Probabilistically Checkable Proofs and the Testing of Hadamard-like Codes

by

Marcos Kiwi

Licenciado en Ciencias de la Ingeniería
Universidad de Chile (1990)

Ingeniero Civil Matemático
Universidad de Chile (1991)

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1996

Signature of Author [illegible]
Department of Mathematics
January 12, 1996

Certified by [illegible]
Michael Sipser
Professor of Mathematics
Thesis Supervisor

Accepted by [illegible]
Richard Stanley, Chairman
Applied Mathematics Committee

Accepted by [illegible]
David Vogan, Chairman
Departmental Graduate Committee

[redacted]

Probabilistically Checkable Proofs and the Testing of Hadamard-like Codes

by

Marcos Kiwi

Submitted to the Department of Mathematics on January 12, 1996,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Recent results concerning probabilistically checkable proofs (PCPs) enable the encoding of mathematical proofs so as to allow very efficient probabilistic verification. Probabilistic verification consists of a simple randomized test that looks at a few bits of the proof and decides to accept or reject the proof's validity by performing a simple computation on those bits. Valid proofs are always accepted. Incorrect proofs are rejected with a non-negligible probability.

The interest in PCPs stems from their complexity theoretic implications. Especially interesting are the surprising way in which they characterize NP and the hardness of approximation results derived through them. This thesis contributes to further our understanding of complexity theoretic issues through the study of PCPs.

We address a basic problem that arises in the construction of PCPs and the derivation of hardness of approximation results: linearity testing over the two element field. We give a thorough analysis of this problem. Some of our results are obtained by showing a new connection between testing over the field of size two and discrete Fourier analysis. This relationship is further illustrated by studying a pair of tests that are related to the Hadamard code test.

In addition, we establish a connection between testing and the theory of weight distributions of dual codes. This connection allows us to formulate a new way of testing for linearity over finite fields. We then show how to analyze, through the MacWilliams Theorems, the tests that fall into our framework. The results we obtain imply nontrivial facts about the functions being tested even when these functions fail the test with relatively large probability. We discuss why these types of results are desirable in the PCP context.

Finally, we propose two models of interactive proof systems. In the first of these proof systems, a computationally limited verifier interacts with a ranked sequence of oracles. In the second proof system, the verifier now interacts with a ranked sequence of provers. The distinguishing characteristic of these proof systems is the ordered relationship amongst the provers and amongst the oracles. We show that, with suitable restrictions on the randomness and query complexity, both proof systems exactly characterize each level of the polynomial hierarchy. The characterization via oracle proof systems extends that of NP via PCPs.

Thesis Supervisor: Michael Sipser
Title: Professor of Mathematics

Credits

The majority of the material in this thesis has appeared in conference proceedings. The material in Chapter 2, except Section 2.8.1 is joint work with Mihir Bellare, Don Coppersmith, Johan Håstad, and Madhu Sudan and appeared in [BCH+95]. Chapter 3 and Section 2.8.1 benefited greatly from discussions with Madhu Sudan, I wish to thank him for this. The material in Chapter 4 is joint work with Carsten Lund, Alexander Russell, Daniel Spielman, and Ravi Sundaram and appeared in [KLR+94].

Many people have contributed useful suggestions to me in this research. I am very grateful to all of them. I would especially like to thank Sanjeev Arora, Uri Feige, Joan Feigenbaum, Michael Goemans, Joe Kilian, Dan Kleitman, Carsten Lund, Michael Sipser, Daniel Spielman, Ravi Sundaram, and Mario Szegedy.

I thank Kristin Ardlie, Jon Kleinberg, S. Ravikumar, Ray Sidney, Daniel Spielman, and Ethan Wolf for useful suggestions in the write-up of this thesis. I am especially grateful to Ravi Sundaram for the many valuable comments.

Thanks to Oded Goldreich, Shafi Goldwasser, and Michael Sipser for participating in my thesis committee.

Acknowledgments

I am greatly indebted to my advisor Mike Sipser. It has been rewarding to experience Mike's approach to discovering and understanding new ideas. I also thank him for generously sharing his time and knowledge, for the many enjoyable conversations, and especially for the inspiring and illuminating ways in which he explained everything to me. Last, but not least, I thank Mike for his counsel on several professional issues. I cannot recall a time that I have regretted following his advice.

I will miss many things from MIT, certainly its challenging academic environment and its never ending sources of new wonders, but most of all, the opportunity that it gave me to meet so many wonderful people.

I was very lucky to have found, upon arrival, such good friends as Erol Çubukçu, Satomi Okazaki, and Ethan Wolf, and shortly afterwards Ray Sidney and Ravi Sundaram. I am also grateful for the stimulating friendship of Richard Ehrenborg, Mauricio Karchmer, Jon Kleinberg, Lillian Lee, Tal Rabin, Alex Russell, Dan Spielman, and Shang-Hua Teng. I especially owe thanks to Jon Kleinberg for good advice and to Dan Spielman for everything I learned from him.

During my studies I discovered and re-discovered many friends beyond MIT. Thus, they shall remain unnamed here. But, I thank them for the warmth and happiness they gave me.

My deepest gratitude is to my parents for the loving education they provided me. I also thank them and my sister for the stubbornness they showed in trying to fool the distance, and for being so supportive and loving in this long absence from home. I am especially indebted to my brother. Our frequent and long conversations were an invaluable source of friendship and wisdom.

I am grateful to the Department of Mathematical Engineering of the University of Chile, for their valuable support and the confidence they showed in me. I especially thank Eric Goles for his early encouragement and for his stimulating and unwavering energy.

Thanks to the MIT Mathematics Department and the Laboratory for Computer Science for all their assistance and for providing a great working environment.

I am grateful to AT&T Bell Laboratories for the fellowship it gave me. Thanks to David Johnson, my mentor, for among other things, making possible my summer visits to Bell Laboratories. These visits were a great experience, and a starting point for a lot of the work leading to this dissertation.

The research for this thesis was partially supported by NSF Grant CCR-9212184, NSF Grant CCR-9503322, MIT Applied Mathematics Fellowship (second year), and the AT&T Bell Laboratories PhD Scholarship (starting my third year).

Table of Contents

CHAPTER 1

Introduction

Computational complexity strives to understand what makes a problem hard to solve. We are far from achieving a comprehensive understanding of this issue; but, over the last few decades, a theory has emerged that considers particular computational tasks and asks: what is the minimum amount of certain resources (like time, space, non-determinism, randomness, parallelism, communication, etc.) needed to carry them out.

Recently, new complexity measures have been proposed and shown to play a crucial role in our comprehension of the resources necessary to efficiently solve combinatorial problems. These complexity measures have been identified and refined throughout a large body of work leading to and concerned with ***probabilistically checkable proofs*** (PCPs). Informally, a PCP is a proof that can be verified (probabilistically) to a significant degree of confidence with a small number of queries (spot-checks). This dissertation contributes to the determination of the quantitative and qualitative aspects concerning the randomness, queries, and achievable confidence with which it is possible to perform certain computational tasks.

Proofs of language membership play a fundamental role in the understanding of complexity theoretic issues. Central among these issues is the question of whether $P \neq NP$. This is one of the reasons for the interest in PCPs, given the surprising way in which they characterize NP as shown by Arora, Lund, Motwani, Sudan, and Szegedy [ALM$^+$92]. But, PCPs do more than just provide an alternative characterization of NP. To explain this, recall that since the development of the theory of NP completeness by Cook, Levin, and Karp [Coo71, Lev73, Kar72] it has been known that determining the optimum of many natural optimization problems is NP-hard. However, it was noted that good approximations would suffice for many applications. In fact, for several optimization problems good approximations were found. For many NP-optimization problems, however, no progress was made either on the approximation or the intractability front. This changed when Feige, Goldwasser, Lovász, Safra, and Szegedy [FGL$^+$91] showed that, under certain complexity theoretic assumptions, reasonable approximations of the clique number of a graph cannot be obtained efficiently. Their argument, coupled with the PCP techniques, has been exploited to show the intractability of approximating a wide range of optimization problems. In some cases tight results have been proved. This thesis contributes to the ongoing effort of determining the true non-approximability factors of combinatorial optimization problems.

Recently, Bellare, Goldreich, and Sudan [BGS95] have shown that the connection between PCPs and the hardness of computing the clique number of a graph can be reversed. They establish that if approximating the clique number of a graph up to a certain factor is NP-hard, then certain types of PCPs exist.[1] This shows that the understanding and refinement of the complexity theoretic measures that arise in the PCP context are inherently linked to the resolution of complexity theoretic problems. In addition, Arora [Aro95b] has shown that improving, beyond a certain threshold, the hardness of approximation factors via PCP techniques would raise questions regarding our understanding of classical notions of reducibility. One of our goals is to further our understanding of complexity theoretic issues through the study of PCPs.

A PCP has to be extremely redundant since every part of the proof has to witness the correctness or incorrectness of the whole. Otherwise a probabilistic verification that performs a small number of spot-checks would not detect incorrect proofs with a significant confidence. The error-detection properties that PCPs exhibit are inherited from the techniques used to construct them. To explain this, recall that since the work of Arora and Safra [AS92b], PCPs have been built by recursion. Each level of the recursion uses a

[1] It is out of the scope of this introduction to explain what the characteristics of these PCPs are. The interested reader is referred to [BGS95].

distinct form of error-correcting code. Correct encodings are viewed as representations of functions that satisfy a pre-specified property. Thus, a central problem in the construction of PCPs is to probabilistically check (test) function properties with as few oracle queries as possible. However, this problem was first formulated in a different context: that of program checking by Blum *et al.* [Blu88, BK89], and self-testing/correcting programs by Blum, Luby, and Rubinfeld [BLR90]. This dissertation contributes to our understanding of the coding theoretic and self-testing issues that naturally arise in the PCP context.

1.1 Overview of main results

CHAPTER 2. We address one of the most basic questions that arises in the construction of PCPs and the derivation of hardness of approximation results, namely, the linearity testing problem over the two element field. We give a thorough and nearly complete analysis of this problem. Some of our results are obtained by showing a new connection between testing over the finite field of size two and discrete Fourier analysis. We further illustrate this relationship by studying a pair of tests that are related to the Hadamard code test.

CHAPTER 3. We propose a framework in which to formulate and carry out the analyses of tests. This framework establishes a connection between testing and the theory of weight distributions of dual codes. We illustrate this connection by formulating a new way of testing for linearity over finite fields, namely the *extended linearity test*. We then derive, from the MacWilliams Theorems, results through which the tests that fall into our framework can be analyzed. The results we obtain imply nontrivial facts about the functions being tested even when these functions fail the test with relatively large probability. We also discuss why these sort of results are desirable in the PCP context.

CHAPTER 4. We propose a new model of interactive proof systems. In these proof systems a computationally limited verifier interacts with a finite number of provers. Provers are divided into two teams. One team is trying to convince the verifier to output accept and the other one to output reject. There is a ranking associated to the provers. Each prover knows the strategies that higher ranked provers use in answering the questions posed to them. Provers do not get to see other provers' messages.

We also study the proof systems obtained from the ones described in the previous paragraph by replacing the provers with oracles.

We show that, with suitable restrictions on the randomness and query complexity, both proof systems yield characterizations of each level of the polynomial hierarchy. The characterizations via oracle proof systems extends that of NP via PCPs. We also discuss some of the properties of the proof systems we propose and contrast them with the ones of previously defined proof systems.

1.2 Useful references

We will not attempt to explain the history of the many works leading to [ALM$^+$92] and the derivation of its many consequences. The interested reader is referred to the surveys of Babai [Bab92], Goldreich [Gol94] and Johnson [Joh88, Joh92]. For a detailed proof of the characterization of NP via PCPs the reader is referred to the dissertations of Arora [Aro94a] and of Sudan [Sud92]. For a thorough discussion of the hardness of approximation results derived from PCPs see the work of Bellare, Goldreich, and Sudan [BGS95]. A discussion of issues concerning program checking can be found in the survey of Blum and Wasserman [BW94] and the thesis of Rubinfeld [Rub90]. For general background on complexity theory the reader is referred to [BDG95, HU79, Pap94, Sip96, vL90].

CHAPTER 2

Linearity testing

Mathematicians ascertain the validity of their claims through proofs. They communicate their proofs in a semi-formal language, but, with the implicit understanding that their proofs can be made rigorous, i.e. they can be translated into a sequence of applications of very simple derivation rules to a collection of axioms. The validity of such proofs can be ascertained by checking that the initial assumptions correspond to the axioms, and that each step in the proof is a consequence of a valid derivation rule.

Recent results concerning probabilistically checkable proofs (PCPs) permit the encoding of mathematical proofs so as to allow very efficient probabilistic verification. Probabilistic verification means that there is a simple randomized *test* that looks at a few bits of the proof, and decides to accept or reject its validity by performing a simple computation on these bits. Valid proofs will always be accepted. Incorrect proofs will be rejected with a non-negligible probability.

PCPs are built by recursion. Each level of the recursion makes use of a different encoding. Each correct encoding describes a function that satisfies some characteristic property. Thus, a crucial problem in the construction of PCPs is how to verify (test) function properties with a small number of queries. Yet the problem itself is older, with the basic formulation first made in the context of program checking.

It is a feature of the area that while tests are easy to specify, they are generally hard to analyze, especially to analyze well. Yet, good analyses are, for several reasons, worth striving for. There is, first, the inherent mathematical interest of getting the best possible analysis and understanding of a well-defined combinatorial problem. But, there is a more pragmatic reason: better analyses typically translate into improved hardness of approximation results.

The particular testing problem that we study concerns the BLR (Blum-Luby-Rubinfeld) test. This test addresses one of the most basic questions, namely testing linearity. Our focus is the case of most importance in applications. Several analyses of this problem have appeared, yet none is tight. With each analysis comes an improvement in the non-approximability factor for MaxSNP problems that can be proved. But, the extent to which these factors can grow remains open. The results of this chapter contribute to the resolution of this question. Let us begin by describing the problem more precisely.

2.1 Linearity testing in characteristic two

2.1.1 The problem

In order to discuss past work it will be useful to begin explaining the problem of linearity testing over arbitrary finite groups. Thus, let G and H be finite groups, and recall that a function $f\colon G \to H$ is ***linear*** if $f(u) + f(v) = f(u+v)$ for all $u, v \in G$. (That is, f is a group homomorphism.) Here are some basic definitions:

$\mathsf{Dist}(f,g) \stackrel{\text{def}}{=} \Pr_{u\in_R G}[\, f(u) \neq g(u)\,]$ — (relative) distance between $f, g\colon G \to H$

$\mathsf{Dist}(f) \stackrel{\text{def}}{=} \min\{\, \mathsf{Dist}(f,l) \mid l\colon G \to H \text{ is linear}\,\}$ — distance of f to its closest linear function.

We are given oracle access to a function f mapping G to H. (That is, we can specify $u \in G$ and in one step are returned $f(u) \in H$.) We want to test that f is close (in relative distance) to a linear function. We are charged for each oracle call.

THE BLR TEST. The BLR test is the following [BLR90]— pick $u, v \in G$ at random, query the oracle to obtain $f(u), f(v), f(u+v)$, and reject if $f(u) + f(v) \neq f(u+v)$. Let

$$\mathsf{Rej}(f) \stackrel{\text{def}}{=} \Pr_{u,v\in_R G}[\, f(u) + f(v) \neq f(u+v)\,]\,.$$

Thus, $\mathsf{Rej}(f)$ denotes the probability that the BLR test rejects f. The issue in linearity testing is to study how $\mathsf{Rej}(f)$ behaves as a function of $\delta = \mathsf{Dist}(f)$. In particular, we would like to know what is the smallest probability of the BLR test rejecting a function that is at distance δ from the space of linear functions, i.e. to derive good lower bounds on $\mathsf{Rej}(f)$ as a function of δ. In other words, we want to determine the form of the *linearity testing curve.* Equivalently, for every $\delta \in [0,1]$ we want to find the value

$$\Gamma_{G,H}(\delta) \stackrel{\text{def}}{=} \min\{\, \mathsf{Rej}(f) \mid f\colon G \to H \text{ s.t. } \mathsf{Dist}(f) = \delta \,\}\,.$$

2.1.2 Previous work

Blum, Luby, and Rubinfeld [BLR90] first investigated the shape of the linearity testing curve. They addressed the problem in the general context where G and H are arbitrary finite groups. Their analysis showed that $\Gamma_{G,H}(\delta) \geq 2\delta/9$. Interest in the tightness of the analysis begins with Bellare, Goldwasser, Lund, and Russell [BGLR93] in the context of improving non-approximability factors for MaxSNP problems. They showed that $\Gamma_{G,H}(\delta) \geq 3\delta(1-2\delta)$. Meanwhile, it was observed in [BS94] that the arguments in [BLR90] implied that if $\Gamma_{G,H}(\delta) \leq 2/9$, then $\Gamma_{G,H}(\delta) \geq 2\delta/3$. The last two bounds supersede the first. So, the following theorem captures the state of knowledge.

Theorem 2.1.1 [BGLR93, BS94] Let G and H be arbitrary finite groups. Then:

1. $\Gamma_{G,H}(\delta) \geq 3\delta(1-2\delta)$.

2. If $\delta \geq 1/3$, then $\Gamma_{G,H}(\delta) \geq 2/9$.

Figure 2-1 summarizes the state of knowledge concerning the BLR test previous to this work. Note that the best bound implied by previous work decreases when $\delta \geq 1/4$.

THE KNEE OF THE LINEARITY TESTING CURVE. In [ALM+92, BGLR93, BS94] one parameter was identified as important in connection with MaxSNP hardness results. This parameter is a single number. It is defined by

$$\mathsf{Knee}_{\mathsf{G,H}} \stackrel{\text{def}}{=} \min\{\, \Gamma_{\mathsf{G,H}}(\delta) \mid \delta \geq 1/4 \,\} \text{ — knee of the linearity testing curve.}$$

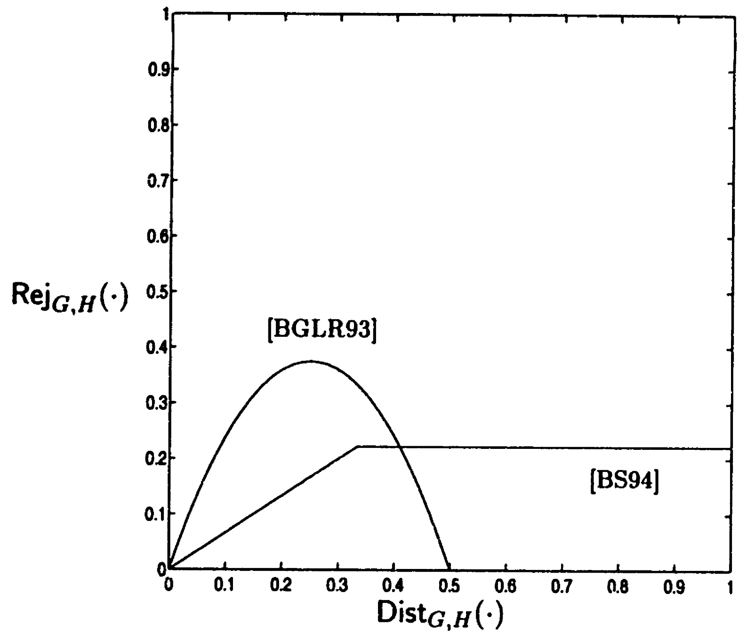

Figure 2-1: The best lower bound for the BLR test that can be gleaned from the existing literature previous to this work.

Improvements (increases) in the lower bound that can be shown on $\mathsf{Knee}_{\mathsf{G,H}}$ translate directly, via [BGS95], into improved (increased) non-approximability results for some NP-optimization problems. They also translate into improved (decreased) probe complexity in the characterizations of NP via PCPs. Exactly how or why this is the case is outside the scope of this discussion.

As indicated, improved lower bounds for the knee lead to better non-approximability results. But, in this general setting, we can do no better than the $\mathsf{Knee}_{\mathsf{G,H}} \geq 2/9$ bound suggested by Theorem 2.1.1. An example due to Coppersmith shows that the previous bound is in fact tight in the case of general groups. Indeed, let m be a positive integer divisible by 3 and let f be the function from $\mathcal{Z}_m^n$ to $\mathcal{Z}_m$ such that at $u = (\, u_1, \ldots, u_n \,)$ takes the value $f(u) = 3k$, if $u_1 \in \{\, 3k-1, 3k, 3k+1 \,\}$. Observe that $f(u) + f(v) \neq f(u+v)$ only if $u_1 = v_1 = 1 \pmod 3$, or $u_1 = v_1 = -1 \pmod 3$, i.e. $\mathsf{Rej}(f) = 2/9$. Furthermore, $\mathsf{Dist}(f) = 2/3$. This coupled with Theorem 2.1.1 shows that $\mathsf{Knee}_{\mathsf{G,H}} = 2/9$. Coppersmith's example leads into our research. We note that the problem to which linearity testing is applied in the proof system constructions of [ALM+92, BGLR93, BS94] is that of testing

Hadamard codes, and the long code in the proof systems derived in [BGS95]. But, this corresponds to the linearity testing problem in the special case where $G = \mathrm{GF}(2)^n$ and $H = \mathrm{GF}(2)$. Here, G is regarded as an additive group in the obvious way. Namely, the elements are viewed as n-bit strings or vectors over GF(2) and operations are component-wise over GF(2). For this case, the example of Coppersmith does *not* apply, and we can hope for better results.

2.1.3 Main results

We look at the performance of the BLR test when the underlying groups are $G = \mathrm{GF}(2)^n$ and $H = \mathrm{GF}(2)$, where n is a positive integer. For notational simplicity we now drop the groups G, H from the subscripts, writing $\Gamma(\delta)$ and Knee — it is to be understood that we mean $G = F^n$ and $H = F$, where from now on, unless explicitly said otherwise, F denotes GF(2).

We provide two new analyses of the linearity testing problem over the field of size two. The first of them establishes a connection between linearity testing and discrete Fourier analysis. (This connection has been further exploited by Håstad [Hås95] in the analysis of a new test for the long code [BGS95].) The casting of the linearity testing problem in the language of Fourier series enables us to study the BLR test. The outcome is the following:

Theorem 2.1.2 $\Gamma(\delta) \geq \delta$.

Apart from lending a new perspective to the linearity testing problem, the result exhibits a feature which distinguishes it from almost all previous results. Namely, it shows that $\Gamma(\delta)$ increases with δ and in fact is 1/2 at $\delta = 1/2$.[1] (According to the previous analysis, namely Theorem 2.1.1, $\Gamma(\delta)$ may have been bounded above by 2/9 for all $\delta \geq \delta_0$, where δ_0 is the larger root of the equation $3\delta(1-2\delta) = 2/9$.)

The proof of Theorem 2.1.2 captures the intuition that the probability with which a function fails the BLR test depends on how far away that function is from *each* linear function. This intuition is not elicited by the other arguments that have been used to study

[1] Note that $\mathrm{Dist}(f) \leq 1/2$ for all $f\colon F^n \to F$ because we are working over GF(2), so only the portion $\delta \in [0, 1/2]$ of the curve is interesting.

not only the BLR test, but also other tests. We think that the strength of the above referred lower bound stems from the fact that our proof argument substantiates this intuition.

Theorem 2.1.2 combined with the first part of Theorem 2.1.1 shows that $\mathsf{Knee} \geq 1/3$. However, this is not tight. Our second analysis of the BLR test focuses on finding the value of the knee. This leads us to an isoperimetric problem about a 3-regular hypergraph on the vertices of the n-dimensional hypercube. We state and prove a lemma, henceforth referred to as the Summation Lemma, which provides a tight isoperimetric inequality for this problem. We then use this result to determine the exact value of the knee of the linearity testing curve.

Theorem 2.1.3 $\mathsf{Knee} = 45/128$.

We also analyze the tightness of the lower bounds we obtain. The following result, summarizes the state of knowledge regarding the linearity testing curve when the underlying field is the two element field.

Corollary 2.1.4 For every $\delta \in [0, 5/16]$, $\Gamma(\delta) = 3\delta(1-2\delta)$. Moreover, if $\delta \in [5/16, 1/2]$, then $\Gamma(\delta) \geq \max\{45/128, \delta\}$.

SUMMARY. Figure 2-2 summarizes the results of this chapter concerning the BLR test. The points $\{(\mathsf{Dist}(f), \mathsf{Rej}(f)) \mid f\colon F^n \to F\}$ lie in the white region of the graph. The darkly shaded region represents the forbidden area before our work. The lightly shaded region represents what we add to the forbidden area. Note we both extend the lower bound and provide upper bounds. The dots are actual computer constructed examples; they indicate that perhaps the lower bound may be improved, but not by much.[2] In particular, the knee value is tight. Also, the upper bound is tight.

2.1.4 Relationship to other work

APPLICATION TO MaxSNP HARDNESS. Usage of the linearity test in the construction of efficient PCPs, and in the derivation of hardness of approximability results for MaxSNP

[2] More precisely, we have a randomized procedure that with high probability can construct, for each plotted point, a function f such that $(\mathsf{Dist}(f), \mathsf{Rej}(f))$ is arbitrarily close to the point in question (details are given in Appendix A).

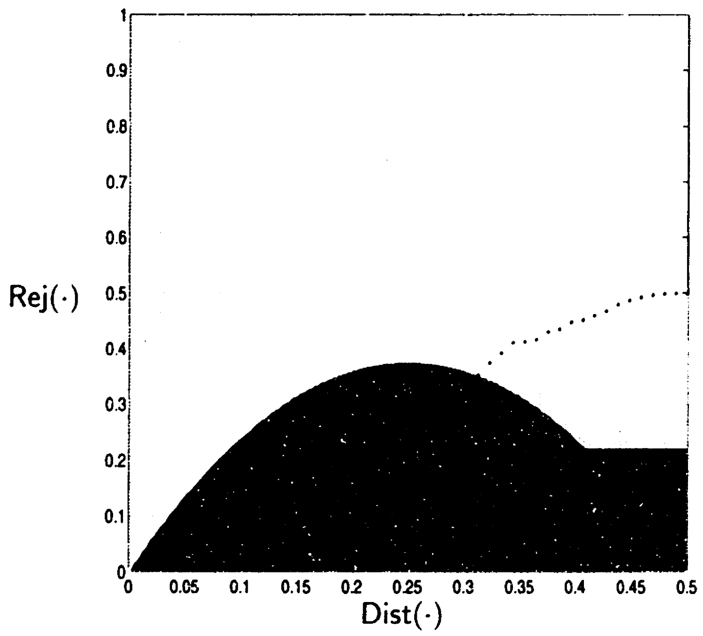

Figure 2-2: Main results of this chapter. (See text for discussion.)

problems, begins in [ALM$^+$92] and continues in [BGLR93, BS94, BGS95]. In the first three cases, it is used to test the Hadamard code; in the last case, to test the long code. In all cases the underlying problem is the one we have considered above, i.e. linearity testing over the two element field.

The MaxSNP hardness result of [BGLR93] used only two things: the lower bound $\Gamma(\delta) \geq 3\delta(1-2\delta)$ of Theorem 2.1.1, and the best available lower bound k on the knee. They expressed the non-approximability factor for Max-3SAT as an increasing function $g_1(k)$ depending solely on k. The lower bound on the knee that they used was $\mathsf{Knee} \geq 1/6$.[3] Their final result was that approximating Max-3SAT within $113/112 \approx 1.009$ is NP-hard.

Improved proof systems were built in [BS94]. Again, their non-approximability factor had the form $g_2(k)$ for some function g_2 depending only on the best available lower bound k on the knee. They also used $\mathsf{Knee} \geq 2/9$ to show that approximating Max-3SAT within $74/73 \approx 1.014$ is NP-hard.

Theorem 2.1.3 would yield direct improvements to the results of [BGLR93, BS94] with

[3] This lower bound is obtained from the first part of Theorem 2.1.1 and the bound $\Gamma(\delta) \geq \delta/2$ (implicit in [BLR90]).

no change in the underlying proof systems or construction. However, better proof systems are now known, namely the ones of [BGS95]. The analysis in the latter uses both our results (Theorem 2.1.2 and Theorem 2.1.3). They showed that approximating Max-3SAT within 1.038 is NP-hard. They also exploit our analyses to derive improved non-approximability results for other MaxSNP problems (like Max-2SAT and Max-Cut) and for Vertex Cover.

LOW-DEGREE TESTING. There are a variety of problems which are studied under the label of low-degree testing. Linearity testing being one of them. Below we briefly explain what the other problems are and how the results concerning them differ from ours.

The problem in low-degree testing is the following: we are given an oracle for a function $f\colon K^n \to K$, where K is a finite field, and we are given a positive integer d. In the low *individual degree* testing problem we are asked to determine whether f is close to some polynomial of degree d in each of its n variables. When specialized to the case of $d = 1$, this task is referred to as *multi-linearity testing.* In the low *total degree* testing problem we are asked to determine whether f is close to some polynomial of total degree d in its n variables.[4] Multi-linearity tests were studied by [BFL90, FGL$^+$91]. Low individual degree tests were studied by [BFLS91, AS92b, FHS94, PS94]. Total degree tests were studied by [GLR$^+$91, ALM$^+$92, RS93, FS95].

What we are looking at, namely linearity testing over GF(2), is a variant of the total degree testing problem in which the degree is $d = 1$, K is set to GF(2), and the constant term of the polynomials are forced to 0.

The above mentioned works have put a significant amount of effort into the analyses of the low degree tests. However, these analyses do not appear to be tight for any case. In particular, one cannot use them to derive the results we obtain here. In fact, the tightness of the results obtained here raises the issue as to whether similar techniques can be used to improve the analyses of the above referred tests.

THE ROLE OF TESTING IN PROOF SYSTEMS. To explain this, recall that proof systems are built by recursion [AS92b]. Each level of recursion will typically use some form of low-degree

[4] To illustrate the difference between individual and total degree, note that $f(x_1, \ldots, x_n) = x_1 x_2$ is multi-linear but of total degree 2.

testing, the kind differing from level to level.

Babai, Fortnow, and Lund [BFL90] initiated the use of multi-linearity testing. For efficiency reasons, researchers beginning with Babai, Fortnow, Levin, and Szegedy [BFLS91] then turned to low individual degree testing. This testing is used in the 'higher' levels of the recursion. Linearity testing showed up for the first time in the lowest level of the recursion, in the checking of the Hadamard code [ALM$^+$92]. The proof systems we discuss use all these different testers, but, as we explained, the final non-approximability factors obtained can be expressed only in terms of the linearity testing curve.

PROGRAM CHECKING. The BLR test first appeared in a different context, namely, that of *self-testing*, *self-correcting* and *self-checking* programs. These programs have been proposed as an alternative approach for assuring software reliability.

Informally, self-checking means that a program purported to compute a function can be used to verify (with high probability) that its output is correct. Self-testing means that a program purported to compute a function can be used to check that it is indeed correct, but in a way that does not involve recomputing the function nor checking the program's code. Self-correction assumes that we have a program that works correctly in a large fraction of its (finite) input space. This same program is then used to find the correct value everywhere (with high probability).

For more comprehensive sources of information on this topic the reader is referred to [Rub90, Sud92, BW94].

2.1.5 Chapter organization

In Section 2.2 we review the basic elements of discrete Fourier analysis. In Section 2.3, we prove the first of the main results of this chapter: Theorem 2.1.2. In Section 2.4 we prove a refinement of the result of [BGLR93] concerning the BLR test.

In Section 2.5 we prove an isoperimetric inequality: the Summation Lemma. We also show that this lemma reveals other facts about the relationship between $\mathsf{Rej}(f)$ and $\mathsf{Dist}(f)$. In particular, it establishes a tight *upper bound* on $\mathsf{Rej}(f)$ as a function of $\delta = \mathsf{Dist}(f)$. We then undertake, in Section 2.6, a combinatorial analysis of the BLR test based on the

Summation Lemma and prove Theorem 2.1.3.

In Section 2.7 we analyze the tightness of the lower bounds we obtain for $\Gamma(\delta)$. We show that the lower bounds are tight on a large part of the range of δ and close to optimal on the rest.

While the main focus of this chapter is the BLR test, we also study, in Section 2.8, two tests that are related to the Hadamard code test. The purpose is to further illustrate the strength and elegance of the Fourier analysis technique, as well as its more general applicability to the problem of analyzing program testers.

2.2 Discrete Fourier transform.

The purpose of this section is to review as well as provide a handy reference for the basic notions concerning discrete Fourier analysis.

We consider the family of all real-valued functions on F^n. This collection of functions is a $|F|^n$-dimensional real vector space with the following inner product:

$$\langle \phi, \theta \rangle \stackrel{\text{def}}{=} \frac{1}{|F|^n} \sum_{u \in F^n} \phi(u)\theta(u).$$

When one studies a linear space of functions defined on an abelian group, choosing a special basis for the linear space might be very useful. This special basis is the *characters*[5] of the group at hand. In our case the group is F^n. Thus, we chose as basis of our linear space the basis $\{ \psi_\alpha \mid \alpha \in F^n \}$, where

$$\psi_\alpha(u) \stackrel{\text{def}}{=} (-1)^{\alpha \cdot u},$$

and $\alpha \cdot u \stackrel{\text{def}}{=} \sum_{i=1}^{n} \alpha_i u_i$. (Throughout this chapter, when elements of the field F appear as exponents they are to be interpreted as real values. Thus, $(-1)^{\alpha \cdot u}$ is well defined.)

It can be easily verified that the family $\{ \psi_\alpha \mid \alpha \in F^n \}$ forms an orthonormal basis.

[5] A character of an abelian group G is any homomorphism from G to the multiplicative group of the complex numbers of absolute value 1.

It follows that any real-valued function ϕ over F^n can be uniquely expressed as a linear combination of the ψ_α's, namely, $\phi = \sum_{\alpha \in F^n} \widehat{\phi}_\alpha \psi_\alpha$. The coefficient $\widehat{\phi}_\alpha$ is referred to as the α-th Fourier coefficient of ϕ. By the orthonormality property of our chosen basis, $\widehat{\phi}_\alpha = \langle \phi, \psi_\alpha \rangle$. The orthonormality of the basis also implies *Parseval's equality*:

$$\langle \phi, \phi \rangle = \sum_{\alpha \in F^n} \left(\widehat{\phi}_\alpha \right)^2 .$$

The *convolution* of two functions ϕ and θ is denoted by $\phi * \theta$ and defined as follows:

$$(\phi * \theta)(x) \stackrel{\text{def}}{=} \frac{1}{|F|^n} \left(\sum_{u,v \,:\, u+v=x} \phi(u)\theta(v) \right) .$$

Note that the vector space of real-valued functions on F^n with the standard addition operator and multiplication operator $*$ is a commutative ring. In particular, the convolution operator is associative, commutative, and distributive with respect to addition.

The following *convolution identity* shows the relationship between the Fourier coefficients of two functions ϕ, θ, and the Fourier coefficients of their convolution:

$$\left(\widehat{\phi * \theta}\right)_\alpha = \widehat{\phi}_\alpha \widehat{\theta}_\alpha .$$

2.3 A lower bound for the linearity test

To lower bound $\mathsf{Rej}(f)$ we use discrete Fourier analysis techniques. In particular, we provide an interpretation of $\mathsf{Dist}(f)$ and $\mathsf{Rej}(f)$ in terms of the Fourier coefficients of an appropriate transformation of f. More precisely, we show that if the distance from f to the nearest linear function is large, then the Fourier coefficients of $(-1)^f$ are small.[6]

Lemma 2.3.1 Let $\delta \in [0,1]$ and $f \colon F^n \to F$. Set $h \equiv (-1)^f$. Then, $\mathsf{Dist}(f) = \mathsf{Dist}(f, 0) = \delta$ if and only if $\widehat{h}_0 = 1 - 2\delta$ and $\widehat{h}_0 \geq \widehat{h}_\alpha$ for all $\alpha \in F^n$.

[6] Throughout this chapter, $h \equiv (-1)^g$, means that h denotes the real valued function defined on the domain of g that sends u to $(-1)^{g(u)}$

Proof: Let $l_\alpha\colon F^n \to F$ denote the linear function that sends u to $\alpha \cdot u$. Note that $(-1)^{f(u)+l_\alpha(u)}$ equals 1 if $f(u) = l_\alpha(u)$, and -1 if $f(u) \neq l_\alpha(u)$. Thus

$$\widehat{h}_\alpha = \frac{1}{|F|^n} \sum_{u \in F^n} (-1)^{f(u)+l_\alpha(u)} = 1 - 2\,\mathsf{Dist}(f, l_\alpha)\,.$$

Hence, $\mathsf{Dist}(f) = \mathsf{Dist}(f,0)$ if and only if $\widehat{h}_\alpha \leq \widehat{h}_0$ for all $\alpha \in F^n$. Moreover, $\mathsf{Dist}(f,0) = \delta$ if and only if $\widehat{h}_0 = 1 - 2\delta$. ∎

Corollary 2.3.2 For every $f\colon F^n \to F$, if $h \equiv (-1)^f$, and $\alpha \in F^n$, then $\widehat{h}_\alpha \leq 1 - 2\,\mathsf{Dist}(f)$.

Proof of Theorem 2.1.2: The key observation is that

$$\mathsf{Rej}(f) = \frac{1}{2}(1 - (h * h * h)(0))\,.$$

Thus, from the definition of Fourier coefficients and the convolution identity, it follows that:

$$\mathsf{Rej}(f) = \frac{1}{2}\left(1 - \sum_{\alpha \in F^n} \left(\widehat{h * h * h}\right)_\alpha\right) = \frac{1}{2}\left(1 - \sum_{\alpha \in F^n} \left(\widehat{h}_\alpha\right)^3\right).$$

The upper bound for $\widehat{h}_\alpha$ given in Corollary 2.3.2 and Parseval's equality imply that

$$\sum_{\alpha \in F^n} \left(\widehat{h}_\alpha\right)^3 \leq (1 - 2\,\mathsf{Dist}(f)) \sum_{\alpha \in F^n} \left(\widehat{h}_\alpha\right)^2 = 1 - 2\,\mathsf{Dist}(f),$$

thus, $\mathsf{Rej}(f) \geq \mathsf{Dist}(f)$. ∎

The reader can verify that we have just shown (implicitly), that the following identity holds:

$$\mathsf{Rej}(f) = \frac{1}{2}\left(1 - \sum_{l \text{ linear}} (1 - 2\,\mathsf{Dist}(f,l))^3\right).$$

This equation captures the intuition that the probability with which a function fails the BLR test depends on how far away that function is from *each* linear function. This intuition is not elicited by the standard arguments that have been used to analyze not only the BLR test, but the low-degree tests as well. We believe that the strength of the lower bounds obtained here are due to the fact that our approach substantiates this intuition.

2.4 A refinement of the BGLR bound

In this section we present a slightly more refined version of the bound $\Gamma(\delta) \geq 3\delta(1-2\delta)$ derived in [BGLR93]. We use this refinement in the following sections.

To state our refinement we need the following definition:

$\mathsf{sl}(f,g) \overset{\text{def}}{=} \Pr_{u,v \in_R F^n}[f(u) \neq g(u), f(v) \neq g(v), f(u+v) \neq g(u+v)]$ — the slack between functions $f,g\colon F^n \to F$.

Lemma 2.4.1 For every $f,l\colon F^n \to F$ where l is linear,

$$\mathsf{Rej}(f) \quad = \quad 3\,\mathsf{Dist}(f,l) - 6\,\mathsf{Dist}(f,l)^2 + 4\,\mathsf{sl}(f,l)\,.$$

Proof: First, observe that $f(u)+f(v) \neq f(u+v)$ if and only if f differs from l in either exactly one of the points $\{u,v,u+v\}$ or in all of the points $\{u,v,u+v\}$. Thus

$$\begin{aligned}\mathsf{Rej}(f) \quad &= \quad 3\Pr_{u,v\in_R F^n}[f(u) \neq l(u), f(v) = l(v), f(u+v) = l(u+v)] \\ &\quad + \Pr_{u,v\in_R F^n}[f(u) \neq l(u), f(v) \neq l(v), f(u+v) \neq l(u+v)]\,.\end{aligned}$$

The second term on the RHS above is $\mathsf{sl}(f,l)$, and the first one equals

$$3\Pr_{u,v\in_R F^n}[f(u) \neq l(u), f(v) = l(v)] - 3\Pr_{u,v\in_R F^n}[f(u) \neq l(u), f(v) \neq l(v)] + 3\,\mathsf{sl}(f,l)\,.$$

Observing that the events $\{(u,v) \mid f(u) = l(u)\}$, and $\{(u,v) \mid f(v) = l(v)\}$ are independent, and performing a simple algebraic manipulation, suffices to conclude the proof of the lemma.

■

Corollary 2.4.2 (Bellare-Goldwasser-Lund-Russell [BGLR93]) $\Gamma(\delta) \geq 3\delta(1-2\delta)$.

The analysis of the linearity test thus far presented is already an improvement upon *the best analysis that can be gleaned from [BLR90] and [BGLR93] (for the case in which* $F = \mathrm{GF}(2)$). But, we can still do better, as the following two sections will show.

2.5 The Summation Lemma

This section is devoted to proving a combinatorial result of independent interest, but necessary in the tighter analysis of the linearity test that we give in Section 2.6. We also apply this result to obtain a tight upper bound on the probability that the BLR test fails.

First, recall that the lexicographic order in F^n is the total order relation $\leq$ such that $u \leq v$ if and only if $\sum_{i=1}^{n} u_i 2^{-i} \leq \sum_{i=1}^{n} v_i 2^{-i}$ (arithmetic over the reals).

We will show that given three subsets A, B, C of F^n, the number of triplets (u, v, w) in $A \times B \times C$ such that $u + v + w = 0$ is maximized when A, B, C are the lexicographically smallest $|A|$, $|B|$, $|C|$ elements of F^n respectively.

Before we state the main result of this section, we introduce some notation: for every nonnegative integer n, and $A, B, C \subseteq F^n$ let

$$\Phi_n(A,B,C) \stackrel{\text{def}}{=} \{(u,v,w) \in A \times B \times C \,|\, u+v+w=0\},$$

and let

$$\phi_n(A,B,C) \stackrel{\text{def}}{=} \frac{1}{|F|^{2n}} \, |\Phi_n(A,B,C)| .$$

Also, for $S \subseteq F^n$ we let S^* denote the smallest, in lexicographic order, $|S|$ elements of F^n.

The following lemma, independently proved by D. J. Kleitman [Kle94], gives a precise statement of the above discussed fact.

Lemma 2.5.1 (Summation Lemma) For every $A, B, C \subseteq F^n$

$$\phi_n(A,B,C) \leq \phi_n(A^*,B^*,C^*).$$

Proof: We proceed by induction. The case $n = 1$ can be easily verified. For the inductive step, consider $i \in \{1, \ldots, n\}$ and $b \in F$. Let $f_{i,b}$ be the function such that at $u = (u_j : j \neq i)$ in F^{n-1} equals u_j if $j \neq i$, and b otherwise, i.e. $f_{i,b}$ is the function that embeds F^{n-1} onto $\{u \in F^n \,|\, u_i = b\}$ in the natural way. For $S \subseteq F^n$, let $S_b^{(i)}$ be the natural projection into F^{n-1} of the elements of S whose i-th coordinate is b, i.e.

$S_b^{(i)} = \{ (u_j : j \neq i) \in F^{n-1} \mid f_{i,b}(u) \in S \}$. Let

$$S^{(i)} = f_{i,0}\left((S_0^{(i)})^*\right) \bigcup f_{i,1}\left((S_1^{(i)})^*\right) .$$

Observe that $|S| = |S_0^{(i)}| + |S_1^{(i)}|$, $|(S_0^{(i)})^*| = |S_0^{(i)}|$, and $|(S_1^{(i)})^*| = |S_1^{(i)}|$. Since $f_{i,0}$ and $f_{i,1}$ are injective and their ranges are disjoint it follows that $|S^{(i)}| = |S|$.[7]

Now, given $A, B, C \subseteq F^n$

$$\begin{aligned}\phi_n(A,B,C) &= \phi_{n-1}(A_0^{(i)}, B_0^{(i)}, C_0^{(i)}) + \phi_{n-1}(A_1^{(i)}, B_1^{(i)}, C_0^{(i)}) \\ &\quad + \phi_{n-1}(A_1^{(i)}, B_0^{(i)}, C_1^{(i)}) + \phi_{n-1}(A_0^{(i)}, B_1^{(i)}, C_1^{(i)}) .\end{aligned}$$

Applying the inductive hypothesis to each term on the RHS above shows that

$$\begin{aligned}\phi_n(A,B,C) &\leq \phi_{n-1}((A_0^{(i)})^*, (B_0^{(i)})^*, (C_0^{(i)})^*) + \phi_{n-1}((A_1^{(i)})^*, (B_1^{(i)})^*, (C_0^{(i)})^*) \\ &\quad + \phi_{n-1}((A_1^{(i)})^*, (B_0^{(i)})^*, (C_1^{(i)})^*) + \phi_{n-1}((A_0^{(i)})^*, (B_1^{(i)})^*, (C_1^{(i)})^*) .\end{aligned}$$

In the previous inequality, the RHS is equal to $\phi_n(A^{(i)}, B^{(i)}, C^{(i)})$. Hence, $\phi_n(A,B,C) \leq \phi_n(A^{(i)}, B^{(i)}, C^{(i)})$. We can assume that for all $i \in \{1, \ldots, n\}$, $A^{(i)} = A$, $B^{(i)} = B$ and $C^{(i)} = C$, since otherwise we can repeat the above argument by considering $A^{(i)}$, $B^{(i)}$, $C^{(i)}$ instead of A, B, C. This iterative process is guaranteed to eventually stop. To explain this, we abuse notation and let $u \in F^n$ represent the integer with binary expansion u. Note now that if $A^{(i)} \neq A$, or $B^{(i)} \neq B$, or $C^{(i)} \neq C$, then $\sum_{u \in S} u > \sum_{u \in S^{(i)}} u$ for some $S \in \{ A, B, C \}$.

One would like to conclude the proof of the lemma by claiming that, if $A^{(i)} = A$, $B^{(i)} = B$, $C^{(i)} = C$ for all i, then A, B, C are equal to A^*, B^*, C^* respectively. The latter claim is 'almost' true, in the sense that, if S is a set such that $S^{(i)} = S$ for all i, then either $S = S^*$, or $S = \{ u \in F^n \mid u_1 = 0 \text{ or } u = 10\cdots0 \} \setminus \{ 01\cdots1 \}$. The lemma follows by case analysis. There are three cases to consider depending on how many of the sets A, B, C are lexicographically ordered.

[7] The following example might help in clarifying the notation so far introduced: if $n = 3$ and $S = \{010, 011, 100, 101, 111\}$, then $S_0^{(2)} = \{10, 11\}$, $S_1^{(2)} = \{00, 01, 11\}$, $(S_0^{(2)})^* = \{00, 01\}$, $(S_1^{(2)})^* = \{00, 01, 10\}$, and $S^{(2)} = \{000, 001, 010, 011, 110\}$.

<u>**Case 1; $A = A^*$, $B = B^*$, $C \neq C^*$:**</u> Then, $C = (V \setminus \{e\}) \cup \{e'\}$, where $V = \{u \in F^n \,|\, u_1 = 0\}$, $e = 10\cdots 0 \in F^n$, and $e' = 01\cdots 1 \in F^n$. Thus,

$$\phi_n(A,B,C) \;=\; \phi_n(A,B,V) + \phi_n(A,B,\{e\}) - \phi_n(A,B,\{e'\}).$$

To conclude, note that $\phi_n(A,B,\{e'\}) = 2\min\{0, |A \cap V| + |B \cap V| - |V|\}$, and $\phi_n(A,B,\{e\}) = \min\{|A \setminus V|, |B \cap V|\} + \min\{|A \cap V|, |B \setminus V|\}$. Hence, $\phi_n(A,B,\{e\}) \leq \phi_n(A,B,\{e'\})$. Moreover, $\phi_n(A,B,V) = \phi_n(A^*,B^*,C^*)$, thus $\phi_n(A,B,C) \leq \phi_n(A^*,B^*,C^*)$.

The other two cases are left to the reader. ■

By definition, for every pair of points u, v in a subspace, the point $u + v$ is also in the subspace. This motivates using $\varphi(S) \stackrel{\text{def}}{=} |\Phi_n(S,S,S)|/|S|^2$, as a measure of how *close* a set $S \subseteq F^n$ is to being a subspace. The set S is a subspace, if and only if, $\varphi(S) = 1$, and an affine space which is not a subspace, if and only if, $\varphi(S) = 0$. The larger the quantity $\varphi(S)$ is, the closer the set S is from being a subspace. From this point of view, the Summation Lemma implies that the collection of the lexicographically smallest m elements of F^n is the subset of F^n (of cardinality m) that most closely resembles a subspace.

Lemma 2.5.2 Suppose $f \colon F^n \to F$ is such that $\delta = \mathsf{Dist}(f)$. Let k be the unique integer such that $2^{-k} \leq \delta < 2^{-k+1}$, and let $\gamma = 2^{-k}$. Then

$$\mathsf{Rej}(f) \;\leq\; 3\delta(1-2\delta) + 4\gamma^2 + 12\,(\delta-\gamma)^2\,.$$

Proof: Let $S = \{u \in F^n \,|\, f(u) \neq l(u)\}$, where l is the closest linear function to f. Note that $|S^*| = |S| = \delta\,|F|^n$. It follows from the definition of $\mathsf{sl}(f,l)$ given in Section 2.4, that $\mathsf{sl}(f,l) = \phi_n(S,S,S)$. Thus, by the Summation Lemma, $\mathsf{sl}(f,l) \leq \phi_n(S^*,S^*,S^*)$. This, coupled with Lemma 2.4.1 shows that

$$\mathsf{Rej}(f) \;=\; 3\delta(1-2\delta) + 4\,\mathsf{sl}(f,l) \;\leq\; 3\delta(1-2\delta) + 4\,\phi_n(S^*,S^*,S^*)\,.$$

Now, let V be the lexicographically smallest $\gamma\,|F|^n$ elements of F^n. Note that V is a subspace. Since $\phi_n(S^*\backslash V, V, V)$, $\phi_n(V, S^*\backslash V, V)$, $\phi_n(V, V, S^*\backslash V)$, and $\phi_n(S^*\backslash V, S^*\backslash V, S^*\backslash V)$

are all equal to 0 we get that

$$\begin{aligned}\phi_n(S^*,S^*,S^*) &= \phi_n(S^*\setminus V, S^*\setminus V, V) + \phi_n(S^*\setminus V, V, S^*\setminus V) \\ &\quad + \phi_n(V, S^*\setminus V, S^*\setminus V) + \phi_n(V,V,V).\end{aligned}$$

Note that $\phi_n(V,V,V) = \gamma^2$. Moreover, $\phi_n(S^*\setminus V, S^*\setminus V, V)$, $\phi_n(S^*\setminus V, V, S^*\setminus V)$, and $\phi_n(V, S^*\setminus V, S^*\setminus V)$ are all equal to $(\delta-\gamma)^2$. Thus, $\phi_n(S^*,S^*,S^*) = \gamma^2 + 3\,(\delta-\gamma)^2$. ∎

Lemma 2.5.2 is the best possible. To prove this consider the function $f\colon F^n \to F$. Let S_f be the set of points in F^n where f differs from its closest linear function. The proof of Lemma 2.5.2 shows that, among all functions that are at distance δ from being linear, Rej(f) achieves its maximum if $\phi_n(S_f,S_f,S_f) = \phi_n(S_f^*,S_f^*,S_f^*)$. We shall now construct, for each $\delta \in [0,1/2]$, a function f such that Dist$(f) = \delta$ and $\phi_n(S_f,S_f,S_f) = \phi_n(S_f^*,S_f^*,S_f^*)$. Indeed, let $S \subseteq F^n$ be the set of the lexicographically smallest $\delta\,|F|^n$ elements of F^n. Let f be the function that evaluates to 1 at every $u \in S$, and to 0 elsewhere. Since $S = S^*$ and $|S| = \delta\,|F|^n$, once we show that $S_f = S$ it will follow that Rej(f) meets the upper bound of Lemma 2.5.2 and Dist$(f) = \delta$. To show that $S_f = S$ it suffices to prove that the zero function is the closest linear function to f. We consider the following two cases:

<u>**Case 1; $\delta \in [0,1/4]$:**</u> Here, the zero function is at distance δ from f. If some other linear function was at distance less than δ from f, then such linear function would be at distance at most $2\,\delta$ from the zero function. A contradiction, since two distinct linear functions are at distance 1/2.

<u>**Case 2; $\delta \in (1/4,1/2]$:**</u> Let V be the largest subspace of F^n contained in S, and let V' be the smallest subspace of F^n that contains S. Note that since S is the set of the lexicographic smallest $\delta|F|^n$ elements of F^n, then $|V| = \frac{1}{4}|F|^n$ and $|V'| = \frac{1}{2}|F|^n$. For the sake of contradiction, assume $l\colon F^n \to F$ is a nonzero linear function whose distance to f is less than δ. We shall heavily rely on the fact that a linear function which is nonzero over a subspace of F^n must evaluate to 1 in exactly half the elements of that subspace. If l always evaluates to 0 over V, then l disagrees with f in every element of V and in at least $\frac{1}{2}|F|^n - |S\setminus V|$ of the elements not in V. Hence, the distance between f and l is at least

$1/4 + 1/2 - (\delta - 1/4) \geq \delta$, a contradiction. On the other hand, if l does not evaluate to 0 over V, then l disagrees with f in half the elements of V and in half the elements of V'. Hence, l disagrees with f in half the elements of V, in at least $|S \setminus V| - \frac{1}{2}|V' \setminus V|$ of the elements of $V' \setminus V$, and in half the elements not in V'. Thus, the distance between f and l is at least $1/8 + (\delta - 1/4) - 1/8 + 1/4 \geq \delta$, again a contradiction.

2.6 Combinatorial analysis of the linearity test

In this section we prove a lower bound on the value of the Knee of the linearity testing curve, namely that Knee $\geq 45/128$. The tightness discussion of Section 2.7 will show that this bound is best possible.

We now informally describe the argument used to prove the lower bound. Given a function $f\colon F^n \to F$ define a function $g_f\colon F^n \to F$, whose value at u is $\textsc{Plurality}\{\, f(u+v) - f(v) \,|\, v \in F^n \,\}$.[8] Then, if Rej$(f)$ is 'sufficiently small' three things occur: an overwhelming majority of the values $\{\, f(u+v) - f(v) \,|\, v \in F^n \,\}$ agree with $g_f(u)$, g_f is linear, and g_f is 'close' to f. It remains to specify what 'sufficiently small' and 'close' stand for. This argument was first employed in [BLR90] to study the linearity test over finite groups. We will show how this argument can be tightened in the case of linearity testing over the two element field.

The proof of this section's main result is a consequence of the following three lemmas which we now state and whose proofs are postponed.

Lemma 2.6.1 For all $f\colon F^n \to F$, Rej$(f) \geq$ Dist$(f, g_f)/2$.

Lemma 2.6.2 For all $f\colon F^n \to F$, if g_f is linear, then Rej$(f) \geq 2\,$Dist$(f, g_f)\ (1 -$ Dist$(f, g_f))$.

Lemma 2.6.3 For all $f\colon F^n \to F$, if Rej$(f) < 45/128$, then g_f is linear.

Hence, if Knee $< 45/128$, there would be a function f whose probability of failing the BLR test would be smaller than $45/128$. The latter of the lemmas above implies that g_f

[8] A multiset's *plurality* is the most commonly occurring its elements (ties are broken arbitrarily).

would be linear. Hence, we can apply the first two lemmas stated above to lower bound the probability that f fails the BLR test. This leads to a contradiction. Formally, we have

Proof of Theorem 2.1.3: Assume Knee $< 45/128$, then, there is a function $f\colon F^n \to F$, such that $\mathsf{Rej}(f) < 45/128$ and $\delta = \mathsf{Dist}(f) \geq 1/4$. By Corollary 2.4.2, $\mathsf{Rej}(f) \geq 3\,\delta\,(1-2\delta)$, thus, we need only consider the case in which δ is at least $5/16$. By Lemma 2.6.3, g_f is a linear function. Thus, $1 \geq \mathsf{Dist}(f, g_f) \geq \delta \geq 5/16$, which together with Lemma 2.6.1 and Lemma 2.6.2 imply that $\mathsf{Rej}(f) \geq \min_{x \in [5/16,1]} \max\{\, x/2, 2\,(1-x)\,x \,\} = 3/8$, a contradiction. Hence, Knee $\geq 45/128$. In Section 2.7 we show that there exists a function $f\colon F^n \to F$ such that $\mathsf{Dist}(f) = 5/16$ and $\mathsf{Rej}(f) = 45/128$. Hence, Knee $= 45/128$. ∎

The rest of this section is dedicated to proving Lemmas 2.6.1 through 2.6.3. The proofs of Lemma 2.6.1 and Lemma 2.6.2 are based on the following fact (which is implicit in [BLR90]): for all $u \in F^n$, since $g_f(u)$ is equal to the most common among the two values that $f(u+v) - f(v)$ takes when v varies over F^n, we get that $\Pr_{v \in_R F^n}[\, f(u+v) - f(v) = g_f(u)\,]$ is at least $1/2$. Hence, if $f(u) \neq g_f(u)$, then $f(u)$ is different from $f(u+v) - f(v)$ at least half of the time, which implies

$$\Pr_{u,v \in_R F^n}[\, f(u) + f(v) \neq f(u+v) \mid f(u) \neq g_f(u)\,] \;\geq\; 1/2\,. \tag{2.1}$$

Proof of Lemma 2.6.1: Simple conditioning says that $\mathsf{Rej}(f)$ is at least

$$\Pr_{u,v \in_R F^n}[\, f(u) + f(v) \neq f(u+v) \mid f(u) \neq g_f(u)\,] \cdot \mathsf{Dist}(f, g_f)\,.$$

But, by (2.1) we know this is at least $\mathsf{Dist}(f, g_f)/2$. ∎

Proof of Lemma 2.6.2: Assume g_f is linear. As observed in the proof of Lemma 2.4.1

$$\begin{aligned} \mathsf{Rej}(f) &= 3\Pr_{u,v \in_R F^n}[\, f(u) \neq g_f(u), f(v) = g_f(v), f(u+v) = g_f(u+v)\,] \\ &\quad + \Pr_{u,v \in_R F^n}[\, f(u) \neq g_f(u), f(v) \neq g_f(v), f(u+v) \neq g_f(u+v)\,] \\ &= 3\Pr_{u,v \in_R F^n}[\, f(u) + f(v) \neq f(u+v) \mid f(u) \neq g_f(u)\,] \cdot \mathsf{Dist}(f, g_f) \\ &\quad - 2\Pr_{u,v \in_R F^n}[\, f(u) \neq g_f(u), f(v) \neq g_f(v), f(u+v) \neq g_f(u+v)\,]\,. \end{aligned}$$

By (2.1), the first term of the latter expression is lower bounded by $3\,\mathsf{Dist}(f,g_f)/2$. The second term is equal to $2\,\mathsf{sl}(f,g_f)$. Thus, $\mathsf{Rej}(f) \geq 3\,\mathsf{Dist}(f,g_f)/2 - 2\,\mathsf{sl}(f,g_f)$. Applying Lemma 2.4.1, we get that $\mathsf{Rej}(f) \geq 3\,\mathsf{Dist}(f,g_f) - 3\,\mathsf{Dist}(f,g_f)^2 - \mathsf{Rej}(f)/2$. The lemma follows. ∎

Proof of Lemma 2.6.3: By contradiction. Assume g_f is not linear. Then there are $x \neq y$ such that $g_f(x)+g_f(y) \neq g_f(x+y)$. Thus, at least one of $g_f(x)$, $g_f(y)$, and $g_f(x+y)$ has to be nonzero. Without loss of generality assume that $g_f(x+y)=1$. Note that, since by construction $g_f(0)=0$, then x and y are nonzero. Thus, $x, y, x+y$ are distinct.

Let $S = \{0, x, y, x+y\}$. For every $s \in F^n$, define f_s to be the function from S to F, such that $f_s(u) = f(s+u)$. Hence, if $p_{s,t} \overset{\text{def}}{=} \Pr_{u,v\in_R S}[\,f_s(u)+f_t(v) \neq f_{s+t}(u+v)\,]$, then

$$\mathsf{Rej}(f) = \mathsf{E}_{s,t\in_R F^n}[p_{s,t}] = \mathsf{E}_{s,t\in_R F^n}\left[\Pr_{u,v\in_R S}[\,f_s(u)+f_t(v) \neq f_{s+t}(u+v)\,]\right]. \quad (2.2)$$

But, $p_{s,t}$ depends only on the values that f_s, f_t, and f_{s+t} take. That is, on the *trace* of f at s, t and $s+t$, where the trace of f at w is defined as $[\,f_w(0), f_w(x), f_w(y), f_w(x+y)\,]$, and denoted by $tr_f(w)$. To lower bound $p_{s,t}$, the following partition of the elements $s \in F^n$, according to the trace of f at s, plays a crucial role:

$$\begin{aligned}
H_0 &= \{\, s \in F^n \mid tr_f(s) \text{ equals } [0,0,0,0] \text{ or } [1,1,1,1] \,\} \\
H_x &= \{\, s \in F^n \mid tr_f(s) \text{ equals } [0,0,1,1] \text{ or } [1,1,0,0] \,\} \\
H_y &= \{\, s \in F^n \mid tr_f(s) \text{ equals } [0,1,0,1] \text{ or } [1,0,1,0] \,\} \\
H_{x+y} &= \{\, s \in F^n \mid tr_f(s) \text{ equals } [0,1,1,0] \text{ or } [1,0,0,1] \,\} \\
H_{odd} &= \{\, s \in F^n \mid tr_f(s) \text{ has an odd number of 1's} \,\}.
\end{aligned}$$

We also partition $F^n \times F^n$ into six sets as follows:

$\mathcal{A}$ — Set of pairs (s,t) such that all of $\{s, t, s+t\}$ are in the same set, either H_0 or H_x or H_y or H_{x+y}

$\mathcal{B}$ — Set of pairs (s,t) such that two of $\{s, t, s+t\}$ are in the same set H_0 or H_x or H_y or H_{x+y}, and the other one is in H_{odd}

$\mathcal{C}$ — Set of pairs (s,t) such that at least two of $\{s,t,s+t\}$ are in H_{odd}

$\mathcal{D}$ — Set of pairs (s,t) such that $\{s,t,s+t\} \subset H_0 \cup H_x \cup H_y \cup H_{x+y}$ with exactly two elements from the same set H_0, H_x, H_y or H_{x+y}

$\mathcal{E}$ — Set of pairs (s,t) such that one of $\{s,t,s+t\}$ is in H_{odd}, the other two are from different sets in H_0, H_x, H_y and H_{x+y}

$\mathcal{F}$ — Set of pairs (s,t) such that $\{s,t,s+t\}$ are from different sets H_0, H_x, H_y, H_{x+y}.

We now proceed to show a lower bound for Rej(f) which depends on the relative size of the sets $\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D},\mathcal{E}$, and $\mathcal{F}$. Indeed, observe that if (s,t) is in $\mathcal{B}$, then $p_{s,t}$ is at least 1/4. If (s,t) is in $\mathcal{C}$, then $p_{s,t}$ is at least 3/8. And if (s,t) is in $\mathcal{D}$, $\mathcal{E}$ or $\mathcal{F}$, then $p_{s,t}$ is equal to 1/2. Hence, if for a set $S \subseteq F^n \times F^n$ we let $\mu(S) = |S|/|F|^{2n}$, then (2.2) yields

$$\text{Rej}(f) \;\geq\; \mu(\mathcal{B})/4 + 3\mu(\mathcal{C})/8 + (\mu(\mathcal{D}) + \mu(\mathcal{E}) + \mu(\mathcal{F}))/2 .$$

Recalling that $\mu(\mathcal{C}) = 1 - \mu(\mathcal{A}) - \mu(\mathcal{B}) - \mu(\mathcal{D}) - \mu(\mathcal{E}) - \mu(\mathcal{F})$, we can conclude that

$$\text{Rej}(f) \;\geq\; 3/8 - (3\mu(\mathcal{A}) + \mu(\mathcal{B}))/8 + (\mu(\mathcal{D}) + \mu(\mathcal{E}) + \mu(\mathcal{F}))/8 . \qquad (2.3)$$

From now on, for simplicity's sake, we denote $|H_0|/|F|^n$, $|H_x|/|F|^n$, $|H_y|/|F|^n$, $|H_{x+y}|/|F|^n$ and $|H_{odd}|/|F|^n$, by h_0, h_x, h_y, h_{x+y} and h_{odd} respectively.

We now derive from (2.3) another lower bound for Rej(f) which will depend solely on $h_0, h_x, h_y, h_{x+y}, h_{odd}$ and $\mu(\mathcal{F})$. But first, we need the following identities relating the measure of the sets $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$ and $\mathcal{F}$, to h_0, h_x, h_y, h_{x+y}, and h_{odd}:

$$3\,\mu(\mathcal{A}) + \mu(\mathcal{B}) + \mu(\mathcal{D}) \;=\; 3\left(h_0^2 + h_x^2 + h_y^2 + h_{x+y}^2\right) \qquad (2.4)$$

$$2\,\mu(\mathcal{D}) + \mu(\mathcal{E}) + 3\,\mu(\mathcal{F}) \;=\; 3\left((1-h_{odd})^2 - \left(h_0^2 + h_x^2 + h_y^2 + h_{x+y}^2\right)\right) . \qquad (2.5)$$

Adding $-1/8$ of (2.4) and 1/8 of (2.5) to (2.3), gives

$$\text{Rej}(f) \;\geq\; \frac{3}{8} + \frac{3}{8}(1-h_{odd})^2 - \frac{3}{4}\left(h_0^2 + h_x^2 + h_y^2 + h_{x+y}^2\right) - \frac{1}{4}\mu(\mathcal{F}) . \qquad (2.6)$$

We now proceed to upper bound $\mu(\mathcal{F})$. We divide the analysis into two cases. But first we assume, without loss of generality, that $h_x \le h_y \le h_{x+y}$. Observe also, that for a randomly chosen s, $f_s(x+y) - f_s(0)$ differs from $g_f(x+y)$ at most half of the time. Moreover, since we have assumed that $g_f(x+y) = 1$, we get that $h_{x+y} \le 1/2$ since

$$1/2 \;\ge\; \Pr_{s \in_R F^n}\left[g_f(x+y) \neq f_s(x+y) - f_s(0)\right] \;=\; h_0 + h_{x+y} + h_{odd}/2\,. \tag{2.7}$$

<u>**Case 1;** $h_x + h_y - h_0 - h_{x+y} > 1/4$ **:**</u> By assumptions $h_x \ge h_x + h_y - h_0 - h_{x+y} > 1/4$. So $h_x, h_y, h_{x+y} \in (1/4, 1/2]$. Now, as in Section 2.5, let

$$\phi_n(A,B,C) \;=\; \frac{1}{|F|^{2n}}\left|\{\,(u,v,w) \in A \times B \times C \,|\, u+v+w = 0\,\}\right|.$$

Observe that for each element (u,v) of $\mathcal{F}$, $\{\,u, v, u+v\,\}$ either contains an element from H_0 or contains one element from each of the sets H_x, H_y, H_{x+y}.

The contribution to $\mathcal{F}$ of the elements (u,v), where $\{\,u, v, u+v\,\}$ contains elements from each of the sets H_x, H_y, H_{x+y}, is upper bounded by $6\,\phi_n(H_x, H_y, H_{x+y})$. Since $h_x, h_y, h_{x+y} \in (1/4, 1/2]$, the Summation Lemma implies that

$$\begin{aligned}
6\,\phi_n(H_x, H_y, H_{x+y}) &\le 6\,\phi_n(H_x^*, H_y^*, H_{x+y}^*) \\
&= 6\left(\frac{1}{4} - \frac{h_x + h_y + h_{x+y}}{2} + h_x h_y + h_x h_{x+y} + h_y h_{x+y}\right) \\
&= \frac{3}{2} - 3\,(h_0 + h_{odd})\,(h_x + h_y + h_{x+y}) - 3\left(h_x^2 + h_y^2 + h_{x+y}^2\right).
\end{aligned}$$

Furthermore, the contribution to $\mathcal{F}$ of the elements (u,v), where $\{\,u, v, u+v\,\}$ contains an element of H_0 is upper bounded by

$$3\,\phi_n(H_0, H_x, H_y \cup H_{x+y}) + 3\,\phi_n(H_0, H_y, H_x \cup H_{x+y}) + 3\,\phi_n(H_0, H_{x+y}, H_x \cup H_y)\,,$$

which is at most $3\,h_0\,(h_x + h_y + h_{x+y})$. Putting it all together, we have

$$\mu(\mathcal{F}) \;\le\; 3/2 - 3\,h_{odd}\,(h_x + h_y + h_{x+y}) - 3\left(h_x^2 + h_y^2 + h_{x+y}^2\right).$$

This latter inequality, together with (2.6), implies that

$$\begin{aligned}
\mathsf{Rej}(f) &\geq \frac{3}{8}+\frac{3}{8}(1-h_{odd})^2-\frac{3}{4}\left(h_0^2+h_x^2+h_y^2+h_{x+y}^2\right) \\
&\quad -\frac{3}{8}+\frac{3}{4}h_{odd}\,(h_x+h_y+h_{x+y})+\frac{3}{4}\left(h_x^2+h_y^2+h_{x+y}^2\right) \\
&= \frac{3}{8}-\frac{3}{8}h_{odd}^2-\frac{3}{4}h_0h_{odd}-\frac{3}{4}h_0^2 \\
&\geq \frac{3}{8}-\frac{3}{8}\left(h_{odd}+4\,h_0\right)^2.
\end{aligned}$$

We conclude the analysis of this case by noting that by case assumption $1/4 \geq 1 - 3\,(h_x+h_y-h_0-h_{x+y})$, and since $h_x \leq h_y \leq h_{x+y}$ then $1-3\,(h_x+h_y-h_0-h_{x+y}) \geq h_{odd}+4\,h_0$.

<u>Case 2; $h_x+h_y-h_0-h_{x+y} \leq 1/4$:</u> To each element (u,v) in $\mathcal{F}$, associate the unique tuple (u',v') in $\{\,u,v,u+v\,\}\times\{\,u,v,u+v\,\}$, such that $(u',v') \in H_0\times H_{x+y}\cup H_x\times H_y$. This scheme associates to each element of $H_0\times H_{x+y}\cup H_x\times H_y$ at most 6 elements of $\mathcal{F}$. Thus, $\mu(\mathcal{F}) \leq 6\,(h_0h_{x+y}+h_xh_y)$. Which jointly with (2.6) implies

$$\begin{aligned}
\mathsf{Rej}(f) &\geq \frac{3}{8}+\frac{3}{8}(1-h_{odd})^2-\frac{3}{2}\,(h_0h_{x+y}+h_xh_y)-\frac{3}{4}\left(h_0^2+h_x^2+h_y^2+h_{x+y}^2\right) \\
&= \frac{3}{8}-\frac{3}{8}\,(h_x+h_y-h_0-h_{x+y})^2\,.
\end{aligned}$$

The analysis of this case concludes by observing that by case assumption $1/4 \geq h_x+h_y-h_0-h_{x+y}$, and that (2.7) implies that $h_x+h_y-h_0-h_{x+y} = 1-h_{odd}-2(h_0+h_{x+y}) \geq 0$.

■

2.7 Tightness discussion

Here, we discuss how tight the results achieved in the preceding sections are. For the rest of this section let $\delta \in [\,0,1\,]$ be such that $\delta\,|F|^n$ is an integer.

<u>Case 1; $\delta > 1/2$:</u> There is no function $f\colon F^n \longrightarrow F$ such that $\mathsf{Dist}(f) = \delta$ (since the expected distance from a randomly chosen linear function to f is at most $\frac{1}{2}\,(1+1/|F|^n)$).

Case 2; $\delta = 1/2$: Choose $f\colon F^n \to F$ such that $f(u) = X_u$, where X_u is a random variable distributed according to a Bernoulli distribution with parameter $p \in [1/2, 1]$.[9] A Chernoff bound (see [AS92a, Appendix A]) shows that with overwhelming probability, $0 \le \delta - \mathsf{Dist}(f) = o(1)$. Moreover, Chebyschev's inequality (see [AS92a, Ch. 4]) implies that with high probability $\left|\,\mathsf{Rej}(f) - (3p(1-p)^2 + p^3)\,\right| = o(1)$. Thus, if $p = 1/2$, Theorem 2.1.2 is almost tight in the sense that $\mathsf{Rej}(f)$ is almost $1/2$.

Case 3; $5/16 < \delta < 1/2$: The lower bounds of Section 2.3 and Section 2.6 show that in this case, if $\mathsf{Dist}(f) = \delta$, then $\mathsf{Rej}(f) \ge \max\{\,45/128, \delta\,\}$. In Appendix 4.6.3 we give evidence that this lower bound is close to optimal when δ is in the range in point. To achieve this we give a randomized procedure that with high probability outputs a function $f\colon F^n \to F$ such that $\mathsf{Dist}(f)$ is approximately δ and $\mathsf{Rej}(f)$ is small.

Rather than searching for a function f, such that $\mathsf{Dist}(f) = \delta$, which minimizes $\mathsf{Rej}(\cdot)$, we reduce the problem to the one of solving a non-linear, high-dimensional, constrained, minimization problem. It is worth noting that this latter optimization is performed over a continuous space. We hope that this relaxation technique proves useful in the determination of the sharpness of the analyses of other tests, and as a tool for assessing the best bounds one could hope for in these other tests.

Case 4; $0 \le \delta \le 5/16$: Corollary 2.4.2 is tight. Indeed, for $u \in F^n$ let $\lfloor u \rfloor_k \overset{\text{def}}{=} u_1 \cdots u_k$. If $S = \{\, u \in F^n \mid \lfloor u \rfloor_4 \in \{\,1000, 0100, 0010, 0001, 1111\,\}\,\}$, then any boolean function f which equals 1 on $\delta\,|F|^n$ elements of S, and 0 elsewhere, satisfies $\mathsf{sl}(f, 0) = 0$. Hence, by Lemma 2.4.1, $\mathsf{Rej}(f) = 3\,\delta\,(1 - 2\delta)$.

Finally, it is interesting to point out that even for small values of n, our bounds are already fairly tight. Figure 2-3 shows all the points of the form $(\,\mathsf{Dist}(f), \mathsf{Rej}(f)\,)$ when f ranges over all boolean functions on 5 variables (compare with Figure 2-2, page 21).

[9] A Bernoulli distribution with parameter p corresponds to the distribution of a $\{\,0, 1\,\}$-random variable with expectation p.

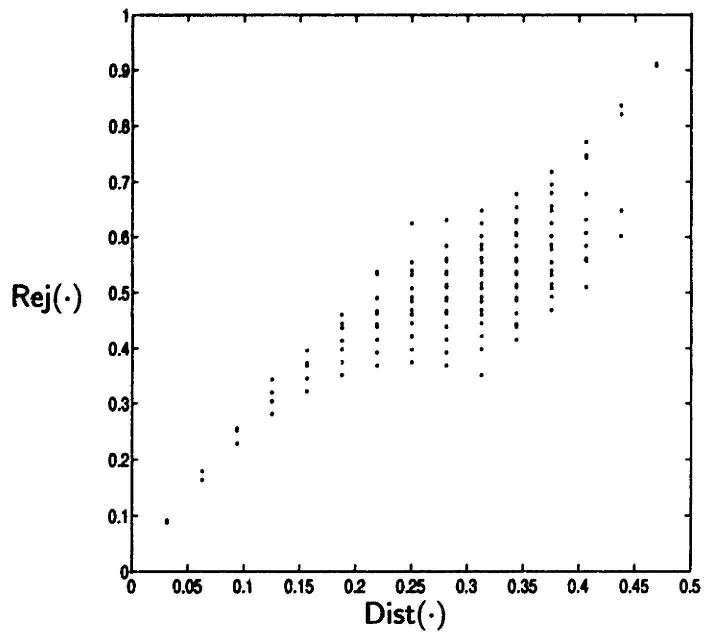

Figure 2-3: The set of points $\{ (\mathsf{Dist}(f), \mathsf{Rej}(f)) \mid f\colon F^5 \to F \}$.

2.8 Extensions

The purpose of this section is to convey that the analysis of the BLR test by means of the discrete Fourier transform has a wider applicability. We illustrate this fact by showing two elegant extensions of the arguments of Section 2.3.

We start by describing the relation between linearity testing and testing Hadamard codes. This relation was first exploited in the PCP context in [ALM+92].

A *Hadamard code* is a collection of $N \overset{\text{def}}{=} |F|^n$ tuples of length $N-1$ of the form $(\, l(u) : u \in F^n \setminus \{0\}\,)$, where $l\colon F^n \to F$ is linear.[10]

We will abuse conventions and say that a Hadamard code, denoted $\mathcal{H}_n$, is the collection of tuples $(\, l(u) : u \in F^n\,)$, where $l\colon F^n \to F$ is linear (e.g., $\mathcal{H}_2 = \{ 0000, 0101, 0011, 0110 \}$).

We refer to the elements of a code as *codewords*. The codes we consider in this section have codewords whose components are indexed by a subset of the elements of F^n. We call this set of indices the *support* of the code (e.g., the support of $\mathcal{H}_n$ is F^n).

[10] For more information on Hadamard codes see [MS77, Ch. 2.§3].

The problem of testing Hadamard codes is a rephrasing of the linearity testing problem. More precisely, assume we are given oracle access to an element of F^N, say $f = (f(u) : u \in F^n)$. It is claimed that f belongs to $\mathcal{H}_n$. We do not trust this claim and we want to verify it. We are charged for each component of f that we access. Note that $f \in \mathcal{H}_n$ if and only if $f(u) + f(v) = f(u+v)$ for all $u, v \in F^n$. Thus, the problem of checking the claim regarding the oracle can be addressed in the same manner as the linearity testing problem. Recall that this is what we have been doing so far. We now consider related problems.

2.8.1 Punctured Hadamard code test

A customary way of constructing new codes from old ones is to *puncture* them. Puncturing a code is the process of deleting one or more coordinates from each codeword. Each time a coordinate is deleted the length of the code drops by one.

For $S \subseteq F^n$, we consider the punctured Hadamard code, denoted $\mathcal{H}_n^S$, whose codewords are of the form $(l(u) : u \in S)$, where l: $F^n \rightarrow F$ is linear. Clearly, the support of $\mathcal{H}_n^S$ is S.

THE PUNCTURED HADAMARD CODE TEST. We are given oracle access to an element of $F^{|S|}$, say $f = (f(u) : u \in S)$. We want to check that f is close (in Hamming distance) to a codeword in $\mathcal{H}_n^S$. We perform the following test — randomly pick $u, v \in S$ such that $u + v \in S$, query the oracle to obtain $f(u), f(v), f(u+v)$. Then, reject if $f(u) + f(v) \neq f(u+v)$, and accept otherwise.

The probability that this test rejects when given oracle access to f is denoted $\mathrm{Rej}_{\mathcal{H}_n^S}(f)$. The relative distance between f and its closest codeword in $\mathcal{H}_n^S$ is denoted $\mathrm{Dist}_{\mathcal{H}_n^S}(f)$. Note that for every f a randomly chosen linear function agrees with f in at least a $(1 - 1/|S|)/2$ fraction of the elements in S. Thus, $\mathrm{Dist}_{\mathcal{H}_n^S}(f) \leq (1 + 1/|S|)/2$.[11]

The issue in testing punctured Hadamard codes is to derive good lower bounds on $\mathrm{Rej}_{\mathcal{H}_n^S}(f)$ as a function of $\mathrm{Dist}_{\mathcal{H}_n^S}(f)$. We will now study this test through discrete Fourier analysis techniques.

[11] If the additive zero of F^n is not in S, then the same argument shows that $\mathrm{Dist}_{\mathcal{H}_n^S}(f) \leq 1/2$.

Proposition 2.8.1 Let $S \subseteq F^n$ and $\varphi(S) = \frac{1}{|S|^2} \left| \{ (u, v, w) \in S \times S \times S \mid u + v + w = 0 \} \right|$. If $\varphi(S) \neq 0$ and $f \colon F^n \to F$, then

$$\mathrm{Rej}_{\mathcal{H}_n^S}(f) \geq \frac{1}{2} - \frac{1}{\varphi(S)} \left(\frac{1}{2} - \mathrm{Dist}_{\mathcal{H}_n^S}(f) \right) = \frac{1}{\varphi(S)} \cdot \mathrm{Dist}_{\mathcal{H}_n^S}(f) - \frac{1}{2} \left(\frac{1}{\varphi(S)} - 1 \right) .$$

Proof: [Sketch] Let g be the indicator of the set S, i.e. $g(u) = 1$ if $u \in S$, and 0 otherwise. Furthermore, let h be the function that sends $u \in F^n$ to $h(u) = g(u) \cdot (-1)^{f(u)}$.

An argumentation akin to the one given in the proof of Theorem 2.1.2 yields that

$$\mathrm{Rej}_{\mathcal{H}_n^S}(f) = \frac{1}{2} \left(1 - \frac{(h * h * h)(0)}{(g * g * g)(0)} \right) = \frac{1}{2} \left(1 - \frac{1}{\varphi(S)\, (\widehat{g}_0)^2} \sum_{\alpha \in F^n} \left(\widehat{h}_\alpha \right)^3 \right) .$$

Moreover, an argument similar to the one of the proof of Corollary 2.3.2 shows that for all $\alpha \in F^n$ it holds that $\widehat{h}_\alpha \leq \widehat{g}_0 \left(1 - 2\,\mathrm{Dist}_{\mathcal{H}_n^S}(f) \right)$. The claim follows by combining the previously stated facts and recalling that by Parseval's $\sum_{\alpha \in F^n} \left(\widehat{h}_\alpha \right)^2 = \widehat{g}_0$. ■

In the case that S is a subspace of F^n, $\varphi(S) = 1$, and Proposition 2.8.1 reduces to Theorem 2.1.2. Moreover, recall that in Section 2.5 we had already encountered the expression $\varphi(S)$. In fact, we argued that the larger $\varphi(S)$ is, the closer the set S is of being a subspace. Thus, the closer S is to a subspace, the better the lower bound of Proposition 2.8.1. This validates the intuition that the more the number of constraints of the form $(u, v, u + v) \in S \times S \times S$, the better the punctured Hadamard test should perform.

2.8.2 Augmented Hadamard code test (or total degree one test)

A customary way of augmenting the Hadamard code $\mathcal{H}_n$ is to consider the code $\mathcal{H}'_n$ consisting of $\mathcal{H}_n$ together with the complement of all its codewords.[12] Equivalently, $\mathcal{H}'_n$ is the collection of codewords of the form $(p(u) : u \in F^n)$, where $p \colon F^n \to F$ is a total degree one polynomial. Note that a total degree one polynomial p is either a linear function or a linear function plus a constant. Thus, since F is of characteristic two, $p(u) + p(v) + p(w) = p(u + v + w)$ for all $u, v, w \in F^n$. The latter is satisfied only if p is of total degree one. In analogy to the case of testing Hadamard codes and punctured Hadamard codes, we are interested in

[12] The complement of $(u_1, \ldots, u_N) \in F^N$ is $(1 - u_1, \ldots, 1 - u_N)$.

obtaining good lower bounds on

$$\mathsf{Rej}_{\mathcal{H}'_n}(f) \stackrel{\text{def}}{=} \Pr_{u,v,w\in_R F^n}\left[\, f(u)+f(v)+f(w) \neq f(u+v+w) \,\right]$$

as a function of the relative distance between f and its closest codeword in $\mathcal{H}'_n$ (henceforth denoted $\mathsf{Dist}_{\mathcal{H}'_n}(f)$).

Proposition 2.8.2 For every $f\colon F^n \to F$, if $\mathsf{Dist}_{\mathcal{H}'_n}(f) = \delta$, then

$$\mathsf{Rej}_{\mathcal{H}'_n}(f) \;\geq\; \max\{\, 8\delta(1-\delta)(1/2-\delta),\, 2\delta(1-\delta)\,\}.$$

Proof: [Sketch] Let $h \equiv (-1)^f$, and observe that

$$\mathsf{Rej}_{\mathcal{H}'_n}(f) \;=\; \frac{1}{2}(1-(h*h*h*h)(0)) \;=\; \frac{1}{2}\left(1-\sum_{\alpha\in F^n}\left(\widehat{h}_\alpha\right)^4\right).$$

Moreover, following the proof of Lemma 2.3.1 we get that $\widehat{h}_\alpha = 1 - 2\,\mathsf{Dist}_{\mathcal{H}'_n}(f, l_\alpha)$, where as usual l_α denotes the linear function that sends $u \in F^n$ to $\alpha\cdot u \in F$. Since both l_α and $l_\alpha + 1$ are polynomials of total degree one, it follows that $1-\delta \geq 1 - \mathsf{Dist}_{\mathcal{H}'_n}(f, l_\alpha + 1) = \mathsf{Dist}_{\mathcal{H}'_n}(f, l_\alpha) \geq \delta$. Hence,

$$\mathsf{Rej}_{\mathcal{H}'_n}(f) \geq \frac{1}{2}\left(1-(1-2\delta)^2\sum_{\alpha\in F^n}\left(\widehat{h}_\alpha\right)^2\right) \;=\; 2\delta(1-\delta).$$

To conclude, note that $f(u)+f(v)+f(w)$ and $f(u+v+w)$ are distinct if and only if f differs from every total degree one polynomial p in exactly one or three of the points $\{u, v, w, u+v+w\}$. This observation leads to a generalization of Lemma 2.4.1, which implies that $\mathsf{Rej}_{\mathcal{H}'_n}(f) \geq 8\delta(1-\delta)(1/2-\delta)$. ∎

CHAPTER 3

Testing and the theory of weight distributions of codes

Recall that probabilistically checkable proofs (PCPs) are built by recursion. Each level of the recursion uses a distinct encoding. Correct encodings are viewed as representations of functions that satisfy a pre-specified property. Thus, a problem in the construction of PCPs is to probabilistically check (test) function properties with as few queries as possible.

In this chapter we address a particular problem that arises in testing. To describe it, assume we are given oracle access to a function which is claimed to satisfy a particular property. We do not trust this claim and we want to verify it. But, we want to avoid querying too many values of the oracle function. So we implement a probabilistic procedure that makes a small number of queries. Based on the answers, we either accept or reject the claim about the oracle. Those oracles that do not satisfy the claim should be rejected with a nonzero probability. We focus on tests where the following holds: if the oracle is rejected with a small probability, then by modifying it a little we obtain an oracle that satisfies the claim. But, we would like to have, for each rejection probability a nontrivial conclusion regarding the degree in which the oracle satisfies the claim. The particular question that we address here is the following: *what can be said about the oracle function when the probabilistic procedure rejects with a very large probability?*

Let us begin by describing this problem more precisely.

3.1 Testing: the high rejection probability problem

3.1.1 The problem

A *test* is a triple of the form $(\mathcal{F}, \mathcal{T}, \mathcal{D})$ where $\mathcal{F}$ is the collection of functions mapping G to H, $\mathcal{T}$ is a collection of functionals mapping $\mathcal{F}$ to $\{accept, reject\}$, and $\mathcal{D}$ is a probability distribution over $\mathcal{T}$. We write $\mathcal{D}$ instead of $(\mathcal{F}, \mathcal{T}, \mathcal{D})$ whenever $\mathcal{F}$ and $\mathcal{T}$ are clear from context. We let $\mathcal{P}$ denote the collection of functions $f \in \mathcal{F}$ for which $T(f)$ equals *accept* for every T which is assigned a positive probability by the distribution $\mathcal{D}$. We assume from now on that $\mathcal{P}$ is nonempty.

One of the issues we are concerned in testing is in understanding the relation between the following two quantities when $f \in \mathcal{F}$:

$\mathsf{Rej}(f) \stackrel{\text{def}}{=} \Pr_{T \sim \mathcal{D}}[T(f) = reject]$ — the probability that $\mathcal{D}$ rejects f,

$\mathsf{Dist}(f) \stackrel{\text{def}}{=} \min\{\Pr_{u \in_R G}[f(u) \neq g(u)] \mid g \in \mathcal{P}\}$ — minimum (relative) distance of f to its closest function in $\mathcal{P}$.

There is an undercurrent in the above formulated testing problem. That is, we are given oracle access to a function $f \in \mathcal{F}$ which is claimed to belong to $\mathcal{P}$. We do not trust this claim. We verify it, probabilistically, by performing the test $\mathcal{D}$ as follows: we choose according to $\mathcal{D}$ a functional T in $\mathcal{T}$; we accept the claim if $T(f) = accept$, and reject it otherwise.

As a first approximation the following notion captures how good a test is: a test is *ξ-strong* if for every $f \in \mathcal{F}$ it holds that $\mathsf{Rej}(f) \geq \xi \cdot \mathsf{Dist}(f) - o(1)$ (where asymptotics are relative to the size of G, H, and the support of $\mathcal{D}$). Clearly, larger values of ξ are preferable. An upper bound on ξ is the ratio between the expected value of $\mathsf{Rej}(f)$ and the expected value of $\mathsf{Dist}(f)$ when f is chosen at random in $\mathcal{F}$. We would like ξ to be this value. Such tests will be henceforth referred to as *optimally strong tests.*[1] Optimally strong tests have an appealing property: if a function f has a rejection probability significantly smaller than

[1] E.g, the BLR test over arbitrary finite groups is $\frac{2}{9}$-strong ([BLR90]), but it is not an optimally strong test. Over the field of two elements it is a 1-strong test, and also an optimally strong test (Theorem 2.1.2).

that of a randomly chosen function in $\mathcal{F}$, then, f has to agree with a function in $\mathcal{P}$ in a significantly larger fraction of values than a randomly chosen function in $\mathcal{F}$.

Most of the tests that fall under the label of low-degree tests have not been proven to be optimally strong. The next section discusses why it is desirable to formulate and explore techniques that can achieve this.

3.1.2 Relationship to other work

The issues we want to address are motivated by the problem of verifying the validity of an assertion with the help of two untrustworthy provers. To explain this point, we view a proof as defining a function $\pi: K^n \to \{0,1\}$, where n is some positive integer and K is a finite field. Let F be a finite field extension of K. Note that π is a polynomial of total degree $d \stackrel{\text{def}}{=} n|K|-1$. Thus, there is a unique polynomial $p: F^n \to F$ of total degree d that agrees with π in all of π's domain [ALM+92]. We think of p as being a redundant extension of the proof represented by π. The segmentation technique of [ALM+92] 'splits' the extension proof p among two provers. Indeed, the second prover's question is a line L of F^n.[2] The first prover's question is a point x belonging to L. The first prover is asked for the value of p at x. He responds according to a function $f: F^n \to F$ purported to be p. The second prover is asked for the univariate polynomial which represents the restriction of p to the line L. He responds with the total degree d polynomial $p_{f,L}: L \subseteq F^n \to F$ which most agrees with the restriction of f to L.

To probabilistically verify that the provers responses are indeed explained by an extension proof p we perform the following $(\mathcal{F}, \mathcal{T}, \mathcal{D})$ test:

> Low total degree test [ALM+92]. Here, $\mathcal{F}$ is the set of functions from F^n to F. $\mathcal{D}$ is the uniform distribution over $\mathcal{T}$. $\mathcal{T}$ is the collection of functionals $T_{x,L}$ where L is a line of F^n and x is a point in L. Moreover, $T_{x,L}(f) = \textit{accept}$, if and only if, f and $p_{f,L}$ agree at x.

The only functions that this test always accepts are the polynomials of total degree d.

[2] A line in F^n is a collection of points of the form $\{u + tv \in F^n \mid t \in F\}$ where $u, v \in F^n$ and $v \neq \mathbf{0}$.

The best known analysis of this test is due to Friedl and Sudan [FS95] who show that it is $\frac{1}{8}$-strong. Better analyses of the low total degree test are desirable since they translate, via [FS95], into improved constructions of codes. Ones that admit very simple randomized error detection. More precisely, into improved constructions of locally testable codes [FS95]. But, the motivation for much of this chapter's discussion comes from the following open problem [Aro94a]: *does the low total degree test work even for high error rates?* A more specific question is: *is there a $\xi > 1/2$ for which the low total degree test is ξ-strong?*

Showing that a generalized version[3] of the low total degree test is an optimally strong test would have complexity theoretic consequences. Indeed, as pointed out by Arora [Aro94b] and Lund [Lun94], it would yield better multi-prover interactive proof systems for NP languages. More precisely, multi-prover proof systems that require $O(\log(n))$ randomness, $O(1)$ provers, $O(\log(n))$ answer size, and that achieve error probability $O(1/\text{polylog}(n))$.

3.1.3 Previous work

The paradigms used to study the low-degree tests can be classified as follows:

ALGEBRAIC ARGUMENTS. E.g., Arora and Safra [AS92b], Arora [Aro95a], and Polishchuk and Spielman [PS94] (all of which analyze the low individual degree test).

PLURALITY FUNCTION ARGUMENT [BLR90, Cop]. We illustrate this argument by showing how it is used in the analysis of the BLR test. Let f be a function from one finite group G into another finite group. Define a function g whose value at u is the most common value of $f(u+v) - f(v)$ when v varies in G, i.e. $\text{PLURALITY}\{\, f(u+v) - f(v) \,|\, v \in G \,\}$. Then, show that if $\mathsf{Rej}(f)$ is sufficiently small, three things happen: an overwhelming majority of the values $\{\, f(u+v) - f(v) \,|\, v \in G \,\}$ agree with $g(u)$, g is linear, and g is close to f. Thus, $\mathsf{Rej}(f)$ cannot be very small if f is far away from linearity. The small rejection probability assumption seems to be an essential component of this argument. Thus, it has failed to

[3] LOW TOTAL DEGREE TEST (GENERALIZED VERSION): Let $\mathcal{K}$ be the collection of curves that pass through $k+2$ points of F^n, k of which are fixed. In this case, $\mathcal{F}$ is the set of functions from F^n to F, $\mathcal{D}$ is the uniform distribution over $\mathcal{T}$, and $\mathcal{T}$ is the collection of functionals $T_{x,C}$, where C is an element of $\mathcal{K}$ and x is a point in C. Moreover, $T_{x,C}(f) = \mathit{accept}$, if and only if, f agrees at x with the degree $(k+1)d$ univariate polynomial $p_{f,C}\colon C \subseteq F^n \to F$ that most agrees with the restriction of f to C.

prove that any of the low-degree tests is an optimally strong test.

The plurality function argument was used in Section 2.6. Moreover, it has been used in many works, e.g., Gemmell, Lipton, Rubinfeld, Sudan, and Wigderson [GLR$^+$91] (for testing polynomials over finite fields), Rubinfeld and Sudan [RS92] (for testing polynomials over rational domains), Arora, Lund, Motwani, Sudan, and Szegedy [ALM$^+$92] (for the low total degree test), and Friedl and Sudan [FS95] (for the low total degree test).

It is interesting to note that, since the mid 50s, coding theorist have used a technique called *majority logic decoding*. This technique is used to correct and detect errors in the transmission, over a noisy channel, of majority-logic decodable codes (see [vL92, Ch. 3.§4] and [PW72, Ch. 10]). The plurality function argument takes a function and uses it to 'correct' its errors. This yields a function with some specific property of interest (e.g., linearity). In essence, majority logic decoding is used for the same purposes.

AD HOC ARGUMENTS. E.g., Babai, Fortnow, and Lund [BFL90] (for the multilinearity test), Feige, Goldwasser, Lovász, Safra, and Szegedy [FGL$^+$91] (for the multilinearity test), Shen [She91] (for the multilinearity test), and Sudan [Sud92] (for the basic univariate test).

DISCRETE FOURIER ANALYSIS ARGUMENT. This argument was illustrated in Chapter 2. Clearly, it is not an instance of the plurality function argument (although it may be considered an application of algebraic techniques). Its use does not require any assumption on the magnitude of the rejection probability of the test. But, more importantly it exhibits an appealing feature that distinguishes it from almost all previous results concerning low-degree tests: it shows that a particular test (the BLR test restricted to the field of two elements) is an optimally strong test in the sense described in Section 3.1.1. In this chapter we show that the discrete Fourier analysis argument is an instance of a more general argument that can be used to study tests. This latter argument is based in coding theoretic techniques.

3.1.4 Main results

The crux of this chapter is the framework which we propose in order to formulate and carry out the analyses of tests. This framework establishes a connection between testing and the theory of dual codes. We illustrate this connection by formulating a new way of testing for linearity over finite fields, namely the *extended linearity test.* We borrow classical tools from the theory of weight distributions of dual codes and prove that the extended linearity test is an optimally strong test. We then discuss why the arguments presented here seem to be especially well suited for proving some of the low-degree tests are indeed optimally strong. Achieving this remains an open problem.

3.1.5 Chapter organization

In Section 3.2 we review some basic notions from coding theory. In Section 3.3 we define the extended linearity test. In Section 3.4 we illustrate the connection between testing and dual codes. In Section 3.5 and Section 3.6 we present the theory of weight distributions of dual codes and the MacWilliams theorem. In Section 3.7, among other things, we fully analyze the extended linearity test, and discuss some of the strengths and weaknesses of the method proposed in this chapter for studying tests.

3.2 Coding theory: the basics

For the sake of brevity the following exposition will be terse. For an in-depth discussion of issues in coding theory, the reader is referred to [MS77, PW72, vL92].

A *code* of *block length* N over the alphabet F is a subset C of F^N. We will only consider codes whose alphabet F is a finite field. The elements $x = (x_1, \ldots, x_N) \in C$ are called *codewords*, and their *length* is said to be N.

If x and y are codewords in C, then the *Hamming distance*, $\mathrm{d}(x,y)$, between x and y is equal to the number of components where x and y are distinct. Formally, $\mathrm{d}(x,y) \overset{\mathrm{def}}{=} |\{\, i \mid x_i \neq y_i \,\}|$. The *Hamming weight* of codeword x is the number of nonzero components of x and is denoted $\mathrm{wt}(x)$.

Definition 3.2.1 The minimum distance of a code C is $\min\{\, \mathrm{d}(x,y) \mid x \in C, y \in C, x \neq y \,\}$. The minimum weight of a code C is $\min\{\, \mathrm{wt}(x) \mid x \in C, x \neq 0 \,\}$.[4]

[4] We adopt the unusual convention that the empty code has minimum distance and minimum weight 0.

Definition 3.2.2 (Weight distribution) For a code C of block length N we let $A_i(C)$ be the number of codewords in C of weight i, and say that $(A_0(C), \ldots, A_N(C))$ is its weight distribution.

Definition 3.2.3 (Linear codes) A code C of block length N is linear if it is a subspace of F^N. If C has dimension k then C is called an $[N, k]$ code.

For a linear code, the minimum distance is equal to the minimum weight.

Definition 3.2.4 (Dual code) If C is a code of block length N then its dual code $C^{\perp}$ is the collection of y's of length N such that $\langle x, y \rangle \overset{\text{def}}{=} \sum_{i=1}^{N} x_i \cdot y_i = 0$ for all $x \in C$.

Clearly, if C is an $[N, k]$ code, then $C^{\perp}$ is an $[N, N-k]$ code.

Example: Consider the family of Hadamard codes defined in Section 2.8. In particular, $\mathcal{H}_2 = \{0000, 0101, 0011, 0110\}$. Thus $\mathcal{H}_2$ has minimum distance 2 and is a $[4,2]$ code. Finally, observe that $\mathcal{H}_2^{\perp} = \{0000, 1000, 0111, 1111\}$. ■

3.3 Linearity testing revisited

Consider the function $f \colon F^n \to F$, where F is a finite field, n is a positive integer, and N is the size of the domain of f. The following are three alternative interpretations of what it means for f to be linear:

— There exists a c in F^n such that for all u in F^n, $f(u) = \sum_{i=1}^{n} c_i u_i$.

— The value of f at the sum of two points equals the sum of f evaluated at those two points, i.e. for all u and v in F^n, $f(u) + f(v) = f(u+v)$.

— If a linear combination of points of F^n is $\mathbf{0}$,[5] then the same linear combination of the values of f at those points is also 0, i.e. for any choice of scalars λ_u in F where u varies over F^n, if $\sum_{u \in F^n} \lambda_u u = \mathbf{0}$, then $\sum_{u \in F^n} \lambda_u f(u) = 0$.

[5] For ease of reading the zero element of F^n will appear in boldface type in order to distinguish it from the zero element of F.

The first alternative above tells which functions are linear. The second alternative defines linear functions by one of their properties. This view of linear functions leads to the BLR test: randomly choose two points $u, v \in F^n$ and check that $f(u) + f(v) = f(u+v)$.

The third of the alternatives above also leads to a test for linearity: randomly choose scalars λ_u in F such that $\sum_{u \in F^n} \lambda_u u = \mathbf{0}$, and check that $\sum_{u \in F^n} \lambda_u f(u) = 0$. This test has a fatal drawback. With high probability λ has large *support*, i.e. many nonzero components. Thus, checking that $\sum_{u \in F^n} \lambda_u f(u) = 0$ requires querying too many values of f, something we want to avoid. We can overcome this drawback by picking λ's with small support. In particular, with three nonzero components. This yields the following new test for linearity:

EXTENDED LINEARITY TEST: *Randomly choose distinct $u, v, w \in F^n$, and nonzero scalars $\lambda_u, \lambda_v, \lambda_w \in F$ such that $\lambda_u u + \lambda_v v + \lambda_w w = \mathbf{0}$. Then, reject if $\lambda_u f(u) + \lambda_v f(v) + \lambda_w f(w) \neq 0$, and accept otherwise.*

We call this test the *extended linearity* (EL) test in order to distinguish it from the BLR test. Clearly, it always accepts linear functions. Its main motivation goes back to the multilinearity test of [BFL90].

We will show that the EL test is provably more efficient than the BLR test at the cost of using a negligible amount of additional randomness.[6] Intuitively, the gain in efficiency comes from the fact that the EL test looks at the same constraints that the BLR test does, and more (by a factor of $\Theta(|F|^2)$). Note that if $F = \mathrm{GF}(2)$, then the BLR and the EL test are the same. In the following sections we extend the discrete Fourier analysis technique used to analyze the linearity test over GF(2) (Chapter 2). This extension establishes a connection between testing and the theory of dual codes.

[6] Nevertheless, it should be noted that the BLR test only requires that the function to be tested take values from one finite group into another finite group. In contrast, the EL test requires that the function to be tested take values from F^n into F, where F is a finite field. The finite group assumption is used in the context of program checking, e.g., in [ERS95]. However, the finite field assumption is the standard assumption in the PCP context.

3.4 Tests and dual codes

Consider the set C of all tuples of the form $(\, l(u) : u \in F^n\,)$, where l: $F^n \rightarrow F$ is linear. Note that C is a linear code. Let f: $F^n \rightarrow F$ be given. We denote by C_f the set of the tables of the functions that are linear combinations of f and a linear function, i.e. the set of all tuples of the form $(\,(\phi f + \theta l)(u) : u \in F^n\,)$, where l: $F^n \rightarrow F$ is linear and $\phi, \theta \in F$. The key observation is that the EL test can be rephrased as follows:

> EXTENDED LINEARITY TEST: *Randomly choose a dual codeword $\lambda \in C^{\perp}$ of weight three. Then, accept if $\lambda \notin C_f^{\perp}$, and reject otherwise.*

If we denote by $\mathsf{Rej}(f)$ the probability that the EL test rejects, and by $\mathsf{Dist}(f)$ the distance from f to its closest linear function, then

(*i*) $1 - \mathsf{Rej}(f)$ equals the ratio between the number of codewords of weight three in $C_f^{\perp}$ to the number of codewords of weight three in $C^{\perp}$,

(*ii*) $\mathsf{Dist}(f)$ equals the minimum weight of the code $C_f \setminus C$ normalized by its block length.

Coding theorist have known, for a long time, relationships between the weight distribution of a linear code and its dual.[7] These distributions *cannot* be arbitrary, thus, the relation between $\mathsf{Dist}(f)$ and $\mathsf{Rej}(f)$ *cannot* be arbitrary either. We exploit this fact to derive good lower bounds for $\mathsf{Rej}(f)$ in terms of $\mathsf{Dist}(f)$ for the EL test as well as a closely related test.

In fact, we do more, we take care in setting up a framework that might help in resolving the questions raised in Section 3.1.2.

The scenario we consider is the following: we are given oracle access to a function f: $D \subseteq F^n \rightarrow F$. We abuse notation and view f both as a function and as a table for a function, i.e. as a tuple of length $N = |D|$ with components in F or equivalently as an element of F^N. We want to determine if the table f has a specific property, say it represents a linear function or a low-degree polynomial. Equivalently, we want to verify that f belongs to a specific subset C of F^N. For simplicity's sake, we focus on function properties that are

[7] The relations we are referring to are known as the MacWilliams Theorems. The first of these relations were obtained in [Mac62].

preserved under addition of functions. In other words, we only consider subsets $C \subseteq F^N$ which are subspaces, i.e. linear codes of block length N over the alphabet F. We stress that this restriction is not essential, since the results of coding theory that we will use have been generalized to the case of nonlinear codes.

We associate to f the smallest linear code of block length N containing both f and every codeword in C, i.e.

$$C_f \overset{\text{def}}{=} \{\phi f + \theta g \,|\, g \in C \text{ and } \phi, \theta \in F\}.$$

Note that $C \subseteq C_f$, hence $C_f^{\perp} \subseteq C^{\perp}$. Equality holds, if and only if, f belongs to C. Thus, $C^{\perp} = C_f^{\perp}$ if f exhibits the function property of interest. To test the claim *f belongs to C*, we perform the following:

Duality test: *Randomly choose a codeword $g \in C^{\perp}$ with small support. Then, accept if $g \in C_f^{\perp}$, and reject otherwise.*

Particular choices of the code C and the distribution by which dual codewords are sampled will give specific tests. For example, instances of the duality test are the BLR test over the two element field, the punctured Hadamard code test (Section 2.8.1), and the augmented Hadamard code test (Section 2.8.2).

Remark 3.4.1 There is an issue regarding the duality test that we need to address, its efficiency. In the PCP context we are in general interested in instances of the duality test in which we sample dual codewords whose support is of $O(1)$ size. Typically, we require the sampling procedure to be efficient, i.e. we should be able to sample in time which is polynomial in the size of the input of the function f. In the context of self-testing programs we are not so much concerned with restricting the support of the dual codewords. But, we require that the running time of the overall procedure be faster than that of any correct program for computing f.

We will see that the theory of weight distributions of dual codes tells us that the weight distribution of C_f is fixed once we know the weight distribution of $C_f^{\perp}$. Thus, by performing the duality test we gain information on how much the weight distribution of C_f deviates from that of C. In particular, we gain some information about the minimum weight of $C_f \setminus C$.

This minimum weight, normalized by the block length of $C_f \setminus C$, is the distance from f to the nearest function whose table is represented by one of the codewords in C. The duality test will be a good test if the larger the fraction of values of f that have to be modified in order to obtain a codeword in C, the fewer the number of codewords of small support there are in $C_f^{\perp}$.

The next two sections summarize the part of coding theory that describes the relation between the weight distribution of a code and its dual.

CONVENTIONS: For the remaining part of this chapter F denotes GF(q), where q is a prime power, and $\gamma \overset{\text{def}}{=} q-1$.

3.5 Krawtchouk polynomials

This section provides a handy reference for the basic notions concerning a family of polynomials called *Krawtchouk polynomials.* These polynomials play an important role in several areas of coding theory.

In what follows, we describe the properties of Krawtchouk polynomials that we use in the remaining part of this chapter. For additional facts about these particular type of polynomials we refer the reader to [MS77, Ch. 5.§7] and [vL92, Ch. 1.§2]. Krawtchouk polynomials are orthogonal polynomials. There is a well developed theory concerning orthogonal polynomials, see [Sze75].

Definition 3.5.1 For any positive integer N, the Krawtchouk polynomial $\mathrm{K}_k(x; N, q)$ is[8]

$$\mathrm{K}_k(x;N,q) \overset{\text{def}}{=} \mathrm{K}_k(x) = \sum_{j=0}^{k} (-1)^j \gamma^{k-j} \binom{x}{j}\binom{N-x}{k-j}, \qquad k = 0,1,\ldots,N.$$

Clearly, $\mathrm{K}_k(x)$ is a polynomial of degree k in x.

Multiplying the Taylor series expansion of $(1+\gamma z)^{N-x}$ and $(1-z)^x$ we get the generating

[8] The binomial coefficient $\binom{x}{j}$ denotes $x(x-1)\cdots(x-j+1)/j!$.

function for the sequence $K_0(x), K_1(x), \ldots$,

$$\sum_{k=0}^{\infty} K_k(x)z^k = (1+\gamma z)^{N-x}(1-z)^x. \tag{3.1}$$

If x is an integer with $0 \le x \le N$ the upper limit of summation can be replaced by N. An alternative expression for this generating function follows:

$$\sum_{k=0}^{\infty} K_k(x)z^k = (1+\gamma z)^N \left(1 - \frac{qz}{1+\gamma z}\right)^x$$

This expression shows that the constant coefficient of $K_k(x)$ is $\binom{N}{k}\gamma^k$. From Definition 3.5.1 it follows that the leading coefficient of $K_k(x)$ is $(-q)/k!$.

Differentiating (3.1) with respect to z, multiplying by $(1+\gamma z)(1-z)$, and equating coefficients of z^k shows that Krawtchouk polynomials satisfy the following three term recurrence: $(k+1)K_{k+1}(x) = [(\gamma N - qx) - k(\gamma - 1)]K_k(x) - \gamma(N-k+1)K_{k-1}(x)$, for every positive integer k, with initial values $K_0(x) = 1$, $K_1(x) = \gamma N - qx$.

Krawtchouk polynomials are called orthogonal polynomials due to the following *orthogonality* relations: for every nonnegative integers r, s,

$$\sum_{i=0}^{N} \binom{N}{i} \gamma^i K_r(i) K_s(i) = q^N \gamma^s \binom{N}{s} \delta_{r,s},$$

where $\delta_{r,s}$ is the Kronecker symbol, i.e. $\delta_{r,s} = 1$ if $r = s$, and 0 otherwise.

Rearranging the binomial coefficients in Definition 3.5.1 shows that if i and j are nonnegative integers, then

$$\gamma^i \binom{N}{i} K_j(i) = \gamma^j \binom{N}{j} K_i(j). \tag{3.2}$$

Thus, we can rewrite the orthogonality relations in the following useful way: for every nonnegative integer r, s,

$$\sum_{i=0}^{N} K_r(i) K_i(s) = q^N \delta_{r,s}. \tag{3.3}$$

These orthogonality relations let us express a polynomial $P(x)$ of degree k as a linear combination of $\mathrm{K}_0(x), \mathrm{K}_1(x), \ldots, \mathrm{K}_k(x)$. Formally

Theorem 3.5.2 (see [MS77, Ch. 5.§7]) If $P(x)$ is a polynomial of degree k in x then its Krawtchouk expansion is

$$P(x) = \sum_{j=0}^{k} c_j(P)\mathrm{K}_j(x), \qquad \text{where } c_j(P) = q^{-N}\sum_{i=0}^{N} P(i)\mathrm{K}_i(j).$$

3.6 MacWilliams Theorems

We saw that the Hamming weight, or just the weight, of a codeword equals the number of its nonzero coordinates. We denoted by $A_i(C)$ the number of codewords of weight i in C. The *Hamming weight enumerator* of the code C of block length N is

$$\mathrm{W}_C(x,y) \stackrel{\mathrm{def}}{=} \sum_{i=0}^{N} A_i(C)x^{N-i}y^i.$$

Note that the Hamming weight enumerator of a code tells us everything about its weight distribution, i.e. it completely determines $(A_0(C), \ldots, A_N(C))$. The surprising fact is that if C is a linear code, then the weight enumerator of the dual code $C^\perp$ is completely determined by the weight enumerator of C itself. In fact, it is given by a linear transformation of the weight enumerator of C.

Theorem 3.6.1 (MacWilliams theorem for linear codes, see [MS77, Ch. 5.§6]) If C is a linear code over F

$$\mathrm{W}_{C^\perp}(x,y) = \frac{1}{|C|}\mathrm{W}_C(x+\gamma y, x-y).$$

Corollary 3.6.2 If C is a linear code over F of block length N, then for $k \in \{0, \ldots, N\}$

$$A_k(C^\perp) = \frac{1}{|C|}\sum_{i=0}^{N} A_i(C)\mathrm{K}_k(i).$$

Proof: Setting $x = 1$ in the MacWilliams theorem for linear codes implies that

$$\sum_{k=0}^{N} A_k(C^{\perp})y^k = \frac{1}{|C|}\sum_{i=0}^{N} A_i(C)(1+\gamma y)^{N-i}(1-y)^i .$$

Since (3.1) implies that $(1+\gamma y)^{N-i}(1-y)^i = \sum_{k=0}^{N} \mathrm{K}_k(i)y^k$, the corollary follows by equating the RHS and LHS coefficients of y^k in the expression above. ■

3.7 Analyses of tests through coding theoretic arguments

The theory discussed in the preceding sections can be used to analyze tests. To show this we first develop some general useful results. We then use them to analyze the extended linearity test, and a test for total degree one polynomials over finite fields.

Proposition 3.7.1 Let $P(x)$ be a polynomial of degree k and $c_j(P) = q^{-N}\sum_{i=0}^{N} P(i)\mathrm{K}_i(j)$. Then, for every linear code C of block length N

$$\sum_{j=0}^{k} c_j(P)A_j(C^{\perp}) = \frac{1}{|C|}\sum_{i=0}^{N} A_i(C)P(i) = \mathsf{E}_{g\in_R C}\left[P(\mathrm{wt}(g))\right] .$$

Proof: The second equality is obvious. To prove the first identity recall that by Theorem 3.5.2 we know that $P(i) = \sum_{j=0}^{k} c_j(P)\mathrm{K}_j(i)$. Multiplying both sides of the previous expression by $A_i(C)$, summing over $i \in \{0,\ldots,N\}$, and dividing by $|C|$ we get

$$\frac{1}{|C|}\sum_{i=0}^{N} A_i(C)P(i) = \sum_{j=0}^{k} c_j(P)\left(\frac{1}{|C|}\sum_{i=0}^{N} A_i(C)\mathrm{K}_j(i)\right) .$$

Corollary 3.6.2 implies that the inner summation on the RHS is equal to $A_j(C^{\perp})$. ■

Recall that C_f denotes the smallest linear code containing both C and the table for the function $f\colon F^n \to F$. The following lemma gives expressions for the number of codewords in $C^{\perp}$ that are not in $C_f^{\perp}$.

Lemma 3.7.2 (Duality Testing Lemma) Let $P(x)$ be a polynomial of degree k and let $c_j(P) = q^{-N}\sum_{i=0}^{N} P(i)\mathrm{K}_i(j)$. Then, for any linear code $\mathcal{C}$ of block length N and every $f \in F^N$

$$\sum_{j=0}^{k} c_j(P)\left(A_j(\mathcal{C}^\perp) - A_j(\mathcal{C}_f^\perp)\right) \;=\; \frac{\gamma}{q}\left(\mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(g)) - P(\mathrm{wt}(f-g))\right]\right).$$

Proof: Two applications of Proposition 3.7.1 yield

$$\sum_{j=0}^{k} c_j(P)\left(A_j(\mathcal{C}^\perp) - A_j(\mathcal{C}_f^\perp)\right) \;=\; \mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(g))\right] - \mathsf{E}_{g\in_R\mathcal{C}_f}\left[P(\mathrm{wt}(g))\right].$$

Since, $\mathsf{E}_{g\in_R\mathcal{C}_f}\left[P(\mathrm{wt}(g))\right] = \frac{\gamma}{q}\mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(f-g))\right] + \frac{1}{q}\mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(g))\right]$, the lemma follows. ■

Corollary 3.7.3 Let $P(x)$ be a polynomial of degree k and $\mathcal{C}$ be a linear code of block length N whose dual code does not have codewords of weight $i \in \{1,\ldots,k\}$. Then, for every $f \in F^N$

$$\mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(f-g))\right] \;=\; \mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(g))\right].$$

Proof: Observe that in this case $A_i(\mathcal{C}^\perp) = A_i(\mathcal{C}_f^\perp)$ for all $i \in \{0,\ldots,k\}$, and apply the Duality Testing Lemma. ■

If $\mathcal{C}, P$ satisfy the hypothesis of the above corollary, then $\mathsf{E}_{g\in_R\mathcal{C}}\left[P(\mathrm{wt}(f-g))\right]$ does not depend on $f \in F^N$. The collection of tables of polynomials of degree $k+1$ form a code whose dual does not have codewords of weight $i \in \{1,\ldots,k\}$.

Remark 3.7.4 Let the random variable X be distributed according to a *Binomial* $(N, \gamma/q)$.[9] Observe that by (3.2), we have the following identities for the coefficients in the Krawtchouk expansion of the polynomial P:

$$c_j(P) \;=\; \frac{1}{\binom{N}{j}\gamma^j}\sum_{i=0}^{N}\binom{N}{i}\left(\frac{\gamma}{q}\right)^i\left(\frac{1}{q}\right)^{N-i} P(i)\mathrm{K}_j(i) \;=\; \frac{1}{\binom{N}{j}\gamma^j}\mathsf{E}\left[P(X)\mathrm{K}_j(X)\right].$$

[9] The distribution *Binomial* (m,p) corresponds to the distribution of the sum of m independent identically distributed $\{0,1\}$-random variables with expectation p.

Because of this and other reasons that will later become apparent we recall some facts about the moments about the mean of the binomial distribution. The moment generating function of $X-\mu$, where $\mu=\gamma N/q$, is given by

$$\Phi(t) \overset{\text{def}}{=} \mathsf{E}\left[e^{t(X-\mu)}\right] = e^{-t\mu}\left(pe^{t}+1-p\right)^{N}.$$

If k is a nonnegative integer, then the k-th moment about the mean of random variable X is $\nu_k \overset{\text{def}}{=} \mathsf{E}\left[(X-\gamma N/q)^k\right]$. Moreover, ν_k equals the k-th derivative of $\Phi(t)$ evaluated at $t=0$. Thus, $\nu_0=1$, $\nu_1=0$, $\nu_2=N\gamma/q^2$, $\nu_3=-N\gamma(\gamma-1)/q^3$, $\nu_4=3N^2\gamma^2/q^4+N\gamma\left(1-6\gamma/q^2\right)/q^2$, and $\nu_5=-10N^2\gamma^2(\gamma-1)/q^5-N\gamma(\gamma-1)\left(1+12\gamma/q^2\right)/q^3$.

3.7.1 Testing total degree one polynomials

Assume $f\colon F^n \to F$ is given. As usual, we abuse notation and view f both as a function and as a table for the function f. In this case, we are concerned with checking if f is a polynomial of total degree one, i.e. if f belongs to

$$\mathcal{C} = \{(p(u) : u\in F^n)\,|\,p\colon F^n\to F \text{ is a total degree one polynomial}\}.$$

This code is a generalized version of the augmented Hadamard code (Section 2.8.2). Note, that there are no dual codewords of weight one or two in $\mathcal{C}^\perp$. Thus, the codewords of weight three in $\mathcal{C}^\perp$ are the natural focus of our attention. The duality test becomes

TOTAL DEGREE ONE TEST: *Randomly choose a dual codeword $\lambda\in\mathcal{C}^\perp$ of weight three. Then, accept if $\lambda\in\mathcal{C}_f^\perp$, and reject otherwise.*

Equivalently, randomly select distinct $u,v,w\in F^n$, and nonzero scalars $\lambda_u,\lambda_v,\lambda_w\in F$ such that the following two equalities hold: $\lambda_u+\lambda_v+\lambda_w=0$, and $\lambda_u u+\lambda_v v+\lambda_w w=\mathbf{0}$. Then, accept if $\lambda_u f(u)+\lambda_v f(v)+\lambda_w f(w)=0$, and reject otherwise.

We denote by $\mathsf{Rej}_1(f)$ the rejection probability of the above test, and by $\mathsf{Dist}_1(f)$ the smallest fraction of values in which f differs from a polynomial of total degree one. We now lower bound the former expression in terms of the latter.

Proposition 3.7.5 Let $|F|>2$. For every $f\colon F^n\to F$, $\mathsf{Rej}_1(f)\geq \mathsf{Dist}_1(f)-O(1/|F|)$.

Proof: Clearly, there is exactly one codeword of weight 0 in both $\mathcal{C}^\perp$ and $\mathcal{C}_f^\perp$, hence $A_0(\mathcal{C}^\perp) = A_0(\mathcal{C}_f^\perp) = 1$. Moreover, there are no codewords of weight one or two in $\mathcal{C}^\perp$, thus $A_1(\mathcal{C}^\perp) = A_1(\mathcal{C}_f^\perp) = 0$ and $A_2(\mathcal{C}^\perp) = A_2(\mathcal{C}_f^\perp) = 0$.

Let N be the size of the domain of f. Now, let $P_k(x) = c_k\,(x - \gamma N/q)^k$, where $c_k = (-q)^k/k!$. Since $P_k(x)$ and $\mathrm{K}_k(x)$ have the same leading term, then, in the Krawtchouk expansion of $P_k(x)$ the coefficient multiplying $\mathrm{K}_k(x)$ equals 1, i.e. $c_k(P_k) = 1$, where $c_k(\cdot)$ is as defined in Theorem 3.5.2. Thus, from Proposition 3.7.1 and Remark 3.7.4 we conclude that $\mathsf{E}_{g\in_R\mathcal{C}}\left[P_2(\mathrm{wt}(g))\right] = c_2\nu_2$, and $\mathsf{E}_{g\in_R\mathcal{C}}\left[P_3(\mathrm{wt}(g))\right] = A_3(\mathcal{C}^\perp)+c_3\nu_3$, where ν_2 and ν_3 are the second and third moment about the mean of a *Binomial* $(N,\gamma/q)$, respectively. Hence, the Duality Testing Lemma yields

$$A_3(\mathcal{C}^\perp) - A_3(\mathcal{C}_f^\perp) \;=\; \frac{\gamma}{q}\left(A_3(\mathcal{C}^\perp) + c_3\nu_3 - \mathsf{E}_{g\in_R\mathcal{C}}\left[P_3(\mathrm{wt}(f-g))\right]\right).$$

Observe that the probability that the total degree one test accepts equals the ratio between the number of weight three codewords in $\mathcal{C}_f^\perp$ and $\mathcal{C}^\perp$. Thus, dividing both sides of the above equality by $A_3(\mathcal{C}^\perp)$ shows that

$$\mathsf{Rej}_1(f) \;=\; \frac{\gamma}{q}\left(1 - \frac{c_3}{A_3(\mathcal{C}^\perp)}\left(\mathsf{E}_{g\in_R\mathcal{C}}\left[\left(\mathrm{wt}(f-g) - \gamma N/q\right)^3\right] - \nu_3\right)\right).$$

Moreover, note that a weight three codeword is in $\mathcal{C}^\perp$, if and only if, its support is contained in a line of F^n.[10] Since exactly γ such codewords share the same support, we get that $A_3(\mathcal{C}^\perp) = \gamma\binom{q}{3}L$, where L is the number of lines in F^n, i.e. $L = \binom{N}{2}/\binom{q}{2}$ (here is where we require the assumption that $q = |F| > 2$).

Observe that since a randomly chosen polynomial of total degree one agrees with f in a fraction of $1/q$ places, $\mathsf{Dist}_1(f) \leq \gamma/q$.

Furthermore, since $A_i(\mathcal{C}_f^\perp) = A_i(\mathcal{C}^\perp)$ for $i \in \{0,1,2\}$, Corollary 3.7.3 implies that $\mathsf{E}_{g\in_R\mathcal{C}}\left[\left(\mathrm{wt}(f-g) - \gamma N/q\right)^2\right] = \mathsf{E}_{g\in_R\mathcal{C}}\left[\left(\mathrm{wt}(g) - \gamma N/q\right)^2\right] = \frac{1}{c_2}\mathsf{E}_{g\in_R\mathcal{C}}\left[P_2(\mathrm{wt}(g))\right]$. But, as observed above, $\mathsf{E}_{g\in_R\mathcal{C}}\left[P_2(\mathrm{wt}(g))\right] = c_2\nu_2$. Thus, $\mathsf{E}_{g\in_R\mathcal{C}}\left[\left(\mathrm{wt}(f-g) - \gamma N/q\right)^3\right]$ is at

[10] Recall that a line in F^n is a set of points of the form $\{\, u + tv \in F^n \mid t \in F \,\}$ where $u, v \in F^n$ and $v \neq \mathbf{0}$.

least $\nu_2 N\left(\mathrm{Dist}_1(f) - \gamma/q\right)$. Hence, if $c_3' \stackrel{\mathrm{def}}{=} -c_3/\binom{q}{3} > 0$, then

$$\begin{aligned}\mathrm{Rej}_1(f) &\geq \frac{\gamma}{q}\left(1 + c_3'\frac{\nu_2 N}{\gamma L}\left(\mathrm{Dist}_1(f) - \gamma/q\right) - c_3'\frac{\nu_3}{\gamma L}\right)\\ &\geq \frac{\gamma}{q}c_3'\frac{\nu_2 N}{\gamma L}\mathrm{Dist}_1(f) + \frac{\gamma}{q}\left(1 - \frac{\gamma}{q}c_3'\frac{\nu_2 N}{\gamma L}\right).\end{aligned}$$

The latter inequality follows since $\nu_3 \leq 0$. Finally, note that $\frac{\gamma}{q}c_3'\frac{\nu_2 N}{\gamma L} = 1 + O(1/q)$. ■

3.7.2 Extended linearity test

The scenario is the same as in the preceding section, but now we consider the code C whose elements are of the form $(l(u) : u \in F^n)$ where $l \colon F^n \to F$ is linear. This code is a generalized version of the Hadamard code (Section 2.8).

In Section 3.4 we argued that the EL test can be rephrased in the following form:

> EXTENDED LINEARITY TEST: *Randomly choose a dual codeword $\lambda \in C^\perp$ of weight three. Then, accept if $\lambda \notin C_f^\perp$, and reject otherwise.*

We denote by $\mathrm{Rej}(f)$ the probability that this test rejects and by $\mathrm{Dist}(f)$ the distance from f to its nearest linear function.

Lemma 3.7.6 For every $f \colon F^n \to F$, $\mathrm{Rej}(f) \geq \mathrm{Dist}(f) - O(1/|F|^n)$.

Proof: We restrict our discussion to the case where $f(0) = 0$. The general case follows immediately. The rest of the proof follows the argument of the proof of Proposition 3.7.5.

Let N be the size of the domain of f. Note that $A_0(C_f^\perp) = A_0(C^\perp)$, $A_1(C_f^\perp) = A_1(C^\perp)$, and $A_3(C_f^\perp) \leq A_3(C^\perp) = N^2\gamma^3/3! - O(Nq^3)$. Now, let $P_k(x) = c_k\,(x - \gamma N/q)^k$, where $c_k = (-q)^k/k!$. Since $P_k(x)$ and $\mathrm{K}_k(x)$ have the same leading term, the coefficient multiplying $\mathrm{K}_k(x)$ in the Krawtchouk expansion of $P_k(x)$ is equal to 1, i.e. $c_k(P_k) = 1$, where $c_k(\cdot)$ is as defined in Theorem 3.5.2. Thus, by the Duality Testing Lemma and observing that $\mathrm{Rej}(f) = 1 - A_3(C_f^\perp)/A_3(C^\perp)$, we have

$$\begin{aligned}\mathrm{Rej}(f) \;\geq\; & -\frac{\gamma}{q}\frac{6}{N^2\gamma^3}\mathrm{E}_{g \in_R C}\left[P_3\left(\mathrm{wt}(f-g)\right) - P_3\left(\mathrm{wt}(g)\right)\right] \\ & - c_2(P_3)\frac{6}{N^2\gamma^3}\left(A_2(C^\perp) - A_2(C_f^\perp)\right).\end{aligned} \tag{3.1}$$

There is one codeword of weight 0 in $\mathcal{C}$. All of its other $N-1$ codewords have weight $N\gamma/q$. Hence, $\mathsf{E}_{g\in_R\mathcal{C}}\left[P_3\left(\mathrm{wt}(g)\right)\right] = -c_3N^2\gamma^3/q^3$.

From the recurrence relation for Krawtchouk polynomials and Remark 3.7.4 we get that $c_2(P_3) = \frac{c_3}{2\binom{N}{2}\gamma^2}\left(q^2\nu_5 + q(\gamma-1)\nu_4 - \gamma N\nu_3\right)$, where ν_i is the i-th moment about the mean of a *Binomial* $(N,\gamma/q)$. Hence, $\left|c_2(P_3)\right| = O(q)$. Furthermore, $A_2(\mathcal{C}_f^\perp) \leq A_2(\mathcal{C}^\perp) \leq N\gamma^2$. Thus, if we let $c_3' \stackrel{\mathrm{def}}{=} -6c_3/\gamma^3 > 0$, then (3.1) implies that

$$\mathsf{Rej}(f) \;\geq\; c_3'\frac{\gamma}{q}\frac{1}{N^2}\left(\mathsf{E}_{g\in_R\mathcal{C}}\left[\left(\mathrm{wt}(f-g) - N\gamma/q\right)^3\right] + N^2\gamma^3/q^3\right) - O\left(1/N\right).$$

Moreover, $\mathsf{E}_{g\in_R\mathcal{C}}\left[\left(\mathrm{wt}(f-g)-N\gamma/q\right)^3\right] \geq -\frac{1}{c_2}N\left(\gamma/q - \mathsf{Dist}(f)\right)\mathsf{E}_{g\in_R\mathcal{C}}\left[P_2(\mathrm{wt}(f-g))\right]$.

Since $c_2(P_2) = 1$, $A_0(\mathcal{C}_f^\perp) = A_0(\mathcal{C}^\perp)$, $A_1(\mathcal{C}_f^\perp) = A_1(\mathcal{C}^\perp)$, and $A_2(\mathcal{C}_f^\perp) \leq A_2(\mathcal{C}^\perp)$, by the Duality Testing Lemma, $\mathsf{E}_{g\in_R\mathcal{C}}\left[P_2(\mathrm{wt}(f-g))\right] \leq \mathsf{E}_{g\in_R\mathcal{C}}\left[P_2(\mathrm{wt}(g))\right] = c_2N\gamma^2/q^2$. A randomly chosen linear function agrees with f in a fraction of at least $\left(1-1/N\right)/q$ points. Thus, $\mathsf{Dist}(f) \leq \gamma/q$. Putting everything together shows that

$$\mathsf{Rej}(f) \;\geq\; c_3'\left(\gamma/q\right)^3\mathsf{Dist}(f) - O\left(1/N\right).$$

Finally, observe that $c_3'(\gamma/q)^3 = 1$. ■

The following lower bound complements Lemma 3.7.6. This result is to the EL test as the lower bound of Corollary 2.4.2 ([BGLR93]) is to the BLR test.

Proposition 3.7.7 Let $q = |F|$ and $\gamma = q-1$. For every $f\colon F^n \to F$,

$$\mathsf{Rej}(f) \;\geq\; 3\,\mathsf{Dist}(f)\left(1 - q\mathsf{Dist}(f)/\gamma\right) - O\left(1/q^n\right).$$

Proof: Observe that $\mathsf{Rej}(f) = \Pr\left[\lambda_u f(u) + \lambda_v f(v) + \lambda_w f(w) \neq 0\right] - O\left(1/|F|^n\right)$, where the probability is taken over the choices of the nonzero scalars $\lambda_u, \lambda_v, \lambda_w \in F$ and the points $u, v \in F^n$, and where w is the unique solution of $\lambda_u u + \lambda_v v + \lambda_w w = 0$. Thus,

$$\begin{aligned}\mathsf{Rej}(f) \;\geq\;& 3\Pr\left[\lambda_u f(u) + \lambda_v f(v) \neq 0, f(u)\neq 0, f(v)\neq 0, f(w) = 0\right]\\ &+ 3\Pr\left[f(u)\neq 0, f(v) = 0, f(w) = 0\right] - O\left(1/|F|^n\right).\end{aligned}$$

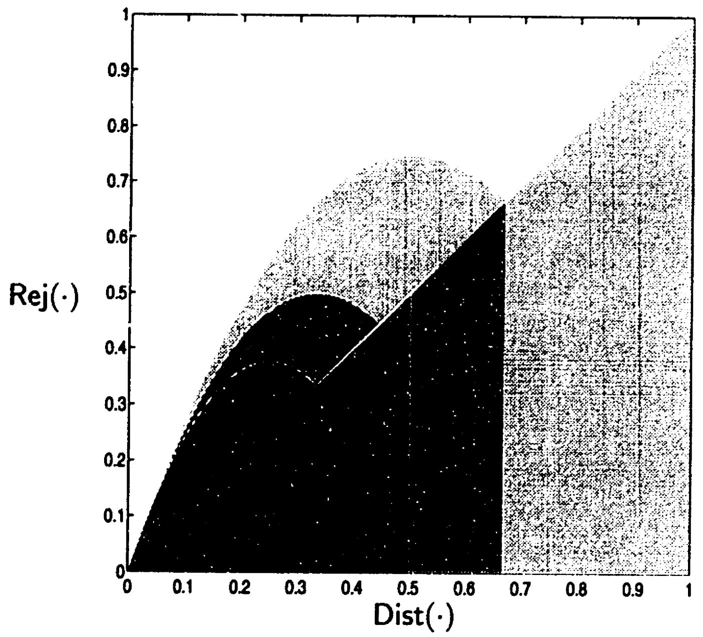

Figure 3-1: Lower bound for the EL test over a large domain. (See text for discussion.)

It follows that,

$$\begin{aligned}\mathsf{Rej}(f) &= -3\Pr\left[\lambda_u f(u)+\lambda_v f(v)=0, f(u)\neq 0, f(v)\neq 0, f(w)=0\right]\\ &\quad +3\Pr\left[f(u)\neq 0, f(v)=0\right]-O\left(1/|F|^n\right).\end{aligned}$$

The first of the latter terms is at least $-3\Pr\left[\lambda_u f(u)+\lambda_v f(v)=0, f(u)\neq 0, f(v)\neq 0\right] = -3\,\mathsf{Dist}(f)^2/\gamma$, and the second term equals $3\,\mathsf{Dist}(f)\,(1-\mathsf{Dist}(f))$. The claim follows. ■

Figure 3-1 illustrates the state of knowledge concerning the EL test when applied to functions on many variables. The dark shaded region represents the lower bound for the EL test when the underlying field is of size 2 (recall that for this case the EL test becomes the BLR test). The medium dark and light shaded regions represent, respectively, the lower bounds for the EL test when the underlying field is of size 3 and when the underlying field is very large.

3.7.3 Discussion

The techniques discussed in this chapter seem to be especially well suited for achieving nontrivial lower bounds on the rejection probability of a test even when this probability is high. Our approach takes a function and associates to it a combinatorial object (a code). Then, we try to show that codes with some specific combinatorial characteristic cannot exist. Hence, functions at a given distance from the underlying space of interest and with some particular probability of failing the test in point cannot exist either. This argument does not need to make any assumptions about the magnitude of the rejection probability of the test. We feel that this is the main reason why the analyses of Section 3.7.2 and Section 3.7.1 show nontrivial lower bounds even when the rejection probabilities are high.

The techniques discussed in Section 3.1.3 yield two results that show bounds of the kind obtained in this chapter. Indeed, Arora[Aro95a] uses algebraic arguments to prove a lower bound of the form referred before for the low individual degree test, but under the assumption that the underlying field is of size *exponential* in the degree. For the case of testing *univariate* polynomials, Sudan [Sud92] shows that such lower bounds can be obtained for the basic univariate test.[11] We are not aware of other analyses that yield lower bounds of the kind obtained in this chapter for tests involving *multivariate* functions and with nothing but the most basic assumptions on the size of the underlying field.

Another appealing aspect of the approach we have discussed is that it does not require to impose restrictions on the domain of the functions to be tested (other than it should be a subset of $\mathrm{GF}(q)^n$). This is illustrated by the analysis of the punctured Hadamard code test discussed in Section 2.8.1.

The techniques we have employed have, so far, failed to give nontrivial lower bounds for tests concerning *multivariate* polynomials whose total degree is not necessarily bounded by one. This issue needs to be addressed in order for this chapter's arguments to have a wider applicability.

[11] An alternative proof of this result, albeit a more involved one, can be obtained with the arguments used in this chapter.

CHAPTER 4

Alternation in Interaction

Assume we have an assertion whose validity we want to verify. We consider the scenario in which two provers will assist us in this endeavor. There is a key feature in the scenario we focus on: the two provers have opposite claims regarding the assertion's validity. However, only one of the provers' claim is correct. To determine which claim is correct we carry out a probabilistic interaction with these two provers. The interaction consists of a question to each prover. The provers respond with short answers. However, we impose a crucial asymmetry between the two provers. The second prover picks his strategy, for responding, *after* the first one does. Performing a simple computation on the provers' responses, one decides to accept or reject the first prover's claim. Incorrect claims are accepted with a negligible probability. The question is: *which assertions can be thus verified?* Here, we show that probabilistically checkable proofs allow efficient verification through a probabilistic interaction with these two provers.

We also address the above stated question in a more general setting, where a finite number of provers, instead of only two, aid us in the verification process. But again, the distinguishing characteristic of the scenario we consider is the hierarchical relation amongst the provers.

Let us begin by precisely describing the interactive proof systems on which we focus.

4.1 Competing prover proof systems

In order to formally describe the problem we are interested in, it is necessary to start by discussing the context in which it arises, namely, *interactive proof systems.*

Consider the following scenario: two characters, an all-powerful prover and a computationally bounded verifier, want to decide if a given word ω is in a pre-specified language L. Both prover and verifier have access to the word ω and to secret sources of unbiased random bits. Prover and verifier interact by sending each other messages according to some protocol enforced by the verifier. Interaction proceeds in *rounds.* A round consists of a question computed by the verifier which is sent to the prover, and the prover's answer to it. The verifier's questions is a function of the input ω, his random bits, and the communication that has taken place. The prover answers according to a fixed *strategy*, a function of the input ω, the prover's random bits, and the communication history. After a finite number of rounds, the verifier is required to either *accept* or *reject* the provers claim that ω is in L. The goal of the prover is to make the verifier accept. The goal of the verifier is to determine the truth of the prover's claim.

An *interactive proof system for language L* is one where the scenario is as described above, the verifier is a probabilistic polynomial-time Turing machine, and such that

— COMPLETENESS: For every string ω in L, there is a strategy for prover **P** that causes the verifier to always accept.

— SOUNDNESS: For every string ω not in L, and any strategy for prover **P**, the verifier accepts with probability at most 1/3 (where the probability is taken over the verifier's and prover's random bits).

The class IP is the collection of languages that are recognizable by interactive proof systems. Interactive proof systems and the class IP were independently proposed by Babai [Bab85] and Goldwasser, Micali, and Rackoff [GMR85]. A major question left open by these papers was to determine the precise relation between IP and classical complexity classes. This was finally settled by Shamir *et al.* [LFKN90, Sha90] who showed that IP = PSPACE.

In order to allow for perfect zero-knowledge proofs without computational assumptions, Ben-Or, Goldwasser, Kilian, and Wigderson [BGKW88] introduced the concept of a *multi-prover interactive proof.* Here, the scenario is similar to the one of interactive proof systems, now, however, there are k powerful provers with whom the verifier interacts. Moreover, there is a crucial restriction on the provers, they cannot communicate with each other.

Formally, a *multi-prover interactive proof system for language L* is one where the scenario is as described in the previous paragraph, the verifier is a probabilistic polynomial-time Turing machine, and there exists a nonnegative integer k such that the following holds:

— COMPLETENESS: For every string ω in L, there are strategies for provers $\mathbf{P}_1, \ldots, \mathbf{P}_k$ that cause the verifier to always accept.

— SOUNDNESS: For every string ω not in L, and any strategy for provers $\mathbf{P}_1, \ldots, \mathbf{P}_k$, the verifier accepts with probability at most $1/3$ (where the probability is taken over the verifier's and provers' random bits).

The set of languages recognizable by multi-prover interactive proof systems is denoted MIP. Clearly, IP $\subseteq$ MIP. Babai, Fortnow, and Lund [BFL90] showed that NEXP = MIP. This characterization was strengthened when [LS91, FL92] showed that NEXP languages have two-prover one-round proof systems.

The class of languages recognized by multi-prover proof systems remains the same if the provers loose access to their sources of randomness.

For a more thorough account of the work in interactive proof systems we refer the reader to the surveys of Johnson [Joh88, Joh92].

Another line of research deriving from the MIP = NEXP result concerns the alternative characterization of MIP proposed by Fortnow, Rompel, and Siper [FRS88]. This alternative characterization is what more modern terminology refers to as *probabilistically checkable proofs* (PCPs). Here, the scenario is as in interactive proof systems, but instead of the verifier having access to a prover he has random access to a proof tape (formally an oracle). More precisely, we say that language L can be recognized by a probabilistically checkable proof system using $r(n)$ random bits and $q(n)$ query bits, denoted $L \in \text{PCP}(r(n), q(n))$, if there exists a probabilistic oracle Turing machine that runs in time polynomial in the size of the input, n, uses at most $O(r(n))$ random bits, queries the oracle in at most $O(q(n))$

places, and such that the following holds:

— COMPLETENESS: For every string ω in L, there exists an oracle that causes the verifier to always accept.

— SOUNDNESS: For every string ω not in L, no matter what the oracle is, the verifier accepts with probability at most 1/3 (where the probability is taken over the verifier's random bits).

In essence, [FRS88] shows that the class of languages having probabilistically checkable proofs is precisely MIP, hence, NEXP = PCP($n^{O(1)}, n^{O(1)}$) [BFL90]. This result was followed by a sequence of highly technical developments that led to the celebrated NP = PCP($\log n, 1$) result of Arora, Lund, Motwani, Sudan, and Szegedy [ALM+92]. Key among this sequence of developments were the papers of Babai, Fortnow, Levin, and Szegedy [BFLS91], Feige, Goldwasser, Lovász, Safra, and Szegedy [FGL+91], and Arora and Safra [AS92b].

More recently, Feige and Kilian [FK94], building on top of the NP = PCP($\log n, 1$) result of [ALM+92], show that NP is characterized by two-prover one-round proof systems in which the verifier has access to only $O(\log n)$ random bits, the provers' responses are constant size, and the probability that the verifier incorrectly accepts a string not in the language is at most ϵ, where ϵ can be an arbitrarily small positive constant.

In the prover proof systems that we have discussed so far, *all* the provers are trying to convince the verifier of the validity of a given statement. In other words, the provers always *cooperate* with each other. The framework of cooperating provers has received much attention. A natural question arises: what happens in a scenario in which provers *compete* with each other?

We are led to consider the scenario of *competing-prover proof systems.* In this scenario two teams of competing provers $(P_i : i \in I)$ and $(P_i : i \in \bar{I})$, (where $I \subseteq \{0, \ldots, k-1\}$, and $\bar{I} = \{0, \ldots, k-1\} \setminus I$), interact with a probabilistic polynomial-time verifier. For some pre-specified word ω and language L, the first team of provers tries to convince the verifier that ω is in L, the second team has the opposite objective. The verifier knows the team to which each prover belongs but does not know which team to trust. Before the protocol begins, the provers fix their strategies in the order specified by their subindices. These strategies are deterministic.[1] To model the situation of k competing provers, we propose and study the class k-APP of languages that have k-alternating-prover one-round proof systems.

Definition 4.1.1 (k-Alternating-prover proof systems) A language L is said to have a k-APP system with parameters $(r(n), q(n), [\mathrm{Q}_0, \ldots, \mathrm{Q}_{k-1}])$, $\mathrm{Q}_i \in \{\exists, \forall\}$, and error $(\epsilon_{acc}, \epsilon_{rej})$ if there is a probabilistic verifier V which runs in time polynomial in the size of the input, n, interacts with two teams of competing provers, and the following holds:

if $\omega \in L$, then $\mathrm{Q}_0 P_0, \ldots, \mathrm{Q}_{k-1} P_{k-1}$ such that

$$\Pr_\rho[(V \leftrightarrow P_0, \ldots, P_{k-1})(\omega, \rho) \ \mathit{accepts}] \ \geq \ 1 - \epsilon_{acc}$$

if $\omega \notin L$, then $\overline{\mathrm{Q}}_0 P_0, \ldots, \overline{\mathrm{Q}}_{k-1} P_{k-1}$ such that[2]

$$\Pr_\rho[(V \leftrightarrow P_0, \ldots, P_{k-1})(\omega, \rho) \ \mathit{accepts}] \ \leq \ \epsilon_{rej},$$

where the probabilities are taken over the random coin tosses of V and the quantifiers range over the set of deterministic provers' strategies. Moreover, the verifier makes only one question to each prover. The verifier is non-adaptive, i.e. the question sent to a prover may not depend on the other provers' responses. Furthermore, V uses $O(r(n))$ coin flips and the provers' responses are of size $O(q(n))$.

A key feature of alternating-prover proof systems is the ranking of the provers according to their subindices. As in multi-prover proof systems no prover has access to the communication generated by or directed to any other prover. But, each prover knows the strategies that higher ranked provers use in answering the questions posed to them. The proof systems previously studied in the literature that more closely resemble the one of Definition 4.1.1 where

[1] We will later see that it is important to distinguish between the cases where the provers can have randomized as opposed to deterministic strategies.

[2] For $\mathrm{Q} \in \{\exists, \forall\}$, $\overline{\mathrm{Q}}$ denotes $\forall$ if $\mathrm{Q} = \exists$, and $\exists$ otherwise.

proposed by Feige, Shamir, and Tennenholtz [FST87]. In the proof systems of [FST87] the provers do not have access to each others' strategies, the verifier is adaptive and may ask many questions to each prover. Yet, the use, in a complexity theoretic setting, of the notion of competition among independent decision makers is older. Stockmeyer [Sto76] used games between competing players to characterize languages in the polynomial hierarchy. These games are similar to the proof systems studied in [FST87] but now the verifier is restricted to being deterministic and the provers get to see each other messages. Other uses of competing players to study complexity classes include Condon [Con88], Feigenbaum, Koller, and Shor [FKS95], Reif [Rei79], Russell and Sundaram [RS95], and Peterson and Reif [PR79].

More recently, Condon, Feigenbaum, Lund, and Shor [CFLS93, CFLS94] introduced *probabilistically checkable debate systems.* In these proof systems two players take turns writing down messages. Players have access to what other players write down. The verifier does not directly interact with the players but has access to their *debate*, i.e. the messages that the players write down. In [CFLS93] it is shown that, with suitable restrictions in the computational resources involved, PSPACE is characterized by probabilistically checkable debate systems. We are interested in this kind of proof systems when the debate is restricted to k rounds. This motivates our following definition.

Definition 4.1.2 (k-Alternating-oracle proof systems) A language L is said to have a k-AOP system with parameters $(r(n), q(n), [\mathbf{Q}_0, \ldots, \mathbf{Q}_{k-1}])$, $\mathbf{Q}_i \in \{\exists, \forall\}$, and error $(\epsilon_{acc}, \epsilon_{rej})$ if there is a probabilistic verifier V which runs in time polynomial in the size of the input, n, queries two teams of competing oracles, and the following holds:

if $\omega \in L$ then $\mathbf{Q}_0 O_0, \ldots, \mathbf{Q}_{k-1} O_{k-1}$ such that

$$\Pr_\rho[(V \leftrightarrow O_0, \ldots, O_{k-1})(\omega, \rho) \ \textit{accepts}] \ \geq \ 1 - \epsilon_{acc},$$

if $\omega \notin L$ then $\overline{\mathbf{Q}}_0 O_0, \ldots, \overline{\mathbf{Q}}_{k-1} O_{k-1}$ such that

$$\Pr_\rho[(V \leftrightarrow O_0, \ldots, O_{k-1})(\omega, \rho) \ \textit{accepts}] \ \leq \ \epsilon_{rej},$$

where the probabilities are taken over the random coin tosses of V and the quantifiers range over the set of possible oracles. The verifier is non-adaptive, i.e. the queries are selected before any of them is performed. Furthermore, V uses $O(r(n))$ coin flips and queries $O(q(n))$ bits.

There is a crucial difference between alternating-prover proof systems and alternating-oracle proof systems. Provers may be asked only one question, whereas oracles may be asked multiple questions but their answers may not depend on the other questions. This requirement was labeled in [FRS88] as the oracle requirement.

CONVENTIONS: When referring to a k-alternating-prover proof system with parameters $(r(n), q(n), [\mathbf{Q}_0, \ldots, \mathbf{Q}_{k-1}])$, $\mathbf{Q}_i \in \{\exists, \forall\}$, and error $(\epsilon_{acc}, \epsilon_{rej})$, if we omit $r(n)$ and $q(n)$ they are assumed to be $\log n$ and 1 respectively; when the quantifiers do not appear, they are assumed to alternate beginning with $\exists$; when $\epsilon_{acc} = \epsilon_{rej} = \epsilon$, we say the system has error ϵ. We abbreviate the sequence of $2t$ alternating quantifiers beginning with $\exists$ by $(\exists, \forall)^t$. We define the class k-APP to be the set of languages that have a k-alternating-prover proof system with error $1/3$. We adopt similar conventions for k-alternating-oracle proof systems. We follow the standard custom of denoting the length of the string ω by n. Finally, we denote provers by boldface letters, and their strategies by italic letters.

4.1.1 Main results

Recall that, in many different scenarios, cooperating-prover one-round proof systems are equivalent in power to oracle proof systems. In this chapter we establish conditions under which alternating-prover and alternating-oracle proof systems are equivalent in power.

We show that alternating-prover proof systems are weakened if we allow the provers to have access to secret sources of unbiased random bits. In contrast the power of cooperating-prover proof systems, remains unchanged if the provers are allowed to use private coins.

We characterize the k-th level of the polynomial hierarchy in terms of k-alternating-oracle proof systems. This extends the NP = PCP($\log n$, 1) result of [ALM+92]. Finally, we give a precise characterization of each level of the polynomial hierarchy in terms of alternating-prover proof systems. The main result of this chapter is

Theorem 4.1.3 For every constant ϵ, $0 < \epsilon < 1/2$, a language L is in Σ_k^P, if and only if, it has a $(k+1)$-alternating-prover proof system with error ϵ.

4.1.2 Chapter organization

In Section 4.2 we characterize NP in terms of two-alternating-prover proof systems. In Section 4.3 we study the effect of allowing the provers of an alternating-prover proof system to have access to private coins. In Section 4.4 we characterize each level of the polynomial hierarchy in terms of alternating-oracle proof systems. In Section 4.5 we review some results of Feige and Kilian [FK94]. In Section 4.6 we prove the main result of this chapter.

4.2 Two-alternating-prover proof systems for NP

Suppose that a language L can be recognized by a two-cooperating-prover one-round proof system with verifier $\widehat{V}$ and provers $\widehat{\mathbf{P}}_0$ and $\widehat{\mathbf{P}}_1$ quantified by $([\exists, \exists])$. The verifier $\widehat{V}$ on random string r, asks prover $\widehat{\mathbf{P}}_0$ question $q^{(0)}(r)$, asks prover $\widehat{\mathbf{P}}_1$ question $q^{(1)}(r)$, and uses the answers to the questions to decide whether or not to accept. We will transform this system into a two-alternating-prover proof system with verifier V and provers $\mathbf{P}_0$ and $\mathbf{P}_1$ quantified by $([\exists, \forall])$. In this latter system, prover $\mathbf{P}_0$ claims that there exist provers $\widehat{\mathbf{P}}_0$ and $\widehat{\mathbf{P}}_1$ that would convince $\widehat{V}$ to accept. Prover $\mathbf{P}_1$ is trying to show that $\mathbf{P}_0$ is wrong. The verifier V will simulate the verifier $\widehat{V}$ from the original system to generate the questions $q^{(0)}(r)$ and $q^{(1)}(r)$ that the verifier $\widehat{V}$ would have asked the cooperating provers. To justify his claim, $\mathbf{P}_0$ will tell V what $\widehat{\mathbf{P}}_0$ or $\widehat{\mathbf{P}}_1$ would have said in answer to either question. To test $\mathbf{P}_0$'s claim, V will pick one of the two questions $q^{(0)}(r)$ and $q^{(1)}(r)$ at random and ask $\mathbf{P}_0$ to respond with what the corresponding prover would have said in answer to the question. Unfortunately, this does not enable V to get the answer to both questions. To solve this problem, we recall that $\mathbf{P}_1$ knows what $\mathbf{P}_0$ would answer to any question. Thus, V will send both questions to $\mathbf{P}_1$, and request that $\mathbf{P}_1$ respond by saying how $\mathbf{P}_0$ would have answered the questions.

If $\omega \in L$, then $\mathbf{P}_0$ is honest, and $\mathbf{P}_1$ has to lie about what $\mathbf{P}_0$ would say in order to get the verifier to reject. If $\mathbf{P}_1$ lies about what $\mathbf{P}_0$ said, he will be caught with probability at least $1/2$, because $\mathbf{P}_1$ does not know which question V sent to $\mathbf{P}_0$. Thus, the verifier will accept with probability at least $1/2$.

On the other hand, if $\omega \notin L$, then $\mathbf{P}_1$ will honestly answer by telling V what $\mathbf{P}_0$ would have answered to both questions. In this case, V will accept only if the provers that $\mathbf{P}_0$ represent would have caused $\widehat{V}$ to accept. Thus, we obtain a two-alternating-prover proof system with error $(1/2, \epsilon)$.

We now show how we can balance these error probabilities by a 'parallelization' of the above protocol with an unusual acceptance criteria.

Lemma 4.2.1 A two-cooperating-prover one-round proof system with error $(0, \epsilon)$ can be simulated by a two-alternating-prover one-round proof system with error $(2^{-m}, m2^{m-1}\epsilon)$, an m-fold increase in the verifier's randomness, and an $m2^m$-fold increase in the length of the answers returned by the provers.

Proof: Let $\widehat{V}$ denote the verifier and $\widehat{\mathbf{P}}_0, \widehat{\mathbf{P}}_1$ the provers of the two-cooperating-prover proof system. Now, consider the two-alternating-prover proof system with verifier V and provers $\mathbf{P}_0, \mathbf{P}_1$ that execute the following protocol, henceforth referred to as the BASIC SIMULATION PROTOCOL:

THE QUERIES

— Verifier V generates m pairs of questions as the old verifier $\widehat{V}$ would have, i.e. V selects random strings $r_1, \ldots, r_m$ and generates $\left(\left(q^{(0)}(r_s), q^{(1)}(r_s)\right) : s = 1, \ldots, m\right)$.

— V then picks one question from each pair, i.e. randomly chooses $\vec{j} = (j_1, \ldots, j_m)$ in $\{0, 1\}^m$, and sends to $\mathbf{P}_0$ the questions $\left(q^{(j_s)}(r_s) : s = 1, \ldots, m\right)$ and $\vec{j}$.

— V sends to $\mathbf{P}_1$ all the tuple of questions $\left(\left(q^{(0)}(r_s), q^{(1)}(r_s)\right) : s = 1, \ldots, m\right)$.

THE RESPONSES

— $\mathbf{P}_0$ is supposed to respond with $\left(\widehat{P}_{j_s}(q^{(j_s)}(r_s)) : s = 1, \ldots, m\right)$, which is how $\widehat{\mathbf{P}}_0$ and $\widehat{\mathbf{P}}_1$ would have answered the questions.

— $\mathbf{P}_1$ is supposed to respond with what $\mathbf{P}_0$ would have said to $\vec{\imath} = (i_1, \ldots, i_m)$ and $\left(q^{(i_s)}(r_s) : s = 1, \ldots, m\right)$, henceforth denoted $\left(a_s^{\vec{\imath}} : s = 1, \ldots, m\right)$, for every one of the 2^m possible assignments to $\vec{\imath}$.[3]

[3] We have to allow $\mathbf{P}_1$ to answer this way. This is because a cheating prover $\mathbf{P}_0$ may vary its strategy according to the value of $\vec{j}$ that it receives, but which $\mathbf{P}_1$ does not see.

ACCEPTANCE CRITERIA

— The verifier V accepts if $\mathbf{P}_1$ did not correctly represent what $\mathbf{P}_0$ said on the tuple of questions that $\mathbf{P}_0$ was asked.

— Otherwise, V accepts if there exists at least one $s \in \{1, \dots, m\}$ and index vector $\vec{\imath}$ such that $(\vec{\imath})_s \neq (\vec{\jmath})_s$ and $\widehat{V}$ on random string r_s would accept if given answers $a_s^{\vec{\imath}}$ from $\widehat{\mathbf{P}}_{(\vec{\imath})_s}$ and $a_s^{\vec{\jmath}}$ from $\widehat{\mathbf{P}}_{(\vec{\jmath})_s}$.

Assume that the two-cooperating-prover proof system accepts. Then, $\mathbf{P}_0$ is honest. Hence, $\mathbf{P}_0$ truthfully answers each question he is asked as $\widehat{\mathbf{P}}_0$ would have. We will see that this forces $\mathbf{P}_1$ to lie in response to all but the set of indices $\vec{\jmath}$. Indeed, suppose there is an m-tuple of questions indexed by $\vec{\imath}$, $\vec{\imath} \neq \vec{\jmath}$, for which $\mathbf{P}_1$ honestly represents what $\mathbf{P}_0$ would say. Then there must be an s such that $(\vec{\imath})_s \neq (\vec{\jmath})_s$. If $\mathbf{P}_1$ honestly represented $\mathbf{P}_0$'s answers on tuple $\vec{\imath}$, the verifier V would accept, since $\widehat{V}$ on random string r_s would accept if given answers $a_s^{\vec{\imath}}$ from $\widehat{\mathbf{P}}_{(\vec{\imath})_s}$ and $a_s^{\vec{\jmath}}$ from $\widehat{\mathbf{P}}_{(\vec{\jmath})_s}$. Since $\mathbf{P}_1$ does not see the randomly chosen $\vec{\jmath}$, he decides to honestly answer the questions sent to prover $\mathbf{P}_0$ with probability at most 2^{-m}. Hence, the probability that V accepts is at least $1 - 2^{-m}$.

On the other hand, assume that the two-cooperating-prover proof system rejects. Then the probability that V accepts is at most ϵ for each s, and $\vec{\imath}$ such that $(\vec{\imath})_s \neq (\vec{\jmath})_s$, for a total error of $m2^{m-1}\epsilon$. ■

Lemma 4.2.1 yields an exponential-in-m blowup in the size of the provers' responses. This may seem to be a fatal drawback. But, we will apply this lemma in situations where m is not too large and ϵ is very small.

We will combine the preceding lemma with the theorem of Feige and Kilian [FK94], which we state below in our terminology:

Theorem 4.2.2 (Feige-Kilian [FK94]) For every constant α, $0 < \alpha < 1/2$, a language L is in NP, if and only if, L has a two-alternating-prover proof system with parameters $([\exists, \exists])$ and error $(0, \alpha)$.

Remark 4.2.3 The proof of Theorem 4.2.2 given in [FK94] is based on a proof of a weak version of the parallel repetition theorem. Shortly after [FK94] appeared, Raz [Raz95] succeeded in proving the parallel repetition theorem. Thus, the above claim can now be proved by combining the NP $=$ PCP$(\log n, 1)$ result of [ALM$^+$92] with the techniques from [FRS88] for simulating an oracle by a pair of provers, and then reducing the error of the proof system thus derived by parallel repetition. This yields an alternative proof of Theorem 4.2.2 where the hidden constants are better (smaller).

Corollary 4.2.4 For every constant ϵ, $0 < \epsilon < 1/2$. A language L is in NP, if and only if, L has a two-alternating-prover proof system with error ϵ.

Proof: The forward direction follows immediately from Lemma 4.2.1 and Theorem 4.2.2 by choosing an appropriate integer m and a positive constant α such that $2^{-m} \leq \epsilon$ and $m2^{m-1}\alpha \leq \epsilon$. For the reverse direction, assume L has a two-alternating-prover proof system with error ϵ. Observe that the strategy of the first prover is of length polynomial in the size of the input ω. Thus, given ω we can non-deterministically guess the optimal strategy of the first prover. We then compute the optimal strategy for the second prover. This can be done in polynomial time. It is now possible to compute the acceptance probability efficiently by simulating the verifier's behavior on each of the $2^{O(\log(|\omega|))}$ random strings. The probability thus computed is at least $1 - \epsilon$, if and only if, $\omega \in L$. ■

Corollary 4.2.5 A language L is in NEXP, if and only if, L has a two-alternating-prover proof system with parameters $(poly(n), poly(n), [\exists, \forall])$ and error $(1/poly(n), 1/exp(n))$.[4]

Proof: We first prove the forward direction. Observe that the NEXP characterization of [FL92] in terms of multi-prover interactive proof systems, can be rephrased (in our terminology) as: a language is in NEXP, if and only if, it can be recognized by a two-alternating-prover proof system with parameters $(poly(n), poly(n), [\exists, \exists])$ and error $(0, 1/exp(n))$. Hence, the corollary follows again from Lemma 4.2.1. The reverse implication follows by properly scaling the proof of the reverse direction of Corollary 4.2.4.

■

[4] The terms $poly(n)$ and $exp(n)$ refer to $O(n^c)$ and $O\left(2^{n^c}\right)$ respectively, where c is some positive constant.

4.2.1 The Robust Simulation Protocol

In Section 4.6, we will need a stronger version of Lemma 4.2.1. This section describes extensions to the Basic Simulation Protocol that yield such strengthening.

In the previous section we saw how two-cooperating-prover proof systems with one sided error could be simulated by two-alternating-prover proof systems. We now study the possibility of carrying out a similar simulation of k-cooperating-prover proof systems, even in the case that the error is two sided. In addition, the simulation is carried out in a more demanding scenario.

As in the Basic Simulation Protocol we consider an alternating-prover proof system verifier V who *concurrently* simulates m rounds of the protocol performed by a cooperating-prover proof system verifier $\widehat{V}$. We refer to each one of these simultaneously performed rounds as a *run*. For each one of these m runs V assigns a label of accept/reject based on computations and simulations of $\widehat{V}$. We model the situation where some of the provers' responses are maliciously corrupted. We do this by allowing up to a ξ-fraction of the labels to be arbitrarily flipped. We refer to these flipped labels as *wild-cards*. The verifier V now decides to accept or reject based on the total number of accept labels. We refer to these systems as *alternating-prover proof systems with a ξ-fraction of wild-cards.*

Lemma 4.2.6 Let $\beta > 1$ and $e^{\beta}\beta^{-\beta} = 1/e$. A k-cooperating-prover one-round proof system with error $(\epsilon_{acc}, \epsilon_{rej})$ can be simulated by a two-alternating-prover one-round proof system with a ξ-fraction of wild-cards, with error $\left(e^{-m\epsilon_{acc}} + (1-1/k)^{(1-2\xi-\beta\epsilon_{acc})m}, mk^{(m-1)(k-1)}\epsilon_{rej}\right)$, an m-fold increase in the verifier's randomness and an mk^m-fold increase in the length of the answers returned by the provers.

Proof: Let $\widehat{V}$ denote the k-cooperating-prover proof systems' verifier and $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_{k-1}$ its provers. Moreover, denote by V the verifier and by $\mathbf{P}_0, \mathbf{P}_1$ the provers of the two-alternating-prover proof system. We will now describe the protocol that the verifier V executes. This protocol is henceforth referred to as the Robust Simulation Protocol with a ξ-Fraction of Wild-cards.

THE QUERIES

— V generates m k-tuples of questions as $\widehat{V}$ would have, i.e. V selects random strings $r_1, \ldots, r_m$ and generates $\Big(\big(q^{(0)}(r_s), \ldots, q^{(k-1)}(r_s)\big) \; : \; s = 1, \ldots, m \Big)$.

— V then chooses one question from each k-tuple, i.e. randomly selects $\vec{\jmath} = (j_1, \ldots, j_m)$ in $\{0, \ldots, k-1\}^m$, and sends to $\mathbf{P}_0$ both $\vec{\jmath}$ and the questions $\Big(q^{(j_s)}(r_s) \; : \; s = 1, \ldots, m \Big)$.

— V sends to $\mathbf{P}_1$ the tuple of questions $\Big(\big(q^{(0)}(r_s), \ldots, q^{(k-1)}(r_s)\big) \; : \; s = 1, \ldots, m \Big)$.

THE RESPONSES

— $\mathbf{P}_0$ is supposed to respond with $\Big(\widehat{P}_{j_s}(q^{(j_s)}(r_s)) \; : \; s = 1, \ldots, m \Big)$, which is how $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_{k-1}$ would have answered the questions.

— Prover $\mathbf{P}_1$ is supposed to respond with what prover $\mathbf{P}_0$ would have answered to $\Big(q^{(i_s)}(r_s) \; : \; s = 1, \ldots, m \Big)$ and $\vec{\imath} = (i_1, \ldots, i_m)$, say $\Big(a_s^{\vec{\imath}} \; : \; s = 1, \ldots, m \Big)$, for every one of the k^m possible assignments to $\vec{\imath}$.[5]

ACCEPTANCE CRITERIA

— V accepts if $\mathbf{P}_1$ did not correctly represent what $\mathbf{P}_0$ said on the tuple of questions that $\mathbf{P}_0$ was asked.

— Otherwise, the verifier V labels a run s accept if there are index vectors $\vec{\imath}_1, \ldots, \vec{\imath}_{k-1}$ such that $(\vec{\jmath})_s$ and $(\vec{\imath}_1)_s, \ldots, (\vec{\imath}_{k-1})_s$ are all distinct, and $\widehat{V}$ on random string r_s accepts given answers $a_s^{\vec{\jmath}}$ from $\widehat{\mathbf{P}}_{(\vec{\jmath})_s}$ and $a_s^{\vec{\imath}_1}, \ldots, a_s^{\vec{\imath}_{k-1}}$ from $\widehat{\mathbf{P}}_{(\vec{\imath}_1)_s}, \ldots, \widehat{\mathbf{P}}_{(\vec{\imath}_{k-1})_s}$ respectively. Up to a ξ-fraction of these labels could be wild-cards. The verifier V accepts if there are at least $\xi m + 1$ runs labeled accept.

Assume the cooperating-prover proof system accepts. Then $\mathbf{P}_0$ honestly answers the question he is asked in each run. Let p be the fraction of the random strings which V selects that would cause $\widehat{V}$ to reject on input ω. If $\mathbf{P}_1$ answers honestly in more than a

[5] We have to allow $\mathbf{P}_1$ to answer this way. This is because a cheating prover $\mathbf{P}_0$ may vary its strategy according to the value of $\vec{\jmath}$ that it receives, but which $\mathbf{P}_1$ does not see.

$(1-1/k)^{(1-2\xi-p)m}$ fraction of the $\vec{\imath}$'s, then V labels at least $2\xi m+1$ of the runs with accept. At most ξm of these labels are wild-cards. Hence, V accepts. A multiplicative Chernoff bound (see [AS92a, Theorem A.12]) shows that the probability that p is greater than $\beta\epsilon_{acc}$ is at most $(e^{\beta}\beta^{-\beta})^{m\epsilon_{acc}}$, which by our choice of β equals $e^{-m\epsilon_{acc}}$. Hence, the verifier V rejects with probability at most $e^{-m\epsilon_{acc}} + (1-1/k)^{(1-2\xi-\beta\epsilon_{acc})m}$.

Assume now that the cooperating-prover proof system rejects. The verifier V accepts if at least $\xi m+1$ of the runs get labeled accept. But, ξm labels could be wild-cards. The probability that there is one more accept label is at most ϵ_{rej} for each s, and $\vec{\imath}_1,\ldots,\vec{\imath}_{k-1}$ such that $(\vec{\jmath})_s$ and $(\vec{\imath}_1)_s,\ldots,(\vec{\imath}_{k-1})_s$ are distinct, for a total error of $mk^{(m-1)(k-1)}\epsilon_{rej}$. ■

4.3 Two-alternating randomized prover proof systems

The power of *cooperating*-prover proof systems, is unchanged if the provers are allowed to choose randomized strategies. In this section we show that the situation is different for competing-prover proof systems.

A *two-alternating randomized-prover proof system*, denoted 2-ARP, is one where the provers have access to private coins. The same conventions that apply to two-alternating-prover proof system will be adopted for two-alternating randomized-prover proof systems.

The main result of this section shows that unless P = NP, a two-alternating randomized-prover proof system with constant error is strictly weaker than a two-alternating-prover proof system with constant error. Intuitively, in a two-alternating randomized-prover proof system the first prover can use his private coins to 'hide' his strategy from the second prover. The second prover cannot then figure out the exact intentions of the first prover. He cannot devise as good a strategy against this type of adversary as opposed to when the first prover acts according to a deterministic strategy. Thus, the second prover is of less use to the verifier, and consequently the whole proof system is weakened.

Lemma 4.3.1 For every constant ϵ, $0<\epsilon<1/2$, a language L is in P, if and only if, L has a two-alternating randomized-prover proof system with parameters $(O(\log(n)), O(\log(n)), [\exists,\forall])$ and error ϵ.

Proof: The forward direction is trivial, since the verifier alone, without the help of the provers, can decide languages in P.

We now prove the converse. Let

— $D_i(q,a)$ be the probability that prover $\mathbf{P}_i$ responds to q with answer a,

— Q_i be the set of possible questions to prover $\mathbf{P}_i$, and A_i be the set of possible responses of prover $\mathbf{P}_i$,

— $D_i = (\, D_i(q,c) : q \in Q_i, a \in A_i \,)$,

— R be the set of random strings that the verifier may generate on input ω,

— π_r be the probability that the verifier generates random string r,

— $q^{(i)}(r)$ be the question that the verifier sends to prover $\mathbf{P}_i$ on random string r,

— $V(r,a_0,a_1)$ be 1 if the verifier V accepts on input ω, random string r and given responses a_0 from $\mathbf{P}_0$ and a_1 from $\mathbf{P}_1$. Otherwise, let $V(r,a_0,a_1)$ be 0.

Feige and Lovász [FL92] have shown that the probability of acceptance of a two-cooperating-prover proof system can be computed by solving a quadratic optimization problem in max form. Applying their technique we get that the probability that the verifier V accepts on input ω is

$$\begin{aligned} &\max_{D_0} C(D_0) \\ &\text{subject to,}\ \ \forall q \in Q_0,\ \sum_{a \in A_0} D_0(q,a) = 1 \\ &\qquad\qquad\qquad \forall q \in Q_0, \forall a \in A_0,\ \ D_0(q,a) \geq 0, \end{aligned}$$

where

$$\begin{aligned} C(D_0) \overset{\text{def}}{=} &\min_{D_1} \sum_{r \in R} \sum_{a_0 \in A_0, a_1 \in A_1} \pi_r V(r,a_0,a_1)\, D_0(q^{(0)}(r),a_0)\, D_1(q^{(1)}(r),a_1) \\ &\text{subject to,}\ \ \forall q \in Q_1,\ \sum_{a \in A_1} D_0(q,a) = 1 \\ &\qquad\qquad\qquad \forall q \in Q_1, \forall a \in A_1,\ \ D_1(q,a) \geq 0. \end{aligned}$$

By the duality theorem of linear programming (see [Sch86, Ch. 7.§4]), $C(D_0)$ can be expressed as the optimum of a linear program in max form. Thus, to compute V's acceptance probability on input ω it is enough to solve a linear program of $poly(|\omega|, |A_0|, |A_1|, |R|)$ size. Since in our case $|A_0|, |A_1|, |R| = 2^{O(\log |\omega|)}$, and linear programming is polynomial-time solvable, the lemma follows. ■

An analogous result for EXP, for the only if part, was independently obtained in [FKS95].

Corollary 4.3.2 For every constant ϵ, $0 < \epsilon < 1/2$, a language L is in P, if and only if, L has a two-alternating randomized prover proof system with error ϵ.

But, in Corollary 4.2.4 we showed that languages in NP have two-alternating-prover proof systems with constant error. Hence, 2-APP $\neq$ 2-ARP, unless P = NP.

4.4 Alternating-oracle proof systems for Σ_k^P

Feige and Kilian's proof of Theorem 4.2.2 is an amplification of the main result of [ALM+92], which states that NP = PCP($\log n, 1$). In our terminology, this says that NP languages have one-alternating-oracle proof systems with error $(0, \epsilon)$, where ϵ can be an arbitrarily small positive constant. In order to extend our results beyond NP, we will need analogous tools that we can apply to languages in Σ_k^P. Thus, we begin by showing that languages in Σ_k^P have k-alternating-oracle proof systems.

We consider from now on only the case in which k is odd. Our results have analogous statements for even k.

The next theorem is implicit in [CFLS94].

Theorem 4.4.1 (k odd) For every constant $\epsilon > 0$, if a language L is in Σ_k^P, then L has a k-alternating-oracle proof system with error $(0, \epsilon)$.

In the proof of this theorem, we will make use of the fact that there are *encodings*, $E(\cdot)$, that associate to a string x, a string $E(x)$ such that the following holds:

— the length of $E(x)$ is polynomial in the length of x.

— there is a polynomial-time algorithm that on input x returns $E(x)$.

— there is a constant ϵ_E and a polynomial-time algorithm D_E such that if y and $E(x)$ differ in at most an ϵ_E fraction of their bits, then $D_E(y) = x$ (in which case we say that x and y are *closer than* ϵ_E). Otherwise, $D_E(y)$ outputs FAILURE (in which case, we say that y is *farther than* ϵ_E from any codeword).

Encodings with this property exist (e.g., Justesen encodings [MS77, Ch. 10.§11]).

Proof of Theorem 4.4.1: Let L be a language in Σ_k^P. That is, there exists a polynomial-time Turing machine M such that $\omega \in L$ if and only if $\exists X_1, \forall X_2, \ldots, \exists X_k\ M(\omega, X_1, \ldots, X_k)$ accepts [CKS81]. As in [CFLS94], we will view the acceptance condition as a game between an $\exists$ player and a $\forall$ player who take turns writing down polynomial-length strings X_i, with the $\exists$ player writing on the odd rounds.

In our k-alternating-oracle proof system, the player who writes in round i purports to write down an encoding $E(X_i)$ of X_i. In addition, in the k-th round, the $\exists$ player is to write down encodings of everything the $\forall$ player said and a PCP proof that $M(\omega, X_1, \ldots, X_k)$ accepts. If each player actually wrote down codewords, then, using the techniques from [ALM$^+$92], the verifier would read a constant number of random bits from each oracle and a constant number of bits to check the PCP proof and accept if M on input $(\omega, X_1, \ldots, X_k)$ would have accepted, or reject with high probability if M would have rejected.

In reality, one of the players will be trying to cheat and will have little incentive to write codewords. Let Y_i denote the oracle that is written in the i-th round. If a player writes an oracle Y_i that is within ϵ_E of a codeword, then the other player will proceed as if Y_i was that codeword. On the other hand, if a player writes an oracle Y_i that is farther than ϵ_E from a codeword, then with high probability the verifier will detect that player's perfidy.

In the last round, the $\exists$ player writes down strings Z_i which he claims are encodings of Y_i, for each even i. In addition, the $\exists$ player will, for each bit of each oracle Y_i, provide a PCP proof that, when decoded, Z_i agrees with Y_i on that bit. For each even i, the verifier will test these PCP proofs to check that $D_E(Z_i)$ agrees with Y_i on some constant number of randomly chosen bits. If any of these tests fails, then the verifier will know that the $\exists$

player has cheated and will reject accordingly. If the Z_i's pass all the tests, then the verifier will be confident that $D_E(Z_i)$ is close to Y_i, for all even i. The verifier then accepts if the last player's proof indicates either that,

— $D_E(Y_i) =$ FAILURE for some even i, or

— $M(\omega, X_1, D_E(Y_2), \ldots, D_E(Y_{k-1}), X_k)$ accepts (note that the PCP proof refers to the X_i's for odd i through their encoding as Y_i, and to the Y_i's for i even through their encoding as Z_i).

To see why this protocol works, assume that $\omega \in L$. In this case, the $\exists$ player will always write codewords. Moreover, regardless of what oracle Y_i the $\forall$ player writes in turn i, the exist player will write $Z_i = E(Y_i)$. Thus, the Z_i's will always pass the consistency tests. If one of the $\forall$ player's oracles is farther than ϵ_E from a codeword, then the last player will include this fact in his proof, and the verifier will reject. On the other hand, if for each even i, Y_i is close to some codeword X_i, then the application of D_E to Y_i will result in Y_i being treated as the encoding of X_i in the $\exists$ player's PCP proof.

If $\omega \notin L$, then the $\exists$ player will have to cheat in order to win. If the $\exists$ player writes a message Y_i that is not close to a unique codeword, then this will be detected with high probability when the verifier checks the validity of the PCP proof supplied in the last round. If one of the Z_i's misrepresents the oracle, $Y_i = E(X_i)$, of a $\forall$ player, then either $D_E(Z_i)$ and Y_i will have to differ in at least an ϵ_E fraction of their bits, or the $\exists$ player will have to falsify the certification of the computation of D_E on Z_i so that it does not decode to X_i. In either case, the $\exists$ player will be caught with high probability. ■

We note that Theorem 4.4.1 exactly characterizes Σ_k^P because an alternating-Turing machine with k alternations can guess the k oracles and then compute the acceptance probability of the k-alternating-oracle proof system.

Using Theorem 4.4.1 and the standard technique of [FRS88] for simulating an oracle by a pair of provers, we can transform a k-alternating-oracle proof system into a $(k+1)$-alternating-prover proof system.

Corollary 4.4.2 (k is odd) Let $k = 2t+1$. Then, L is in Σ_k^P, if and only if, L has a $(k+1)$-

alternating-prover proof system with parameters $([(\exists,\forall)^t, \exists, \exists])$ and error $\left(0, 1-\frac{1}{N}\right)$, where N is a large constant depending on k.

To prove Theorem 4.1.3, we have to reduce the error in Corollary 4.4.2 and show how to change the quantifiers from $([(\exists,\forall)^t, \exists, \exists])$ to $([(\exists,\forall)^t, \exists, \forall])$. In other words, we want an analogue of the parallel repetition theorem and of Lemma 4.2.1 which apply to alternating-prover proof systems. In the next two sections we describe how these goals can be achieved.

4.5 Feige and Kilian lemmas

In Section 4.6, we prove an analogue of the weak version of the parallel repetition theorem of Feige and Kilian [FK94] that applies to k-alternating-prover proof systems. The techniques used in [FK94] provide insight into how a few variables influence the value of a multi-variate function. In order to prove our analogue of their theorem, we need a better understanding of some of their results. Thus, we discuss them in this section.

Consider a prover **P** to which we send m randomly chosen questions $(q(r_1), \ldots, q(r_m))$. The prover answers $(a_1, \ldots, a_m)$ according to a fixed strategy $f = (f_1, \ldots, f_m)$, a function from m-tuples (the m-tuple of questions that **P** receives), to m-tuples (the m-tuple of answers that **P** responds with). We would like **P** to use a global strategy, f, in which each f_i is only a function of the question on *run* i. We say that such a prover behaves *functionally*. We now discuss the consequences of a prover's failure to behave in this way.

Let $\tilde{I}$ be a subset of $\{1, \ldots, m\}$ and $\tilde{Q} = (q(r_i) : i \in \tilde{I})$ be the questions indexed by $\tilde{I}$. Now, suppose we knew both $\tilde{I}$ and $\tilde{Q}$, then, if the prover behaves functionally we can determine $\tilde{A} = (a_i : i \in \tilde{I})$. But, what if the prover does not behave functionally, but 'almost functionally'? To answer this we need to formalize what 'almost functionally' means: say that an f-*challenge*[6] $(\tilde{I}, \tilde{Q})$ is *live* if there is an $\tilde{A}$ such that

$$\Pr\left[f_i(q(r_1), \ldots, q(r_m)) = a_i, \text{ for all } i \in \tilde{I}\right] \;\geq\; \zeta,$$

where the probability is taken over the choices of $(r_i : i \notin \tilde{I})$, and ζ is a parameter to

[6] We omit f whenever it is clear from context.

be fixed later. If the above inequality holds, say that $\tilde{A}$ is a *live answer* for the challenge $(\tilde{I}, \tilde{Q})$. Intuitively, if we know the questions $\tilde{Q}$ that **P** receives on runs $\tilde{I}$ and that $\tilde{A}$ is not a live answer for the challenge $(\tilde{I}, \tilde{Q})$, we should not be willing to bet that **P** will answer $\tilde{A}$ on runs $\tilde{I}$, because, $\tilde{A}$ has too small a chance of occurring. If **P** behaves functionally, then every challenge $(\tilde{I}, \tilde{Q})$ is live, since the questions that **P** is sent on a run completely determines his answer on that run.

For every choice of parameters $\epsilon, \zeta, \gamma, n, m$, and M such that $8\epsilon < \zeta^5$, $\zeta^{-5} < \gamma n$, $0 < \gamma < 1$, $M = m - n$, $\epsilon \geq \max\left\{\frac{256 \ln M}{M}, \left(\frac{8 \ln M}{M}\right)^{1/3}\right\}$ (note that if ϵ is very small, then $m \gg n$), the following lemmas are implicit in the work of Feige and Kilian:

Lemma 4.5.1 (Feige-Kilian) Let $j \in \{0, \ldots, n\}$. If $\tilde{A} = (a_i : i \in \tilde{I})$ is not a live answer for the challenge $(\tilde{I}, \tilde{Q})$, where $\tilde{Q} = (q(r_i) : i \in \tilde{I})$ and $\tilde{I} = \{i_1, \ldots, i_j\}$, then, with probability at least $1 - n\epsilon$ (over the choices of $I \setminus \tilde{I} = \{i_{j+1}, \ldots, i_n\}$ and $\{r_i \mid i \in I \setminus \tilde{I}\}$, where $I = \{i_1, \ldots, i_n\} \subset \{1, \ldots, m\}$) it holds that

$$\Pr\left[f_i(q(r_1), \ldots, q(r_m)) = a_i, \text{ for all } i \in \tilde{I}\right] \;<\; \zeta + (n - j)\epsilon,$$

where the probability is taken over the choices of $(r_i : i \notin I)$.

Informally, the preceding lemma says that if $\tilde{A}$ is not a live answer for the challenge $(\tilde{I}, \tilde{Q})$ and if we knew the questions sent to **P** in runs $I \supseteq \tilde{I}$, we usually should still not be willing to bet that **P** will answer $\tilde{A}$ in runs $\tilde{I}$. Note however, that for this to be true we assume that $|I|$ is a small fraction of all the runs.

Lemma 4.5.2 (Feige-Kilian) There is a good j for $f = (f_1, \ldots, f_m)$, $j < \gamma n$, such that if more than a ζ fraction of the challenges $(\tilde{I}, \tilde{Q})$ (over the choices of $\tilde{I} = \{i_1, \ldots, i_j\}$ and $(r_i : i \in \tilde{I})$) are live, then for a $(1 - \zeta)$ fraction of the live challenges $(\tilde{I}, \tilde{Q})$, for every $i \notin \tilde{I}$, and for each live answer $\tilde{A}$ for the challenge $(\tilde{I}, \tilde{Q})$, there is a function $F_{f,i,\tilde{Q},\tilde{A}}$ such that

$$\Pr\left[\left|\{i \in I \setminus \tilde{I} \mid f_i(q(r_1), \ldots, q(r_m)) \neq F_{f,i,\tilde{Q},\tilde{A}}(q(r_i))\}\right| \leq \frac{2\zeta}{\delta}\left|I \setminus \tilde{I}\right|\right] \;\geq\; 1 - \delta,$$

where the probability is taken over the choices of $(r_i : i \notin \tilde{I})$ and $I \setminus \tilde{I} = \{i_{j+1}, \ldots, i_n\}$.

The previous lemma associates to every strategy of prover **P** a nonnegative integer. For such integer j, if a non-negligible fraction of the challenges $(\tilde{I}, \tilde{Q})$ are live, where $|\tilde{I}| = j$, and if **P** answers the challenge $(\tilde{I}, \tilde{Q})$ with live answer $\tilde{A}$, then, **P** behaves almost functionally in a significant fraction of the runs $I \setminus \tilde{I}$, for most choices of the live challenges $(\tilde{I}, \tilde{Q})$, and of the set $I \setminus \tilde{I}$.

4.6 Alternating-prover proof systems for Σ_k^P

In Section 4.4 we characterized Σ_k^P in terms of alternating-prover proof systems with one sided but large rejection error and where the quantifiers associated to the provers, *except* the last one, alternated beginning with the existential quantifier. The goal of this section is to provide alternating-prover proof systems for Σ_k^P languages which in addition achieve:

— Low error rates, and

— The quantifiers associated to *all* the provers alternate.

To achieve these goals we follow an approach similar to the one taken in the obtaining of two-alternating-prover proof systems for NP (Section 4.2). We again restrict our discussion to the case $k = 2t + 1$, i.e. k is odd. Our results have analogue statements for even k.

First, we prove that languages that have a $(k + 1)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \exists, \exists])$ and one sided but large rejection error can be recognized by a $(k + 3)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \forall, \exists, \exists, \exists])$ and two sided but arbitrarily small constant error.[7] Notice that we achieve this without increasing the number of alternations of the quantifiers associated to the provers. This step is an analogue of the weak version of the parallel repetition theorem of Feige and Kilian [FK94] that applies to competing-prover proof systems.

We then convert the proof systems obtained in the first step into $(k + 1)$-alternating-prover proof systems in which all the quantifiers associated to the provers alternate and where the error is small. This latter step is an analogue of Lemma 4.2.6.

[7] In fact, a $(k + 2)$-APP system with parameters $([(\exists, \forall)^t, \forall, \exists, \exists])$ and similar error rates suffices, but proving this would unnecessarily complicate our exposition.

4.6.1 Achieving low error rates

We now informally describe a simulation protocol that achieves the first goal of this section. We consider a $(k+1)$-alternating-prover proof system with parameters $([(\exists,\forall)^t,\exists,\exists])$ and one sided but large rejection error, with verifier $\widehat{V}$ and provers $\widehat{\mathbf{P}}_0,\ldots,\widehat{\mathbf{P}}_k$. We simulate this proof system with a $(k+3)$-alternating-prover proof system where the error is low. The verifier of this latter proof system is denoted by V, and its provers by $\mathbf{P}_0,\ldots,\mathbf{P}_k$, and $\mathbf{P}_\exists$, $\mathbf{P}_\forall$. Prover $\mathbf{P}_i$ is quantified as $\widehat{\mathbf{P}}_i$ is. Moreover, $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ are quantified by $\exists$ and $\forall$ respectively. The ordering of the provers is such that $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ follow immediately after the last prover among $\mathbf{P}_0,\ldots,\mathbf{P}_k$ which is quantified by $\exists$ and $\forall$ respectively. Below we illustrate the relation among the old and new provers.

$$\begin{array}{ccccc}
\underbrace{\overbrace{\widehat{\mathbf{P}}_0}^{\exists}} & \cdots & \underbrace{\overbrace{\widehat{\mathbf{P}}_{k-3}}^{\exists}}\underbrace{\overbrace{\;\widehat{\mathbf{P}}_{k-2}\;}^{\forall}} & & \underbrace{\overbrace{\widehat{\mathbf{P}}_{k-1}\widehat{\mathbf{P}}_k}^{\exists}} \\
\downarrow & \cdots & \downarrow \quad\quad \downarrow & & \downarrow \\
\mathbf{P}_0 & \cdots & \mathbf{P}_{k-3}\;\mathbf{P}_{k-2}\mathbf{P}_\forall & & \mathbf{P}_{k-1}\mathbf{P}_k\mathbf{P}_\exists
\end{array}$$

The verifier V concurrently simulates many rounds of $\widehat{V}$'s protocol. We refer to each of these rounds as a *run*. The verifier V expects provers to behave functionally, i.e. to answer each of their questions according to a function that does not depend on the other questions they receive. Cheating provers have no incentive to comply with V's expectations. In order to penalize them for not doing so V implements the CONSISTENCY TEST. This test requires two (one for each competing team of provers) additional provers ($\mathbf{P}_\exists$ and $\mathbf{P}_\forall$). If a team of provers fails this test, their claim is rejected. Honest provers behave functionally. Thus, they always pass the CONSISTENCY TEST.

Cheating provers pass the CONSISTENCY TEST with a significant probability, only if they behave 'almost functionally'. But, this is not a guarantee that honest provers can determine these functional strategies. It is important to address this issue when designing protocols for alternating-prover proof systems as opposed to cooperating-prover proof systems.

The protocol implemented by V allows honest provers to make a set of 'educated' guesses regarding the strategies that cheating provers use. In fact, V allows $\mathbf{P}_1$ to make constantly many different guesses regarding the strategy that $\mathbf{P}_0$ uses to answer his questions. Either $\mathbf{P}_0$ has a significant probability of failing the CONSISTENCY TEST or $\mathbf{P}_1$ has a significant probability of making the right guess among its many guesses. Prover $\mathbf{P}_1$ fixes, for each guess, a strategy for answering the questions he receives. The verifier V expects that each strategy that $\mathbf{P}_1$ fixes corresponds to a functional behavior. But, if $\mathbf{P}_1$ is dishonest, he has no incentive in doing so. Hence, the verifier also allows $\mathbf{P}_2$ to make constantly many 'educated' guesses for each of the guesses that $\mathbf{P}_1$ makes, and so on and so forth.

At the end of the protocol, V can determine which of the provers' guesses are correct. With high probability, either cheating provers fail the CONSISTENCY TEST, or V can determine, among the correct guesses, a set of runs most of which can be treated as independent executions of $\widehat{V}$'s protocol. This suffices for achieving our error reduction goals.

We will now describe the precise protocol implemented by the verifier V. We then informally discuss the motivation behind the main components of the protocol.

THE QUERIES

— V chooses $I \subset \{1, \ldots, m\}$, $|I| = n$,[8] at random, and selects for every $i \in I$ a random string r_i as the old verifier $\hat{V}$ would have. The runs indexed by I will be called the *compare* runs, the other runs will be called *confuse* runs.

— For every confuse run i and every prover $\mathbf{P}_l$ the verifier V selects random string $r_i^{(l)}$ as the old verifier $\hat{V}$ would have.

For $i \in \{1, \ldots, m\}$, $l \in \{0, \ldots, k\}$ let $\rho_i^{(l)} = r_i$ if i is a compare run, $r_i^{(l)}$ otherwise.

— For every prover $\mathbf{P}_l$ the verifier V chooses $I_l \subset I \setminus \bigcup_{j<l} I_j$, $|I_l| = \gamma n$, at random together with a random ordering π_l of I_l for $l \in \{0, \ldots, k\}$.[9]

— V sends to $\mathbf{P}_l$, $l \in \{0, \ldots, k\}$, $\left(q^{(l)}(\rho_i^{(l)}) : i \in \{1, \ldots, m\} \setminus \bigcup_{j<l} I_j\right)$.[10]

— V sends to $\mathbf{P}_{l+1}$, $l \in \{0, \ldots, k-1\}$ all questions $\left(q^{(j)}(r_i) : i \in I_j\right)$, the set of indices I_j, and the orderings π_j, for all $j \in \{0, \ldots, l\}$. (We will see that this helps honest provers to make educated guesses about the strategies used by cheating provers that pass the CONSISTENCY TEST with a significant probability.)

— To implement the CONSISTENCY TEST, V sends to $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ the questions $\left(q^{(l)}(r_i) : i \in I_l\right)$, the set of indices I_l and the orderings π_l, for all $l \in \{0, \ldots, k\}$.

The format of each prover's answer is a graph, more precisely a rooted tree.

For a rooted tree T, with node set $V(T)$ and edge set $E(T)$, we say that a node $v \in V(T)$ is at level i if its distance to the root of T is i. We say that an edge $e \in E(T)$ is at level i if it is rooted at a node of level i.

Let J be a set of indices. We say that a tree T is a J-*tree* if nodes at level $j \in J$ have only one child. We consider trees whose nodes and edges are labeled. Internal nodes will be labeled by sets of indices. Edges rooted at a node labeled J will be labeled by a tuple of strings $(a_j : j \in J)$. Leaves will also be labeled by a tuple of strings. We denote the label of a node v by $\mathcal{L}_T(v)$, and the label of the edge e by $\mathcal{L}_T(e)$.

[8] We later set $m \gg n$.
[9] We later set $\gamma = \frac{1}{2(k+1)}$.
[10] When a question is sent to a prover the verifier indicates the run to which the question corresponds.

Definition 4.6.1 Let v be a node of level l of tree T with root v_0. Let $v_0, \ldots, v_{l-1}$ be the sequence of nodes along the path from the root of T to v. Define the history of v as follows: $\mathcal{H}_T(v) \overset{\text{def}}{=} [\mathcal{L}_T(v_0), \mathcal{L}_T(v_0, v_1)] \cdots [\mathcal{L}_T(v_{l-1}), \mathcal{L}_T(v_{l-1}, v)]$. If v is the root of T we say that its history is empty and we denote it by $\emptyset$.

THE RESPONSES

— Prover $\mathbf{P}_l$ responds with a tree T_l, $\mathbf{P}_\exists$ with a tree $T_\exists$, and $\mathbf{P}_\forall$ with a tree $T_\forall$.

— T_l, for $l \in \{0, \ldots, k\}$ is a tree of depth l labeled as follows:

- An internal node v at level $j \in \{0, \ldots, l-1\}$ is labeled by I_j. Every internal node v at level j has an edge rooted at v for every possible tuple of answers of $\mathbf{P}_j$ to questions $\left(q^{(j)}(r_i) : i \in I_j\right)$, and labeled by this tuple of answers.
- A leaf is labeled either FAILURE, SUCCESS, or by a tuple of answers $\left(a_i^{(l)} : i \in \{1, \ldots, m\} \setminus \bigcup_{j<l} I_j\right)$. If $\mathbf{P}_l$ is honest, he labels the leaves of T_l as shown later under HONEST PROVERS PROCEDURE FOR LABELING LEAVES.

— $T = T_\exists$ (respectively $T = T_\forall$) is a K-tree of depth $k+1$, where $K = \{0, 2, \ldots, k-1, k\}$ (respectively $K = \{1, 3, \ldots, k-2\}$), labeled as follows:

- Internal nodes are labeled as the nodes of the trees described above. Edges at levels $l \notin K$ are also labeled as in the trees described above.
- The only edge of level $l \in K$ rooted at node v of T is labeled by a tuple of answers $\left(a'^{(l)}_i : i \in I_l\right)$. An honest prover sets $a'^{(l)}_i = a_i^{(l)}$ for all $i \in I_l$, where $\left(a_i^{(l)} : i \in \{1, \ldots, m\} \setminus \bigcup_{j<l} I_j\right)$ is the label of the leaf of T_l with history $\mathcal{H}_T(v)$. (Note that T belongs to T_l's team.)

Honest Provers Procedure for Labeling Leaves

Assume $\mathbf{P}_l$ is honest, $l \in \{0, \ldots, k\}$, then $\mathbf{P}_l$ will generate a tree T_l of the proper format. Consider the leaf u_l of T_l. Let $u_0, \ldots, u_l$ be a path in T_l from the root u_0 to the leaf u_l. In labeling u_l, $\mathbf{P}_l$ considers the strategies $f_0, \ldots, f_{l-1}$ that $\mathbf{P}_0, \ldots, \mathbf{P}_{l-1}$ use in labeling the leaves (abusing notation) $u_0, \ldots, u_{l-1}$ of $T_0, \ldots, T_{l-1}$ with history $\mathcal{H}_{T_l}(u_0), \ldots, \mathcal{H}_{T_l}(u_{l-1})$ respectively. For $s \in \{0, \ldots, l-1\}$, let j_s be the smallest good j, as defined by Lemma 4.5.2 (hence $j < \gamma n$), for f_s. Let $\widetilde{I}_s \subseteq I_s$ be the set of the first j_s indices of I_s as determined by π_s. Let $\widetilde{Q}_s = (q^{(s)}(r_i) : i \in \widetilde{I}_s)$ be the set of questions on runs $\widetilde{I}_s$, and let $\widetilde{A}_s = (\widetilde{a}_i^{(s)} : i \in \widetilde{I}_s)$ be the set of answers in runs $\widetilde{I}_s$ induced by the label of the leaf u_s. Define the following events:

Event E_1 : $\forall s \in \{0, \ldots, l-1\}$, the answer sets $\widetilde{A}_s$ are live for the f_s-challenge $(\widetilde{I}_s, \widetilde{Q}_s)$.

Event E_2 : $\forall s \in \{0, \ldots, l-1\}$, more than a ζ fraction of the f_s-challenges $(\widetilde{I}_s, \widetilde{Q}_s)$ (over the choices of $\widetilde{I}_s = \{i_1, \ldots, i_{j_s}\}$ and $(r_i : i \in \widetilde{I}_s)$) are live.

For an appropriate choice of parameters we can distinguish two cases,

Events E_1 **and** E_2 **occur:** $\forall i \in \{1, \ldots, m\} \setminus \bigcup_{s<l} I_s$, $\forall s \in \{0, \ldots, l-1\}$, $\mathbf{P}_l$ determines (if possible) the functions $F_{f_s,i,\widetilde{Q}_s,\widetilde{A}_s}$ (as in Lemma 4.5.2), and the optimal strategy for answering run i, say $P_i^{u_l}$, of the $(l+1)$-th prover $\widehat{\mathbf{P}}_l$ assuming that the first l provers strategies are $F_{f_0,i,\widetilde{Q}_0,\widetilde{A}_0}, \ldots, F_{f_{l-1},i,\widetilde{Q}_{l-1},\widetilde{A}_{l-1}}$. If $\mathbf{P}_l$ is unable to determine some $F_{f_s,i,\widetilde{Q}_s,\widetilde{A}_s}$ he labels u_l with Failure, that is, $\mathbf{P}_l$ recognizes that cheating provers have outsmarted him. Otherwise, $\mathbf{P}_l$ answers run i of u_l according to $P_i^{u_l}$.

Otherwise: $\mathbf{P}_l$ labels u_l with Success, that is, $\mathbf{P}_l$ expresses its confidence that the cheating provers will not pass the Consistency test.

The verifier will reject the claim of any team of provers that fails to respond in the proper format. Thus, we henceforth focus on the case where $T_0, \ldots, T_k$ and $T_\exists, T_\forall$ are in the proper format. We are particularly interested on a specific quantity associated to l, for every $l \in \{0, \ldots, k\}$. This quantity will be denoted by $\mathcal{H}(T_0, \ldots, T_l)$ and is recursively defined. Indeed, if v_l is the leaf of T_l with history $\mathcal{H}(T_0,\dots,T_{l-1})$ and label $\left(a_i^{(l)} : i \in \{1,\dots,m\} \setminus \bigcup_{j<l} I_j\right)$ then $\mathcal{H}(T_0,\dots,T_l) \stackrel{\text{def}}{=}$

$$\mathcal{H}(T_0,\dots,T_{l-1})\left[I_l,\left(a_i^{(l)} : i \in I_l\right)\right] = \left[I_0,\left(a_i^{(0)} : i \in I_0\right)\right]\cdots\left[I_l,\left(a_i^{(l)} : i \in I_l\right)\right].$$

Note that there is only one leaf in $T_\exists$ and another in $T_\forall$ both of which have the same history. This history determines a common branch of $T_\exists$ and $T_\forall$ which we denote by

$$\mathcal{H}'(T_\exists,T_\forall) \stackrel{\text{def}}{=} \left[I_0,\left(a_i'^{(0)} : i \in I_0\right)\right]\cdots\left[I_k,\left(a_i'^{(k)} : i \in I_k\right)\right].$$

Acceptance Criteria

— If a prover does not comply with the proper format of the responses, then, the verifier V rejects the claim of the team to which he belongs. (Honest provers will always answer in the proper format.)

— The verifier V implements the following Consistency Test: V determines the largest $l \in \{0,\dots,k+1\}$ such that $\mathcal{H}(T_0),\dots,\mathcal{H}(T_0,\dots,T_{l-1})$ are prefixes of $\mathcal{H}'(T_\exists,T_\forall)$. Two different situations may arise:

- If $l > k$, then we say that the consistency test PASSES.
- Otherwise, the verifier rejects the claim of the team to which prover $\mathbf{P}_l$ belongs.

— If the previous test passes then, V accepts, if and only if, for more than a $1-\frac{1}{cN}$ fraction of the runs $i \in I \setminus \bigcup_{j\leq k} I_j$ the old verifier $\widehat{V}$ on random string r_i would accept if given answers $a_i^{(0)},\dots,a_i^{(k)}$ from provers $\widehat{\mathbf{P}}_0,\dots,\widehat{\mathbf{P}}_k$ respectively.[11]

It is important to note that from the point of view of the verifier V there is only one branch in each of the trees $T_0,\dots,T_k$ and $T_\exists,T_\forall$ which is of relevance.

The idea behind the above described protocol is the following: the verifier V wants to simulate $n/2$ rounds of $\widehat{V}$'s protocol. In order to do this, V generates n sets of questions as $\widehat{V}$ would have and sends them to the first k provers. Half of these questions will be used to

[11] Where c is a large constant whose value we will set later.

implement the CONSISTENCY TEST with the aid of the two last provers. Moreover, these n questions are sent to the first k provers among a much larger set of $m - n$ questions, for a total of m questions. This latter type of questions are called *confuse questions*, since their purpose is to confuse cheating provers. The provers do not know which questions are/are not confuse questions. There is no dependency whatsoever between the confuse questions sent to different provers. If cheating provers answer using a global strategy, then their responses will depend on the confuse questions. This will reduce their probability of passing the CONSISTENCY TEST because the last two provers do not receive the confuse questions. We are left to argue what happens when the provers have a non-negligible probability of passing the CONSISTENCY TEST. The description of the provers' responses show how honest provers can choose their strategies. The next lemma establishes that the parameters of the above described simulation protocol can be chosen so the final error is small, and the blowup in the size of the provers' responses and the overall number of random bits used by the verifier is a constant factor.

Lemma 4.6.2 (k odd) Let $k = 2t + 1$ and $0 < \alpha < 1/2$. Every language L recognized by a $(k+1)$-alternating-prover proof system with parameters $([(\exists,\forall)^t,\exists,\exists])$ and error $\left(0, 1 - \frac{1}{N}\right)$, where N is a large constant, has a $(k+3)$-alternating-prover proof system with parameters $([(\exists,\forall)^t,\forall,\exists,\exists,\exists])$ and error α.

Proof: Denote by $\widehat{V}$ the $(k+1)$-alternating-prover proof systems' verifier, and by $\widehat{\mathbf{P}}_0,\ldots,\widehat{\mathbf{P}}_k$ its provers. Moreover, let V denote the verifier which interacts with provers $\mathbf{P}_0,\ldots,\mathbf{P}_k$ and $\mathbf{P}_\exists,\mathbf{P}_\forall$ according to the simulation protocol previously described. Clearly, the latter proof system is a $(k+3)$-alternating-prover one-round proof system where the provers are quantified according to $([(\exists,\forall)^t,\forall,\exists,\exists,\exists])$. We have to show that there is an appropriate choice of parameters for which this proof systems achieves error α, uses logarithmic number of random bits, and where the provers' responses are constant size.

First, note that honest provers never fail the CONSISTENCY TEST.

Now, let f_0 be the strategy that prover $\mathbf{P}_0$ uses in labeling the only leaf of T_0. Moreover, let $f_1,\ldots,f_k$ be the strategies that $\mathbf{P}_1,\ldots,\mathbf{P}_k$ use in labeling the leaves $v_1,\ldots,v_k$ of $T_1,\ldots,T_k$ with history $\mathcal{H}(T_0),\ldots,\mathcal{H}(T_0,\ldots,T_{k-1})$ respectively. For $s \in \{0,\ldots,k\}$, let j_s

be the smallest good j, as defined by Lemma 4.5.2, for f_s. Let $\tilde{I}_s \subseteq I_s$ be the set of the first j_s indices of I_s as determined by π_s. Let $\tilde{Q}_s = (q^{(s)}(r_i) : i \in \tilde{I}_c)$ be the set of questions on runs $\tilde{I}_s$, and let $\tilde{A}_s = (\tilde{a}_i^{(s)} : i \in \tilde{I}_s)$ be the set of answers in runs $\tilde{I}_s$ induced by the label of the leaf v_s. Finally, define the events E_1 and E_2 as follows

Event E_1 : $\forall s \in \{0,\ldots,k\}$, the answer sets $\tilde{A}_s$ are live for the f_s-challenge $(\tilde{I}_s, \tilde{Q}_s)$.

Event E_2 : $\forall s \in \{0,\ldots,k\}$, more than a ζ fraction of the f_s-challenges $(\tilde{I}_s, \tilde{Q}_s)$ (over the choices of $\tilde{I}_s = \{i_1,\ldots,i_{j_s}\}$ and $(r_i : i \in \tilde{I}_s)$) are live.

We distinguish the following cases:

If E_1 **does not occur:** then, there is an $s \in \{0,\ldots,k\}$ such that $\tilde{A}_s$ is not a live answer for the f_s-challenge $(\tilde{I}_s, \tilde{Q}_s)$. It follows that $\mathbf{P}_s$ is a cheating prover. By Lemma 4.5.1, except for a probability of at most $n\epsilon$ (over the choices of $I \setminus \tilde{I}_s$ and $\{r_i \mid i \in I \setminus \tilde{I}_s\}$), $\mathbf{P}_s$ will pass the CONSISTENCY TEST with probability at most $\zeta + n\epsilon$. thus cheating provers pass the test with probability at most $\zeta + 2n\epsilon \leq 3\zeta$ (assuming $n\epsilon \leq \zeta$).

If E_2 **does not occur:** then, the probability that event E_1 occurs is at most ζ. The preceding case analysis bounds the probability that cheating provers will pass the CONSISTENCY TEST when event E_1 does not occur by 3ζ. Thus, if E_2 does not occur, then cheating provers will pass the CONSISTENCY TEST with probability at most 4ζ.

If E_1 **and** E_2 **occur:** then, Lemma 4.5.2 implies that with probability at most $\frac{(k+2)}{2}\zeta$ the honest provers concede FAILURE in one of the leafs $v_0,\ldots,v_k$. Otherwise, with probability at least $1 - \frac{k+2}{2}\delta$ at least $(1-(k+1)\gamma - \frac{(k+2)\zeta}{\delta})n$ of the runs $i \in I \setminus \bigcup_{j\leq k} I_j$ are *good*, i.e. they are answered according to $F_{f_0,i,\tilde{Q}_0,\tilde{A}_0},\ldots,F_{f_k,i,\tilde{Q}_k,\tilde{A}_k}$.

Let $p = \frac{(k+2)}{2}(\delta+\zeta) + 7\zeta$, and $n' = (1-(k+1)\gamma)n$. Now, choose $\gamma = \frac{1}{2(k+1)}$, $\delta = \sqrt{\zeta}$, ζ small enough so $\zeta \leq 1/(2cN(k+2))^2$ (hence $\mu = (1-(k+1)\gamma - \frac{(k+2)\zeta}{\delta})\frac{n}{n'} \geq 1 - \frac{1}{cN}$) and $p = \frac{(k+2)}{2}(\sqrt{\zeta} + \zeta) + 7\zeta \leq \alpha/2$, n large enough so $n \geq 2N\log(2/\alpha)$ and $n > 2\zeta^{-5}/\gamma$, ϵ small enough so $8\epsilon < \zeta^5$ and $n\epsilon \leq \zeta$, M large enough so if $M = m - n$, $\epsilon \geq \max\left\{\frac{256\ln M}{M}, \left(\frac{8\ln M}{M}\right)^{1/3}\right\}$.

All the above mentioned parameters are independent of the size of the input. Hence, the simulation protocol requires only a constant factor blowup in the size of the provers' responses and in the amount of randomness used by the verifier.

We refer to the runs indexed by elements of $I \setminus \bigcup_{j \leq k} I_j$ as the *relevant* runs.

Assume $\omega \in L$. Then, with probability at least $1 - p \geq 1 - \alpha/2$, a $\mu \geq 1 - \frac{1}{cN}$ fraction of the relevant runs are good runs. But, a good run would make the old verifier $\widehat{V}$ accept. Thus, a fraction of at least $1 - \frac{1}{cN}$ of the relevant runs make the verifier $\widehat{V}$ accept. Hence, V accepts with probability at least $1 - \alpha$.

Assume $\omega \notin L$. Let ρ be the fraction of good runs among the relevant runs. Recall that the verifier V will accept if more than a $1 - \frac{1}{cN}$ fraction of the n' relevant runs would cause the old verifier $\widehat{V}$ to accept. Hence, for V to accept, more than $\left(\rho - \frac{1}{cN}\right) n'$ of the $\rho n'$ good runs have to cause the old verifier $\widehat{V}$ to accept. But, a good run makes $\widehat{V}$ accept with probability at most $1 - \frac{1}{N}$. Note that the probability that $\rho < \mu$ is at most $p \leq \alpha/2$. A bound on the tail of a binomial distribution (see [CLR90, Theorem 6.2]) shows that we can choose c large enough so that with probability at most $\alpha/2$ more than $\left(\mu - \frac{1}{cN}\right) n'$ of the good runs would cause $\widehat{V}$ to accept. Hence, the overall acceptance probability of the verifier V is at most α. ∎

Remark 4.6.3 Note that the task of performing the CONSISTENCY TEST can be distributed among the provers. Provers $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ are only needed in order to insure the functional behavior of the last provers (among $\mathbf{P}_0, \ldots, \mathbf{P}_k$) that are quantified with $\exists$ and $\forall$ respectively. Indeed, the simulation protocol described in this section allows provers more leeway than necessary. In fact, V should also request that the tree T_l with which $\mathbf{P}_l$ responds be a J_l-tree, where J_l is the set of indices $j \in \{0, \ldots, l-1\}$ for which $\mathbf{P}_j$ is in the same team that $\mathbf{P}_l$ is. For every $l \in \{0, \ldots, k\}$ and $j \in J_l$, the verifier V will now check that the label of the unique edge of level j rooted at the node v of T_l is consistent with the set of answers that the leaf of T_j with history $\mathcal{H}_{T_l}(v)$ induces in runs I_j. If one of these consistency check fails, then V rejects the claim of prover $\mathbf{P}_l$'s team, where l is the smallest index for which one of the above consistency check fails.

This remark will be crucial for achieving the goals of the next section.

4.6.2 Fixing the quantifiers

We now show that the conclusion of Lemma 4.6.2 still holds even if we are restricted to using $k+1$ provers, all of whose quantifiers alternate. More precisely, we have the following:

Lemma 4.6.4 (k odd) Let $k = 2t+1$ and $0 < \epsilon < 1/2$. Every language L recognized by a $(k+3)$-alternating-prover proof system with parameters $([(\exists,\forall)^t, \forall, \exists, \exists, \exists])$ and error α for every constant α, can be recognized by a $(k+1)$-alternating-prover proof system with parameters $([(\exists,\forall)^{t+1}])$ and error ϵ.

Proof: [Sketch] Again, the proof consists in simulating the first proof system by the second one. There are three major components to this simulation. Namely, the simulation protocols in which the proofs of Theorem 4.2.2, Lemma 4.2.6, and Lemma 4.6.2 are based. We denote by $\hat{V}$ the $(k+3)$-alternating-prover proof systems' verifier, and its provers by $\hat{\mathbf{P}}_0, \ldots, \hat{\mathbf{P}}_{k-1}$ and $\hat{\mathbf{P}}'_{k-2}, \hat{\mathbf{P}}'_{k-1}, \hat{\mathbf{P}}''_{k-1}$. The provers' ordering is such that $\hat{\mathbf{P}}'_{k-2}$ follows immediately after $\hat{\mathbf{P}}_{k-2}$, and $\hat{\mathbf{P}}'_{k-1}, \hat{\mathbf{P}}''_{k-1}$ follow $\hat{\mathbf{P}}_{k-1}$ (in this order). Hence, the provers with subindex $k-2$ are quantified by $\forall$, and the ones with subindex $k-1$ are quantified by $\exists$. We will simulate this proof system by a $(k+1)$-alternating-prover-proof system with verifier V, and provers $\mathbf{P}_0, \ldots, \mathbf{P}_{k-1}, \mathbf{P}_\forall$ all of whose quantifiers alternate starting with $\exists$. We refer to the former proof system as the old proof system, and to the latter one as the new proof system. Note that an old prover with subindex i is quantified as the new prover with subindex i is. We would like to simulate an old prover with subindex i by the new prover with subindex i. Below we illustrate the relation among the old and new provers.

$$\begin{tikzcd}
\overbrace{\underbrace{\hat{\mathbf{P}}_0}_{}}^{\exists} \arrow[d] & \cdots & \overbrace{\underbrace{\hat{\mathbf{P}}_{k-3}}_{}}^{\exists} \arrow[d] & \overbrace{\underbrace{\hat{\mathbf{P}}_{k-2}\hat{\mathbf{P}}'_{k-2}}_{}}^{\forall} \arrow[d] & \overbrace{\underbrace{\hat{\mathbf{P}}_{k-1}\hat{\mathbf{P}}'_{k-1}\hat{\mathbf{P}}''_{k-1}}_{}}^{\exists} \arrow[d] & \\
{} & \cdots & {} & {} & {} & \\
\mathbf{P}_0 & \cdots & \mathbf{P}_{k-3} & \mathbf{P}_{k-2} & \mathbf{P}_{k-1} & \underbrace{\mathbf{P}_\forall}_{\forall}
\end{tikzcd}$$

The verifier V interacts with the provers as in the simulation protocol discussed in the previous section, but modified according to Remark 4.6.3 and as discussed below. We first describe the queries that V makes. For $l \leq k-1$, denote by $q^{(l)}(r)$ the question that $\hat{V}$

asks $\widehat{\mathbf{P}}_l$ on random string r. Moreover, let $q'^{(k-1)}(r)$, $q'^{(k-2)}(r)$ and $q''^{(k-2)}(r)$ denote the questions that $\widehat{V}$, on random string r, asks $\widehat{\mathbf{P}}'_{k-1}$, $\widehat{\mathbf{P}}'_{k-2}$, and $\widehat{\mathbf{P}}''_{k-2}$ respectively.

THE QUERIES

— V chooses $I \subset \{1,\ldots,m\}$, $|I| = n$, at random, and selects for every $i \in I$ a random string r_i as $\widehat{V}$ would have. The runs indexed by I will be called the *compare* runs, the other runs will be called *confuse* runs.

— For every confuse run i and every $l \leq k-2$, V selects random string $r_i^{(l)}$ as $\widehat{V}$ would have.[12] For every run i, V randomly chooses $a_i \in \{0,1\}$ and $b_i \in \{0,1,2\}$. For every run i and $l < k-1$, let $\rho_i^{(l)} = r_i$ if i is a compare run, and $r_i^{(l)}$ otherwise.

— For every $l \leq k-2$, the verifier V chooses $I_l \subset I \setminus \bigcup_{j<l} I_j$, $|I_l| = \gamma n$, at random together with a random ordering π_l of I_l.

— For every $l \leq k-3$, the verifier V sends to prover $\mathbf{P}_l$ the questions $\left(q^{(l)}(\rho_i^{(l)}) : i \in \{1,\ldots,m\} \setminus \bigcup_{j<l} I_j\right)$.

— V sends to $\mathbf{P}_{k-2}$ questions $\left(q_{a_i}^{(k-2)}(\rho_i^{(k-2)}) : i \in \{1,\ldots,m\} \setminus \bigcup_{j<k-2} I_j\right)$, where $q_{a_i}^{(k-2)}(r)$ equals $(q^{(k-2)}(r),0)$ if $a_i = 0$, and $(q'^{(k-2)}(r),1)$ otherwise.

— V sends to $\mathbf{P}_{k-1}$ questions $\left(q_{b_i}^{(k-1)}(r_i) : I \setminus \bigcup_{j<k-1} I_j\right)$, where $q_{b_i}^{(k-1)}(r)$ equals $(q^{(k-1)}(r),0)$ if $b_i = 0$, $(q'^{(k-1)}(r),1)$ if $b_i = 1$, and $(q''^{(k-1)}(r),2)$ otherwise.

— For $l \leq k-2$, the verifier V sends to $\mathbf{P}_{l+1}$ the questions that were sent to $\mathbf{P}_j$ in runs I_j, the set of indices I_j, and the orderings π_j, for all $j \leq l$.

— V sends to $\mathbf{P}_\forall$ the questions that were sent to $\mathbf{P}_l$ in runs I_l, the set of indices I_l, and the orderings π_l, for all $l \leq k-2$. V also sends the pairs of questions $\left((q^{(k-2)}(r_i), q'^{(k-2)}(r_i)) : i \in I \setminus \bigcup_{j<k-2} I_j\right)$, and the triplets of questions $\left((q^{(k-1)}(r_i), q'^{(k-1)}(r_i), q''^{(k-1)}(r_i)) : i \in I \setminus \bigcup_{j<k-1} I_j\right)$,

[12] No questions are sent to $\mathbf{P}_{k-1}$ in the confuse runs.

The format of the provers' responses, the honest provers' answers, and the verifier's acceptance criteria correspond to modifications of the simulation protocol of Section 4.6.1. We now list these modifications.

— Prover $\mathbf{P}_l$, $l \leq k-1$, responds with a J_l-tree, denoted T_l, of the form described in Remark 4.6.3.

— Prover $\mathbf{P}_\forall$ responds with two trees, $T_\exists$ and $T_\forall$. Tree $T_\forall$ is a tree of the same kind and depth as the response tree T_{k-2}. Tree $T_\exists$ is a tree of the same kind and depth as the response tree T_{k-1}.

— Provers $\mathbf{P}_0, \ldots, \mathbf{P}_{k-3}$ should respond as in the simulation protocol of Section 4.6.1.

— At each leaf of its response tree prover $\mathbf{P}_{k-2}$ is required to simulate $\widehat{\mathbf{P}}_{k-2}, \widehat{\mathbf{P}}'_{k-2}$. This can be done with the help of $\mathbf{P}_\forall$ without giving too much advantage to cheating provers. Indeed, consider the leaf v of T_{k-2}. Let P denote the strategy by which $\mathbf{P}_{k-2}$ labels v. Note that P is a function defined over pairs (q, b), where q is a question and $b \in \{0, 1\}$. Hence, P implicitly defines two strategies $P(\cdot, 0)$ and $P(\cdot, 1)$. The verifier V requires that $\mathbf{P}_\forall$ label the leaf of $T_\forall$ with history $\mathcal{H}_{T_{k-2}}(v)$ according to these two strategies. Thus, $\mathbf{P}_\forall$ reveals the answers to the questions that were not sent to $\mathbf{P}_{k-2}$. Prover $\mathbf{P}_\forall$ succeeds in misrepresenting s of the answers that $\mathbf{P}_{k-2}$ would have given with probability at most $1/2^s$.

— At each leaf of its response tree prover $\mathbf{P}_{k-1}$ is required to simulate $\widehat{\mathbf{P}}_{k-1}, \widehat{\mathbf{P}}'_{k-1}$, and $\widehat{\mathbf{P}}''_{k-1}$. Lemma 4.2.6 shows how this can be done with the help of $\mathbf{P}_\forall$. Indeed, consider the leaf v of T_{k-1}. Let P denote the strategy by which $\mathbf{P}_{k-1}$ labels v. The verifier V requires that $\mathbf{P}_\forall$ label the leaf of $T_\exists$ with history $\mathcal{H}_{T_{k-1}}(v)$ according to how the second prover of the two-alternating-prover proof system of Lemma 4.2.6 would have responded when the first provers' strategy is P.

Upon receiving the provers responses V will perform all the consistency checks suggested by the preceding discussion. If one of these checks fails, then V rejects the claim of the team found at fau t. Otherwise, as in the simulation protocol of Section 4.6.1, V only focuses in the unique branch of $T_0, \ldots, T_k$ and $T_\exists, T_\forall$ which is of relevance. The verifiers acceptance criteria corresponds to the obvious extension of the acceptance criteria of the simulation protocol in which the proof of Lemma 4.2.6 and Lemma 4.6.2 are based.

For an appropriate choice of parameters, the proof of correctness of the simulation protocol just described follows from the same arguments used to prove Theorem 4.2.2, Lemma 4.2.6, and Lemma 4.6.2. ∎

We can now prove our main result.

Proof of Theorem 4.1.3: We prove the theorem for the case where k is odd.

For the reverse direction, assume L has a $(k+1)$-prover proof system with error $\epsilon < 1/2$. Consider the alternating Turing machine that on input ω performs the following three steps: first, it uses its i-th level of alternation, $i \in \{1, \dots, k\}$, to guess the strategy of the i-th prover (i.e. guesses the constant-size answer to each of the $poly(|\omega|)$ many different question that the i-th prover might receive). Second, it computes the optimal strategy of the last prover (i.e. computes for each one of the $poly(|\omega|)$ many questions that the last prover might receive, the optimal constant-size response). Finally, it determines the probability that the verifier accepts (by simulating the verifier in each one of his $poly(|\omega|)$ many random strings), and accepts, if and only if, the verifier's acceptance probability is at least $1-\epsilon$. Clearly, this machine recognizes L, hence, $L \in \Sigma_k^P$.

The forward direction follows from Theorem 4.4.1, Lemma 4.6.2, and Lemma 4.6.4. ∎

Corollary 4.6.5 If k is a positive integer, then $\Sigma_{k-1}^P = k\text{-APP}$.

4.6.3 Comments

In this section we make some general comments and give evidence that the two sided error requirement of Theorem 4.1.3 is necessary. Indeed, unless the polynomial hierarchy collapses, all Σ_{k-1}^P languages cannot have k-alternating-prover proof systems with one sided error.

Lemma 4.6.6 Let ϵ be a constant, $0 \leq \epsilon < 1$. If languages in Σ_k^P have $(k+1)$-alternating-prover proof systems with error $(0, \epsilon)$, then the polynomial hierarchy collapses to Σ_{k-1}^P.

Proof: [Sketch] We prove the lemma for the case $k = 1$. The extension is obvious.

Let L be a language in NP, and assume it has a two-alternating-prover proof system with verifier V and provers $\mathbf{P}_0$ and $\mathbf{P}_1$ quantified by $\exists$ and $\forall$ respectively. Assume that ω is the input to the alternating-prover proof system. For $i \in \{0, 1\}$, let $q^{(i)}(r)$ be the question that V on random string r sends to $\mathbf{P}_i$, Q_i be the set of all possible question that $\mathbf{P}_i$ can receive, and A_i be the set of possible answers of $\mathbf{P}_i$. Moreover, let R the set of random strings that V can select, and let $V(\omega, r, a_0, a_1) = accept$ if V accepts on input ω, random string r, and if given response a_0 from $\mathbf{P}_0$ and a_1 from $\mathbf{P}_1$.

If the two-alternating-prover proof system under consideration has one sided error $(0, \epsilon)$, then, the following are equivalent:

— there exists a strategy P_0 for prover $\mathbf{P}_0$, such that for every strategy P_1 for prover $\mathbf{P}_1$, and for all random strings $r \in R$, $V(\omega, r, a_0, a_1) = accept$, where $a_0 = P_0(q^{(0)}(r))$ and $a_1 = P_1(q^{(1)}(r))$,

— $\forall q_0 \in Q_0$, $\exists a_0 \in A_0$, such that $\forall a_1 \in A_1$ and $\forall r \in R$ such that $q^{(0)}(r) = q_0$, $V(\omega, r, a_0, a_1) = accept$.

The first statement holds, if and only if, $\omega \in L$. Since $|A_0|, |A_1| = O(1)$, $|R|, |Q_0|, |Q_1| = 2^{O(\log |\omega|)}$, and $V(\cdot, \cdot, \cdot, \cdot)$ is polynomial-time computable, the truth value of the last statement above can be decided in polynomial time. Thus, the language L can be recognized in polynomial time. Hence, the polynomial hierarchy collapses to P. ■

Under complexity theoretic assumptions we can use Theorem 4.1.3 and Theorem 4.4.1 to prove hardness of approximation results. These results hold for either artificially constructed optimization problems or problems of only theoretical importance. We hope non-approximability results for non-artificially constructed problems will be found.

More direct proofs of the results of this chapter are desirable. Ones that avoid the Feige and Kilian lemmas. Algebraic arguments are a natural candidate for simplifying our proofs. We hope this will be accomplished in the near future.

Bibliography

[ALM$^+$92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Pittsburgh, Pennsylvania, October 1992. IEEE.

[Aro94a] S. Arora. *Probabilistic checking of proofs and hardness of approximation problems.* PhD thesis, University of California, Berkeley, 1994.

[Aro94b] S. Arora. Talk given at the *Weizmann Workshop on Probabilistic Proof Systems and Cryptography, Program Checking and Approximation Problems*, January 1994.

[Aro95a] S. Arora. Private communication, 1995.

[Aro95b] S. Arora. Reductions, codes, PCPs, and inapproximability. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 404–413, Milwaukee, Wisconsin, October 1995. IEEE.

[AS92a] N. Alon and J. H. Spencer. *The probabilistic method.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., first edition, 1992.

[AS92b] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 2–13, Pittsburgh, Pennsylvania, October 1992. IEEE.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 421–429, Providence, Rhode Island, May 1985. ACM. A subsequent journal paper covering some of this material is: L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. of Computer and System Sciences* 36(2):254-276, April 1988.

[Bab92] L. Babai. Transparent proofs and limits to approximation. In A. Joseph, F. Mignot, F. Murat, B. Prum, and R. Rentschler, editors, *Proceedings of the first European Congress of Mathematics*, volume I of *Progress in Mathematics*, pages 31–91, Paris, France, July 1992. Birkhäuser Verlag.

[BCH+95] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in chacaracteristic two. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 432–441, Milwaukee, Wisconsin, October 1995. IEEE.

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity*. Springer-Verlag, second edition, 1995.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 16–25, St. Louis, Missouri, October 1990. IEEE.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 21–31, New Orleans, Louisiana, May 1991. ACM.

[BGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 113–131, Chicago, Illinois, May 1988. ACM.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 294–304, San Diego, California, May 1993. ACM.

[BGS95] M. Bellare, O. Goldreich, and M. Sudan. Free bits and non-approximability. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 422–431, Milwaukee, Wisconsin, October 1995. IEEE.

[BK89] M. Blum and S. Kannan. Designing programs that check their work. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 86–97, Seattle, Washington, May 1989. ACM.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Selt-testing/correcting with applications to numerical problems. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 73–83, Baltimore, Maryland, May 1990. ACM.

[Blu88] M. Blum. Designing programs to check their work. Technical report, International Computer Science Institure, 1988.

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 184–193, Montréal, Québec, Canada, May 1994. ACM.

[BW94] M. Blum and H. Wasserman. Program result-checking: A theory of testing meets a test of theory. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 382–392, Santa Fe, New Mexico, November 1994. IEEE.

[CFLS93] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 305–314, San Diego, California, May 1993. ACM.

[CFLS94] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Random debaters and the hardness of approximting stochastic functions. In *Proceedings of the 9th Annual Conference on Structure in Complexity Theory*, pages 280–293, Amsterdam, The Netherlands, June 1994. IEEE.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. of the Association for Computing Machinery*, 28(1):114–133, January 1981.

[CLR90] T. H. Cormen, C. E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, first edition, 1990. Thirteenth printing, 1994.

[Con88] A. Condon. *Computational models of games.* PhD thesis, University of Washington, 1988.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 1971. ACM.

[Cop] D. Coppersmith. Manuscript.

[ERS95] F. Ergün, S. Ravikumar, and D. Sivakumar. Manuscript, December 1995.

[FGL$^+$91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 2–12, San Juan, Puerto Rico, October 1991. IEEE.

[FHS94] K. Friedl, Z. Hátsági, and A. Shen. Low-degree tests. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 57–64, Arlington, Virginia, January 1994. ACM.

[FK94] U. Feige and J. Kilian. Two prover protocols – low error at affordable rates. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 172–183, Montréal, Québec, Canada, May 1994. ACM.

[FKS95] J. Feigenbaum, D. Koller, and P. Shor. A game-theoretic classification of interactive complexity classes. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 227–237, Minneapolis, Minnesota, June 1995. IEEE.

[FL92] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 733–744, Victoria, British Columbia, Canada, May 1992. ACM.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3rd Annual Conference on Structure in Complexity Theory*, pages 156–161, Georgetown University, Washington D.C., June 1988. IEEE.

[FS95] K. Friedl and M. Sudan. Some improvements to total degree testing. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems*, pages 190–198, Tel Aviv, Israel, January 1995. IEEE.

[FST87] U. Feige, A. Shamir, and M. Tennenholtz. The noisy oracle model. In *Advances in Cryptology — CRYPTO'87*, pages 284-296, August 1987.

[GLR$^+$91] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 32–42, New Orleans, Louisiana, May 1991. ACM.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, May 1985. ACM.

[Gol94] O. Goldreich. Probabilistic proof systems (A survey). Technical Report TR94-008, Electronic Colloquium on Computational Complexity, 1994.

[Hås95] J. Håstad. Testing of the long code and hardness for clique. Manuscript, November 1995.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation.* Addison-Wesley Publishing Company, first edition, 1979.

[Joh88] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. of Algorithms*, 9(3):426–444, September 1988.

[Joh92] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. of Algorithms*, 13(3):502–524, September 1992.

[Kar72] R. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Kle94] D. J. Kleitman. Private communication, 1994.

[KLR$^+$94] M. Kiwi, C. Lund, A. Russell, R. Sundaram, and D. Spielman. Alternation in interaction. In *Proceedings of the 9th Annual Conference on Structure in Complexity Theory*, pages 294–303, Amsterdam, The Netherlands, June 1994. IEEE.

[Lev73] L. A. Levin. Universal'nyĭe perebornyĭe zadachi. *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 2–10, St. Louis, Missouri, October 1990. IEEE.

[LS91] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXP-time. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 13–18, San Juan, Puerto Rico, October 1991. IEEE.

[Lun94] C. Lund. Personal communication, August 1994.

[Mac62] F. J. MacWilliams. *Combinatorial problems of elementary group theory.* PhD thesis, Harvard University, May 1962.

[MS77] F. J. MacWilliams and N. J. A Sloane. *The theory of error-correcting codes.* Elsevier Science Publisher B.V., first edition, 1977. (Eight impression: 1993).

[Pap94] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, first edition, 1994.

[PR79] G. L. Peterson and J. H. Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 348–363, San Juan, Puerto Rico, October 1979. IEEE.

[PS94] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 194–203, Montréal, Québec, Canada, May 1994. ACM.

[PW72] W. W. Peterson and E. J. Weldon, Jr. *Error-correcting codes.* MIT Press, second edition, 1972. (Tenth printing: 1990).

[Raz95] R. Raz. A parallel repetition theorem. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 447–456, Las Vegas, Nevada, June 1995. ACM.

[Rei79] J. H. Reif. Universal games of incomplete information. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 288–308, Atlanta, Georgia, May 1979. ACM.

[RS92] R. Rubinfeld and M. Sudan. Testing polynomial functions efficiently and over rational domains. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 23–32, Orlando, Florida, January 1992. ACM.

[RS93] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials and their applications to program testing. Technical Report RC 19156, IBM, 1993.

[RS95] A. Russell and R. Sundaram. Symmetric alternation captures BPP. Technical Report MIT-LCS-TM-541, MIT, 1995.

[Rub90] R. Rubinfeld. *A mathematical theory of self-checking, self-testing and self-correcting programs.* PhD thesis, University of California, Berkeley, 1990.

[Sch86] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, first edition, 1986.

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 11–15, St. Louis, Missouri, October 1990. IEEE.

[She91] A. Shen. Manuscript, 1991.

[Sip96] M. Sipser. *Introduction to the theory of computation.* PWS Publishing Company, 1996. (Preliminary version).

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, October 1976.

[Sud92] M. Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems.* PhD thesis, University of California, Berkeley, May 1992.

[Sze75] G. Szegö. *Orthogonal polynomials*, volume 23 of *Colloquium Publications.* American Mathematical Society, fourth edition, 1975. (Reprint: 1991).

[vL90] J. van Leeuwen, editor. *Handbook of theoretical computer science*, volume A Algorithms and complexity. MIT Press/Elsevier, first edition, 1990.

[vL92] J. H. van Lint. *Introduction to coding theory*, volume 86 of *Graduate Texts in Mathematics.* Springer–Verlag, second edition, 1992.

APPENDIX A

Tightness of the linearity test lower bound (GF(2) case)

The purpose of the following discussion is to show that although the lower bounds obtained in Chapter 2 may be improved, they cannot be improved by much.

Let F be the two element field. Recall that in Section 2.7 we argued that if we consider functions $f\colon F^n \to F$ for which $\mathsf{Dist}(f) \in [0, 5/16]$, then the lower bound given in Chapter 2 is best possible. If $\mathsf{Dist}(f) \geq 5/16$, we proved that $\mathsf{Rej}(f) \geq \max\{45/128, \mathsf{Dist}(f)\}$. We now give evidence that this latter bound is not far from optimum. In fact, we show how to compute, for every $\delta \in [0, 1/2]$ and $\epsilon > 0$, a collection of real values $\{\Gamma_{k,\epsilon}(\delta) \mid k > 0\}$, such that for sufficiently large n, there exists a collection of functions $\{f_{n,k}\colon F^n \to F \mid k > 0\}$ for which $\mathsf{Dist}(f_{n,k}) \approx \delta$ and $\mathsf{Rej}(f_{n,k}) \approx \Gamma_{k,\epsilon}(\delta)$. More precisely:

$$|\,\mathsf{Dist}(f_{n,k}) - \delta\,| \leq \epsilon \qquad \text{and} \qquad |\,\mathsf{Rej}(f_{n,k}) - \Gamma_{k,\epsilon}(\delta)\,| \leq O\left(\epsilon\,|F|^k\right).$$

Table 4.6.3 compares the values of $\Gamma_{k,\epsilon}(\delta)$, for $k = 6$ and $\epsilon = 2^{-20}$, with those of $\max\{45/128, \delta\}$ for various values of $\delta \in [5/16, 1/2]$. The table entries where obtained with the aid of the optimization toolbox of MATLAB®. (The table entries where plotted in Figure 2-2, page 21.)

δ	$\max\{45/128, \delta\}$	$\Gamma_{6,2^{-20}}(\delta)$
0.3125	0.3516	0.3516
0.3229	0.3516	0.3737
0.3333	0.3516	0.3921
0.3437	0.3516	0.4107
0.3542	0.3542	0.4124
0.3645	0.3646	0.4163
0.3750	0.3750	0.4299
0.3854	0.3854	0.4347
0.3959	0.3958	0.4475
0.4063	0.4063	0.4512
0.4166	0.4167	0.4590
0.4271	0.4271	0.4679
0.4375	0.4375	0.4798
0.4479	0.4479	0.4871
0.4583	0.4583	0.4928
0.4687	0.4688	0.4966
0.4791	0.4792	0.4988
0.4896	0.4896	0.4998
0.5000	0.5000	0.5000

Table A.1: Middle column represents the lower bound that we can prove. Lower bound cannot be improved above the values in last column. (All digits shown are significant.)

We hope that the technique described below will be useful in the future in assessing the sharpness of the analyses of other tests besides the linearity test.

SECTION ORGANIZATION: First, we state the main result of this section. Before presenting its proof we state a fact that we will need and describe the underlying idea of the proof.

Lemma A.0.7 Let k be a positive integer. For every $\delta \in [0,1]$, $\epsilon > 0$, and large enough n, there is a function $f_{n,k}$: $F^n \to F$ such that

$$|\,\mathsf{Dist}(f_{n,k}) - \delta\,| \leq \epsilon \qquad \text{and} \qquad \mathsf{Rej}(f_{n,k}) \leq \Gamma_{k,\epsilon}(\delta) + O\left(\epsilon\,|F|^k\right),$$

where the hidden constant is small, $\Gamma_{k,\epsilon}(\delta) \overset{\text{def}}{=} \frac{1}{2}\left(1 - \Psi_{k,\epsilon}(\delta)\right)$, and

$$
\begin{array}{llrcl}
(\mathrm{P}_{k,\epsilon,\delta}) & \Psi_{k,\epsilon}(\delta) \overset{\text{def}}{=} & \min\limits_{\vec{x}=(x_v \,:\, v\in F^k)} \sum\limits_{\beta\in F^k} (\widehat{x}_\beta)^3 & & \\
 & & \text{subject to, } \ \widehat{x}_0 & = & 1-2\delta \\
 & & \forall\beta\in F^k,\ \widehat{x}_\beta & = & \frac{1}{|F|^k}\sum\limits_{v\in F^k} x_v(-1)^{\beta\cdot v} \\
 & & \forall\beta\in F^k\setminus\{0\},\ \widehat{x}_0 & \geq & \widehat{x}_\beta + 2\epsilon \\
 & & \forall v\in F^k\setminus\{0\},\ x_v & \in & [-1,1]\,.
\end{array}
$$

Lemma A.0.7 is based on the following fact:

Fact A.0.8 Let $f\colon F^n \to F$, and $h \equiv (-1)^f$, then $\mathrm{Rej}(f) = \frac{1}{2}\left(1 - \sum_{\alpha\in F^n} \left(\widehat{h}_\alpha\right)^3\right)$.

Proof: Implicit in the proof of Theorem 2.1.2. ■

Here is the intuition behind the proof of Lemma A.0.7: if $f\colon F^n \to F$ is such that $\mathrm{Dist}(f) = \delta$ and $\mathrm{Rej}(f)$ is small, then our intuition says that for a small positive integer k, there is a function $f'\colon F^k \to F$, such that $\mathrm{Dist}(f') \approx \mathrm{Dist}(f)$ and $\mathrm{Rej}(f') \approx \mathrm{Rej}(f)$. We could try to brute search for f', but this becomes too computationally intensive even for very small values of k. Instead, we look for the Fourier coefficients of the function $(-1)^{f'}$. We achieve this by solving $(\mathrm{P}_{k,\epsilon,\delta})$, for an appropriate choice of ϵ. Of course, we have no guarantee that given an optimal solution $\vec{x}$ for $(\mathrm{P}_{k,\epsilon,\delta})$, the $\widehat{x}_\beta$'s will correspond to the Fourier coefficients of a function of the form $(-1)^{f'}$ where f' is a boolean function over F^k. Nevertheless, we show that, if n is sufficiently large, there is an $f\colon F^n \to F$, and $h \equiv (-1)^f$ for which $\widehat{h}_\alpha \approx \widehat{x}_{\lfloor\alpha\rfloor_k}$ for every $\alpha \in F^n$ such that $\alpha_{k+1} = \cdots = \alpha_n = 0$,[13] and $\widehat{h}_\alpha \approx 0$ otherwise. For a careful choice of ϵ and n, Lemma 2.3.1 coupled with $\vec{x}$'s feasibility will imply $\mathrm{Dist}(f) \approx \delta$. The optimality of $\vec{x}$ and Fact A.0.8 will imply $\mathrm{Rej}(f) \approx \Gamma_{k,\epsilon}(\delta)$.

Proof of Lemma A.0.7: Consider an optimal solution $\vec{x} = (x_v : v \in F^k)$ to $(\mathrm{P}_{k,\epsilon,\delta})$. Moreover, let p denote the real valued function that sends $u \in F^n$ to $p(u) = x_{\lfloor u\rfloor_k}$. Observe that the function p always takes values in $[-1,1]$. We now proceed to randomly choose a

[13] Recall that if $\alpha = (\alpha_i : i = 1,\ldots,n)$, then, $\lfloor\alpha\rfloor_k$ denotes $(\alpha_i : i = 1,\ldots,k)$.

function $f\colon F^n \to F$. Set $f(u)$ to 1 with probability $(1-p(u))/2$, and to 0 with probability $(1+p(u))/2$. Let $h \equiv (-1)^f$. We now show that, with high probability, the Fourier coefficients of h are close to the Fourier coefficients of p.

Fact A.0.9 The probability that there is an $\alpha \in F^n$ such that $\left|\widehat{h}_\alpha - \widehat{p}_\alpha\right| \geq \epsilon$ is at most $4\,|F|^n\, e^{-\epsilon^2|F|^{n-k}/4}$.

Proof: [Sketch] Note that $\widehat{h}_\alpha - \widehat{p}_\alpha = \frac{1}{|F|^{n-k}}\mathrm{E}_{v\in_R F^k}\left[\sum\left(h(u)-p(u)\right)(-1)^{\alpha\cdot u}\right]$, where the summation ranges over $u \in F^n$ such that $\lfloor u \rfloor_k = v$. Hence,

$$\Pr\left[\left|\widehat{h}_\alpha - \widehat{p}_\alpha\right| \geq \epsilon\right] \;\leq\; \max_{v\in F^k}\Pr\left[\left|\sum_{u\in F^n:\lfloor u\rfloor_k = v}\frac{1}{2}\left(h(u)-p(u)\right)(-1)^{\alpha\cdot u}\right| \geq \frac{\epsilon\,|F|^{n-k}}{2}\right].$$

A Chernoff bound (see [AS92a, Corollary A.7]) implies that the RHS above is at most $2\,e^{-\epsilon^2|F|^{n-k}/2}$ if $\alpha_{k+1} = \cdots = \alpha_n = 0$, and at most $4\,e^{-\epsilon^2|F|^{n-k}/4}$ otherwise. ■

Observe that $\widehat{p}_\alpha$ equals $\widetilde{x}_{\lfloor\alpha\rfloor_k}$ if $\alpha_{k+1} = \cdots = \alpha_n = 0$, and 0 otherwise. But, $\widetilde{x}$ is a feasible solution to $(\mathrm{P}_{k,\epsilon,\delta})$. Thus, choosing n such that $4\,|F|^n\,e^{-\epsilon^2|F|^{n-k}/4} \leq 1/2$, we get from Fact A.0.9 that there exists a function $f_{n,k}\colon F^n \to F$ such that if $h \equiv (-1)^{f_{n,k}}$, then $\widehat{h}_0 \geq \widehat{h}_\alpha$ for all $\alpha \in F^n$, $\left|\widehat{h}_0 - (1-2\delta)\right| \leq \epsilon$, and $\left|\widehat{h}_\alpha\right| \leq \epsilon$ unless $\alpha_{k+1} = \ldots = \alpha_n = 0$. Hence, by Lemma 2.3.1 $|\,\mathrm{Dist}(f_{n,k}) - \delta\,| \leq \epsilon/2$. Moreover,

$$\sum_{\alpha\in F^n}\left(\widehat{h}_\alpha\right)^3 \geq \sum_{\beta\in F^k}(\widetilde{x}_\beta - \epsilon)^3 - \epsilon\sum_{\alpha\in F^n}\left(\widehat{h}_\alpha\right)^2.$$

By Parseval's $\sum_{\alpha\in F^n}\left(\widehat{h}_\alpha\right)^2 = 1$, and since $|\,\widetilde{x}_\beta\,| \leq 1$ for all $\beta \in F^k$

$$\sum_{\alpha\in F^n}\left(\widehat{h}_\alpha\right)^3 \geq \sum_{\beta\in F^k}(\widetilde{x}_\beta)^3 - O\left(\epsilon\,|F|^k\right).$$

Since $\widetilde{x}$ is an optimal solution of $(\mathrm{P}_{k,\epsilon,\delta})$, it follows that $\sum_{\beta\in F^k}(\widetilde{x}_\beta)^3 \geq \Psi_{k,\epsilon}(\delta)$. Thus, Fact A.0.8 yields $\mathrm{Rej}(f) \leq \frac{1}{2}\left(1 - \Psi_{k,\epsilon}(\delta)\right) + O\left(\epsilon\,|F|^k\right)$. ■