ are not all of multiplicity 1, a very similar strategy works, using the multiplicity version of the AG Combinatorial Nullstellensatz for splitting sets (Theorem 4.18) instead.

We now give the details of the proof. The following two interpolation lemmas will be needed.

**Lemma 4.7.6** (Interpolation with a nonzeroness condition). Let $G$ be a nonnegative divisor and $H$ a set of points satisfying $H \cap \text{Support}(G) = \emptyset$ and $\deg G \geq |H| + 2g(C) - 1$. Then there exists $\xi \in \mathcal{L}(2G)$ such that $\xi|_H = 0$ and $\text{Support}((\xi \cdot |_{G}) \cap \text{Support}(G) = \emptyset$.

**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 $\xi_i$ vanishes on $H$, and that for each $i$, $\xi_i$ must have a pole at $P_i$ of order $n_i + 1$.

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

**Lemma 4.7.7.** Let $G$ and $D$ be nonnegative divisors with disjoint support. Let $H \subseteq C$ be a set of points with $H \subseteq \text{Support}(D)$. Suppose $\deg(G) \geq \deg(D) + 2g - 1$. Then there exists $\xi' \in \mathcal{L}(G)$ such that:

- For each $x \in H$, $v_x(\xi') = v_x(D) - 1$.
- For each $x \in \text{Support}(D) \setminus H$, $v_x(\xi') \geq v_x(D)$.

**Proof.** This lemma follows quite easily from Lemma 4.7.5. Nevertheless, we give a different (and simpler) proof based on the Riemann-Roch theorem. This will be important later when we consider non-algebraically-closed fields.
Consider the space \( \mathcal{L}(G - D) \). By the Riemann-Roch theorem, the dimension of the space \( \mathcal{L}(G - D) \) equals 
\[
a_{\text{def}} = \deg(G - D) - g + \ldots + m),
\]
where \( a \neq 0 \) and \( u \) vanishes at \((x_1, \ldots, x_m)\) with individual multiplicity \( \geq (v_{x_1}(D), \ldots, v_{x_m}(D)) \).

\(90\)

We now prove Theorem 4.14 over an algebraically closed field \( K \), which ensures that all points are of degree 1 (and so we can apply the derivative machinery from the previous section).

**Proof.** of Theorem 4.14 over algebraically closed fields Let \( \xi \in \mathcal{L}(2G) \) be given by Lemma 4.7.6 such that \( \xi(x) = 0 \) for each \( x \in H \) and \( \text{Support}((\{0\} \xi)) \cap \text{Support}(G) = \emptyset \).

Let \( D = (\{0\} \xi) \). Let \( \xi' \in \mathcal{L}(3G) \) be given by Lemma 4.7.7 be such that
\[
v_x(\xi') = \begin{cases} v_x(D) - 1 & x \in H \\ v_x(D) & x \in \text{Support}(D) \setminus H \end{cases}
\]

Such a \( \xi' \) exists because \( \deg(3G) \geq \deg(2G) + 2g - 1 \geq \deg(D) + 2g - 1 \). This completes the construction of \( \xi \) and \( \xi' \). We now proceed to prove that \( \xi \) and \( \xi' \) satisfy the conclusions of the theorem.

Define \( \Lambda'(X_1, \ldots, X_m) = \prod_{j=1}^m \xi'(X_j) \). Notice that \( \Lambda'(X_1, \ldots, X_m) \in \mathcal{L}^m(3G) \). It is useful to note that for each \( x \in H \), the local expansion of \( \xi'(X) \) at \( x \) begins with \( at_x(X)^{v_x(D) - 1} \), for some nonzero \( a \in K \). Therefore, for \((x_1, \ldots, x_m) \in H^m \), the local expansion of \( \Lambda'(X_1, \ldots, X_m) \) at \((x_1, \ldots, x_m)\) is of the form
\[
\Lambda'(X_1, \ldots, X_m) = a \prod_{j=1}^m t_{x_j}(X_j)^{-1} + u(X_1, \ldots, X_m),
\]
where \( a \neq 0 \) and \( u \) vanishes at \((x_1, \ldots, x_m)\) with individual multiplicity \( \geq (v_{x_1}(D), \ldots, v_{x_m}(D)) \).

\(90\)
We now prove the easy direction of the theorem. 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:

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

Observe that the right hand side vanishes at each \((x_1, \ldots, x_m) \in \text{Support}(D)^m\) 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 (4.28) the local expansion of \(f(X_1, \ldots, X_m) \cdot \Lambda'(X_1, \ldots, X_m)\) at \((x_1, \ldots, x_m)\) is of
the form:

\[
f(X_1, \ldots, X_m) \cdot \Lambda'(X_1, \ldots, X_m) = a' \cdot a \prod_{j=1}^{m} t_{x_j}^{v_{x_j}(D)-1} + a' \cdot u(X_1, \ldots, X_m),
\]

and thus \(f(X_1, \ldots, X_m) \cdot \Lambda'(X_1, \ldots, X_m)\) 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
\((x_1, \ldots, x_m) \in H^m\).

In the other direction, 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 \(f(X_1, \ldots, X_m) \cdot \Lambda'(X_1, \ldots, X_m) \in \mathcal{L}^m(4G)\).

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

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

2. For \((x_1, \ldots, x_m) \in H^m\), 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 (4.28), we get that in the power series representation of \(f(X_1, \ldots, X_m) \cdot \Lambda'(X_1, \ldots, X_m)\) at \((x_1, \ldots, x_m)\), each monomial has at least one \(i\) such that the
power of \(t_{x_i}(X_i)\) is \(\geq v_{x_i}(D)\). Thus \(f(X_1, \ldots, X_m) \cdot \Lambda'(X_1, \ldots, X_m)\) vanishes at
\((x_1, \ldots, x_m)\) with individual multiplicity at least \((v_{x_1}(D), \ldots, v_{x_m}(D))\).

By the Multiplicity Nullstellensatz (Theorem 4.18), 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:

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

as desired.

This completes the proof over an algebraically closed field \(K\).

Before proving Theorem 4.14 over an arbitrary perfect field, we explain our strategy.
Let $K^+$ be a perfect field, and let $K$ denote its algebraic closure. We will use the theorem we just proved over $K$ to deduce the theorem over $K^+$. This reduction is going to be based on the following basic linear algebraic fact: if a system of linear equations with coefficients in $K^+$ has a solution with entries in $K$, then it also has a solution with entries in $K^+$.

To explain the idea of our reduction, we will demonstrate this idea in detail in a highly simplified setting. Consider the following two statements (both very standard facts about polynomials over the rational numbers $\mathbb{Q}$ and the complex numbers $\mathbb{C}$).

- **Statement A**: Let $S \subseteq \mathbb{C}$ be a finite set. Let $P(T) \in \mathbb{C}[T]$ be the polynomial $\prod_{\alpha \in S} (T - \alpha)$. Then for every polynomial $Q(T) \in \mathbb{C}[T]$ of degree at most $d$, we have that $Q(T)$ vanishes on each point of $S$ if and only if there exists a polynomial $R(T) \in \mathbb{C}[T]$ of degree at most $d$ such that $Q(T) = P(T) \cdot R(T)$.

- **Statement B**: Let $S^+ \subseteq \mathbb{Q}$ be a finite set. Let $P^+(T) \in \mathbb{Q}[T]$ be the polynomial $\prod_{\alpha \in S^+} (T - \alpha)$. Then for every polynomial $Q^+(T) \in \mathbb{Q}[T]$ of degree at most $d$, we have that $Q^+(T)$ vanishes on each point of $S^+$ if and only if there exists a polynomial $R^+(T) \in \mathbb{Q}[T]$ of degree at most $d$ such that $Q^+(T) = P^+(T) \cdot R^+(T)$.

Let us show how to prove Statement B, assuming that we know Statement A.

Consider a finite set $S^+ \subseteq \mathbb{Q}$. Let $P^+(T) = \prod_{\alpha \in S^+} (T - \alpha)$. Take a polynomial $Q^+(T) \in \mathbb{Q}[T]$. When does $Q^+$ have the property that there exists $R^+(T) \in \mathbb{Q}[T]$ with $Q^+(T) = P^+(T) \cdot R^+(T)$? We can express this condition as the existence of a solution to a system of linear equations. Indeed, let

$$R^+(T) = R^+_0 + R^+_1 T + \ldots + R^+_d T^d,$$

where the $R^+_i$ are formal variables. Write both sides of the equation $Q^+(T) = R^+(T) \cdot P^+(T)$ in the basis $1, T, T^2, \ldots$, and equate coefficients. What we get is a system of equations system of linear equations with coefficients in $\mathbb{Q}$ in the variables $R^+_0, \ldots, R^+_d$; the existence of a solution to this system in $\mathbb{Q}^{d+1}$ is equivalent to the existence of such a polynomial $R^+(T)$. Let us call this system of linear equations $(\ast^+)$.

We want to show that the two properties (1) $Q^+(T)$ vanishes on all points of $S^+$, and (2) $Q^+(T)$ is such that there exists $R^+(T) \in \mathbb{Q}[T]$ with $Q^+(T) = R^+(T) \cdot P^+(T)$, are equivalent. Treat the given set $S^+$ (which we know is a subset of $\mathbb{Q}$) as a subset of $\mathbb{C}$. As in the setup for Statement A, define $P(T) = \prod_{\alpha \in S} (T - \alpha)$, and note that $P(T) \in \mathbb{C}[T]$ is the same polynomial as $P^+(T) \in \mathbb{Q}[T]$. View $Q^+(T) \in \mathbb{Q}[T]$ as a polynomial $Q(T) \in \mathbb{C}[T]$. Trivially we have that $Q^+(T)$ vanishes on all points of $S^+$ if and only if $Q(T)$ vanishes on all points of $S$. By Statement A, this latter property is equivalent to the existence of a polynomial $R(T) \in \mathbb{C}[T]$ such that $Q(T) = R(T) \cdot P(T)$. As above, the existence of such an $R(T)$ can be expressed as the existence of a solution

\[\text{9Of course, } S \text{ is the same set as } S^+.\]
over \( \mathbb{C} \) to a certain system of linear equations, which we call (\( \ast \)). The crucial observation is that the system of linear equations (\( \ast \)) is exactly the same as the system of linear equations (\( \ast^+ \)). In particular, the coefficients of the system of equations (\( \ast \)) all lie in \( \mathbb{Q} \). Thus by the linear algebra fact mentioned earlier, the existence of a \( \mathbb{C} \)-solution to (\( \ast \)) is equivalent to the existence of a \( \mathbb{Q} \)-solution to (\( \ast \)), and hence to the existence of a \( \mathbb{Q} \)-solution to (\( \ast^+ \)) (since (\( \ast^+ \)) and (\( \ast \)) are same system of equations). This in turn is equivalent to the existence of \( R^+(T) \in \mathbb{Q}[T] \) such that \( Q^+(T) = R^+(T) \cdot P^+(T) \).

Putting this all together, we get Statement B.

We now implement this kind of argument to deduce the AG Combinatorial Nullstellensatz for perfect fields from the AG Combinatorial Nullstellensatz for algebraically closed fields.

**Proof.** Theorem 4.14 over a general perfect field Let \( K^+ \) be a perfect field, and let \( K \) be the algebraic closure of \( K^+ \).

Suppose, as in the hypothesis of the theorem, we are given an algebraic curve \( C^+ \) defined over \( K^+ \), a divisor \( G^+ \) on \( C^+ \), a function \( f^+ \in L^m(G^+) \), and a set of \( K^+ \)-rational places \( H^+ \subseteq C^+(K^+) \).

We want to view this setup in the algebraic closure \( K \). Formally, we are extending the field of constants of \( C^+ \) from \( K^+ \) to \( K \), see [Sti93, Section 3.6]. We get an algebraic curve \( C \) defined over \( K \), a divisor \( G \) on \( C \) with \( \deg(G) = \deg(G^+) \) (the high-degree points in \( G^+ \) split into many degree 1 points in \( G \)), a function \( f \in L^m(G) \), and a set of \( K^+ \)-rational points \( H \subseteq C \) (with \( |H| = |H^+| \)).

We list some important facts about this operation.

- The rational functions \( f \) and \( f^+ \) are intimately related: they are the same function! The relationship between \( f \) and \( f^+ \) is similar to the relationship between \( P \) and \( P^+ \) in the simpler argument above: the only difference between \( f \) and \( f^+ \) is that we view them as elements of different linear spaces. Thus we have that \( f \) vanishes on \( H^m \) if and only if \( f^+ \) vanishes on \( (H^+)^m \).

- The genus of \( C \) and the genus of \( C^+ \) are equal (by [Sti93, Proposition 3.6.3]): this is where we use the perfectness of \( K^+ \).

- Any \( K^+ \)-basis of \( L(G^+) \) is also a \( K \)-basis of \( L(G) \) (see [Sti93, Proposition 3.6.3]), and thus any \( K^+ \)-basis of \( L^m(G^+) \) is also a \( K \)-basis of \( L^m(G) \).

By the above, we have that \( f \) vanishes on all points of \( H^m \) if and only if \( f^+ \) vanishes on all points of \( (H^+)^m \). By the AG Combinatorial Nullstellensatz for algebraically closed fields, we know that \( f \) vanishes on all points of \( H^m \) if and only if there exist \( f'_1(X_1, \ldots, X_m), \ldots, f'_m(X_1, \ldots, X_m) \in L^m(4G) \) such that:

\[
f(X_1, \ldots, X_m) \cdot \prod_{i=1}^{m} \xi'(X_i) = \sum_{i=1}^{m} f'_j(X_1, \ldots, X_m) \cdot \xi(X_i). \quad (4.29)
\]
We will now show that such \( f_1'(X_1, \ldots, X_m), \ldots, f_m'(X_1, \ldots, X_m) \in \mathcal{L}^m(4G) \) exist if and only if there exist \((f_1 + \cdots + f_m)'(X_1, \ldots, X_m) = (f_1 + \cdots + f_m)'(X_1, \ldots, X_m) \in \mathcal{L}^m(4G^+) \in \mathcal{L}^m(4G^+)\) such that:

\[
f^+(X_1, \ldots, X_m) \cdot \prod_{i=1}^m \xi'(X_i) = \sum_{i=1}^m (f_i^+)'(X_1, \ldots, X_m) \cdot \xi(X_i).
\] (4.30)

The additional main fact that we will need is that the functions \( \xi \) and \( \xi' \), constructed in Theorem 4.14 as elements of \( \mathcal{L}(2G) \) and \( \mathcal{L}(3G) \) respectively, can in fact be chosen to be elements of \( \mathcal{L}(2G^+) \) and \( \mathcal{L}(3G^+) \) respectively (i.e., \( \xi \) and \( \xi' \) are \( K^+ \)-rational). This follows from our construction of \( \xi \) and \( \xi' \) via Lemma 4.7.6 and Lemma 4.7.7: only the Riemann-Roch theorem was used in the construction, and the application of the Riemann-Roch theorem to both \( C \) and \( C^+ \) yields identical formulas for the dimensions of the relevant Riemann-Roch spaces (since the genus of \( C \) equals the genus of \( C^+ \)).

Our strategy is the following. We will express Equations (4.29) and (4.30) as systems of linear equations with \( K^+ \) coefficients, and observe that these systems of equations are in fact identical. Thus by the linear algebraic fact mentioned earlier, (4.29) has a solution over \( K \) if and only if (4.30) has a solution over \( K^+ \). This is the desired equivalence.

Let \( \ell_1, \ldots, \ell_L \) be a basis for \( \mathcal{L}^m(4G^+) \). As mentioned above, this \( \ell_1, \ldots, \ell_L \) is also a basis for \( \mathcal{L}^m(4G) \). Let \((A_{ij} | i \in [m], j \in [L]) \) and \((A_{ij}^+ | i \in [m], j \in [L]) \) be collections of formal variables. Write

\[
(f_i)'(X_1, \ldots, X_m) = \sum_{j \in [L]} A_{ij}\ell_j,
\]

\[
(f_i^+)'(X_1, \ldots, X_m) = \sum_{j \in [L]} A_{ij}^+\ell_j.
\]

Consider the equation:

\[
f^+(X_1, \ldots, X_m) \cdot \prod_{i=1}^m \xi'(X_i) = \sum_{i=1}^m \xi(X_i) \cdot (f_i^+)'(X_1, \ldots, X_m).
\] (4.31)

Both sides of this equation are in \( \mathcal{L}^m(6G^+) \). We can choose a basis for \( \mathcal{L}^m(6G^+) \) express both sides of this equation in terms of this basis, and equate coefficients of the corresponding basis elements. This gives us a system of linear equations in the variables \( A_{ij}^+ \) with \( K^+ \) coefficients.

Now consider the equation:

\[
f(X_1, \ldots, X_m) \cdot \prod_{i=1}^m \xi'(X_i) = \sum_{i=1}^m \xi(X_i) \cdot f_i'(X_1, \ldots, X_m).
\] (4.32)

Both sides of this equation are in \( \mathcal{L}^m(6G) \). Using the above chosen basis for \( \mathcal{L}^m(6G^+) \)
(which is also a basis for $L^m(6G)$), we express both sides of this equation in terms of this basis, and equate coefficients of the corresponding basis elements. This gives us the same system of linear equations in the variables $A_{ij}$ with $K^+$ coefficients.

**Note:** Why are these systems of linear equations the same? The reason is that algebraic relations between the functions $\ell_i$, $\xi$, $\xi'$ and $f$ over $C^+$ are preserved when we view them over $C$ after extending the field of constants. This is just like the fact that identities between polynomials in $\mathbb{Q}[T]$ are preserved when we view them as polynomials in $\mathbb{C}[T]$.

Thus the system (4.29) has a solution over $K$ if and only if the system (4.30) has a solution over $K^+$, as desired.

This completes the proof of the AG Combinatorial Nullstellensatz over perfect fields.
Bibliography


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


[GKS09] Elena Grigorescu, Tali Kaufman, and Madhu Sudan. Succinct representation of codes with applications to testing. In Dinur et al. [DJNR09], pages 534–547.


תקציר
פונניאמעלשדותסופיים,והקודיםלתיקוןשגיאותשנוצריםמהם,הינםכליםשימושייםמאודבכלתחומיהתאוריהשל...

cלכלמשתנהבנפרד.כחלקמהבניהאנחנומראיםכיצדלהכלילאתמשפטהאפסיםהקומבינטורישלאלוןלמקרהשל 
פונניאמעעלעקומיםאלגראיים.
גרסאות מורבות בשתיים של קוד אונומטריה
אלגברה, עם שימור

יחבר על מחקר
ליש מליון חלקי של הדרישות להבנה התאורית
ודוקטור לפילוסופיה

יוחאי קפלן

הנציחacketsינו — מנדי זכויותינו לישראלי
נכתבحضניך, facing, 2016
גרסאות מורבויות של קוד אומטרה
אלגברה, עמ' שימושי

יוחאי קפלן