Query-Efficient Checking of Proofs and Improved PCP Characterizations of NP

by

Venkatesan Guruswami

Bachelor of Technology (Computer Science and Engineering)
Indian Institute of Technology, Madras, India (1997)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

~~May 1999~~

[June 1999]

© Massachusetts Institute of Technology 1999. All rights reserved.

Author ....
Department of Electrical Engineering and Computer Science
May 7, 1999

Certified by.................
Madhu Sudan
Associate Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

WITHDRAWN FROM MIT LIBRARIES

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUL 15 1999

LIBRARIES

ENG

Query-Efficient Checking of Proofs and Improved PCP Characterizations of NP

by

Venkatesan Guruswami

Submitted to the Department of Electrical Engineering and Computer Science
on May 7, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

The advent of Probabilistically Checkable Proofs (PCP) established the surprising result that the validity of a proof can be checked with good accuracy by reading only a very small number of randomly selected locations of the proof. In particular, if one is willing to tolerate a small probability of error, then one need not even read the entire proof in order to check its validity! The celebrated PCP theorem [AS92, ALMSS92] shows that it is possible to encode membership in any NP language into polynomially long proofs in such a manner that a probabilistic polynomial-time verifier can read only a constant number of locations in the proof and still reject any adversarially chosen proof of a false claim of membership with 50% probability. The probability of accepting a "false proof" is called the error probability of the PCP system.

The PCP theorem, in addition to being of inherent interest in proof checking, also has applications in proving hardness of approximation results for a whole genre of optimization problems. The appearance of the PCP theorem spurred a lot of research devoted to finding quantitative strengthenings of it, with improved trade-offs between the different parameters arising in the proof-checking procedure. A sequence of surprising developments along these directions recently culminated in the striking results of Håstad showing that every language in NP has a PCP verifier querying only 3 bits of the proof and having error probability arbitrarily close to 1/2. This characterization of NP is tight as it is known that no verifier querying only 3 bits can achieve an error strictly smaller than 1/2, unless P = NP.

Håstad's construction of the 3-query PCP however has two-sided error in that there is a small but non-zero probability that even a correct proof is rejected, and this fact is used in a very critical manner in his construction. It seems somewhat more natural to require that the verifier only make one-sided error, so that "correct" proofs are always accepted. There is an aesthetically pleasing element to PCP proof systems with one-sided error: Every rejected proof has a short counterexample and the proof system explicitly exhibits a flaw in any proof it rejects; for example in the case of 3 query verifiers, the flaw is in the 3 bits queried.

We give a tight PCP construction for NP that makes only 3 queries to the proof, has error probability arbitrarily close to 1/2, and always accepts "correct" proofs. It is known that such a PCP cannot exist if the 3 queries are made non-adaptively all at once; so one important aspect of our construction is that the 3 queries are made adaptively i.e the next query location is chosen depending upon the value of the previously queried bits. This also establishes for the first time the previously unsuspected result that making queries adaptively is more powerful than making them all at once. We also extend our PCP constructions for a slightly higher number of queries and in many cases construct proof systems with one-sided error achieving the same error probability as the previously best known constructions, which had to resort to two-sided error.

Thesis Supervisor: Madhu Sudan
Title: Associate Professor

Acknowledgments

I would like to thank my advisor Madhu Sudan for all his support and advice, and for always being there to discuss, explain and clarify research and other technical matters. Thanks to Danny Lewin for first explaining to me the work of Håstad, and to Luca Trevisan for all the useful discussions we had and the cool things he taught me when he was at MIT. The results in this thesis are based on joint work with Danny Lewin, Madhu Sudan and Luca Trevisan, and I thank them for their collaboration. Thanks also to Johan Håstad for his wonderful work which prompted and made possible our work.

The author's research was supported in part by NSF grant C-CR-9875511.

I would like to thank the (too many to list) members of the theory group who make it such an exciting and vibrant atmosphere to be part of, and my friends (you know who you are!), both within and outside MIT, both the relatively new ones and the ones who go back a lo..ng way, for their well-wishes and encouragement and for the good times we have had.

Finally, a special thanks to my family for their continuing support and love, and for enduring, with so much difficulty I am sure, the separation of more than 8000 miles between us.

Contents

List of Figures

Chapter 1

Introduction

The notion of proof verification is central to the theory of computing. For example, the fundamental complexity class NP consists of precisely those languages for which there exist short (polynomially long) proofs of membership that can be verified by a deterministic polynomial time algorithm. For the canonical NP-complete problem SAT, the proof that a formula is satisfiable is simply a satisfying assignment itself, and a polynomial time algorithm can check if the assignment indeed satisfies all clauses in the input formula.

Over the past decade there have been a number of new, different proof checking methods that have been proposed and studied, both for intrinsic complexity-theoretic purposes and for other applications. One of these is the now famous notion of a Probabilistically Checkable Proof (henceforth PCP), which is motivated by the following question: "How can a proof be written so that it can be verified very efficiently?"[1] In particular, can one write (encode) the proof in such a manner that it can be checked with reasonable accuracy even without looking at the entire proof, i.e can a verification procedure look at only a few locations of the proof and still spot flaws in incorrect proofs with good probability? This is clearly not possible if the verifier strategy is deterministic, and such a verifier must necessarily look at the entire proof. This, however, is not true for a probabilistic verifier that can toss random coins and probe randomly selected locations of the proof, and this leads to the definition of the intriguing notion of PCP systems as described below.

[1]Historically the primary interest in studying PCPs was their immediate application to proving hardness of approximate versions of NP-hard problems, but there is no denying that there is these are of inherent interest in the domain of proof checking itself.

1.1 Probabilistically Checkable Proofs

Informally, a Probabilistically Checkable Proof (PCP) system for a language L consists of a *verifier* V which is a probabilistic polynomial-time Turing machine whose goal is ascertain membership of an input string x in a language L. The verifier is given oracle access to a (binary) proof string Π which purportedly proves the statement $x \in L$. The verifier tosses a certain number of random coins, and decides, based on its random string and x whether to accept or reject the proof Π.

Since we are interested in the efficiency of the proof-checking procedure, we can parameterize PCP systems by their complexity and performance. For example, for integer valued functions r and q, we can require that the verifier, for inputs x of length n, tosses at most $O(r(n))$ random coins and queries the oracle (proof) Π in at most $O(q(n))$ locations; we refer to such verifiers as $(r(n), q(n))$-restricted verifiers. (We move freely between the usages "querying an oracle" and "reading a proof".) These quantify the complexity of the verification procedure and we expect that $r(n)$ and $q(n)$ will be much smaller functions than n so that the proof-checking will be very efficient. We can also quantify the performance of the verifier by additional parameters $c(n)$ and $s(n)$ (called *completeness* and *soundness* respectively) which satisfy : (a) When $x \in L$, there exists a proof Π such that the verifier with input x and oracle access to Π accepts with probability at least $c(n)$ over its coin tosses, and (b) When $x \notin L$, for all proofs Π the verifier accepts with probability at most $s(n)$ over its coin tosses. This leads to the formal definition of a class of languages in terms of the parameters of the PCP systems languages in the class admit.

Definition 1 *The class* $\mathrm{PCP}_{c(n),s(n)}[r(\cdot), q(\cdot)]$ *consists of all languages for which there exist* $(r(n), q(n))$*-restricted verifiers with completeness* $c(n)$ *and soundness* $s(n)$.

In the notation of the above definition, it is clear that $\mathrm{NP} = \mathrm{PCP}_{1,0}[0, \mathrm{poly}]$.

It might seem a bit strange that we do not always insist that $c(n)$ equal 1 (i.e require that when $x \in L$, the prover can write down a proof which the verifier always accepts), but we will see shortly that not requiring so has led to some very interesting results. When $c = 1$ and $s = 1/2$ we omit the subscripts c, s and refer to the PCP class as simply $\mathrm{PCP}[r(n), q(n)]$.

The study of PCPs has involved interest in a host of parameters which are related to the efficiency of the proof checking procedure. We have already encountered the most basic ones among these like query complexity, randomness complexity, completeness and soundness.

We look at these and further parameters in the next section.

1.2 Parameters of PCPs

We now list the parameters relating to PCPs that have been considered in the literature by various authors; most of these parameters were defined and studied extensively for the purposes of proving stronger, and sometimes tight, inapproximability results for certain classical optimization problems. We only define these parameters in this section; their applications and the motivation to study them are discussed in a later section. All these parameters are functions of the length n of the input x to the PCP verifier V.

Randomness: This refers to the number of coin tosses that the verifier makes; it is typically denoted by $r(n)$.

Completeness: Denoted by c. Formally defined as:

$$c(n) = \min\{\max_{\Pi} \mathop{\mathbf{Pr}}_{R}[V^{\Pi}(x;R) \text{ accepts}] : x \in L \text{ and } |x| = n\} \ .$$

We are usually interested in PCPs with $c(n)$ being some constant function. Of special interest is the case when the PCP has completeness 1, we then say it has *perfect completeness.* A family of PCP constructions, with completeness $1 - \varepsilon$ for any $\varepsilon > 0$, is said to have *near-perfect completeness.*

Soundness: This denotes the maximum probability of accepting a "false proof". Formally, the soundness s is defined as:

$$s(n) = \max\{\max_{\Pi} \mathop{\mathbf{Pr}}_{R}[V^{\Pi}(x;R) \text{ accepts}] : x \notin L \text{ and } |x| = n\} \ .$$

As with the completeness, we are usually interested in PCPs with $s(n)$ being some constant function.

Error-probability: This stands for the ratio s/c of soundness to completeness.

We now move to various measures of information conveyed by the proof oracle to the verifier. We only consider oracles which return a single bit on each query.

Query Complexity: Denoted by $q(n)$ this refers to the maximum number of queries made by the verifier to the proof oracle Π over all proofs Π and all strings x of length n.

Free-bit complexity: Intuitively the *free-bit complexity* of a PCP verifier is f, if after reading f bits of the proof, the verifier knows the (unique) values it should expect in the remaining bits it is going to read. For instance if the verifier reads three bits of the proof and accepts iff their parity is 0, then the free-bit complexity of the verifier is 2, since once it reads the values of the first two bits, it knows the value of the third bit which will cause it to accept. To formalize this, let us denote by $\mathsf{pattern}_V(x; R)$ the set of all sequences a such that the verifier accepts with input x and random string R when the sequence of bits it reads from the proof oracle is a. The free-bit complexity of V is the maximum of $(\lg |\mathsf{pattern}_V(x; R)|)$ over all x, R (lg denotes logarithm to the base 2). Equivalently, a PCP verifier uses f free bits if for any input x, and any fixed random string R, its acceptance predicate (which is a boolean function of the bits of proof) can be expressed by a DNF formula with at most 2^f terms every pair of which are *inconsistent*, i.e no truth assignment to the bits of proof can simultaneously satisfy more than one of these terms.

Amortized query complexity: V is said to have *amortized query complexity* $\bar{q}$ (for a constant $\bar{q}$) if for some constant k it reads at most $\bar{q}k$ bits and has error-probability at most 2^{-k}. The amortized query complexity of a PCP that has query complexity q (for constant q) and soundness and completeness s, c is defined as: $\bar{q} = q/\lg(c/s)$.

Amortized free-bit complexity: The amortized *free-bit complexity* of a PCP that has free-bit complexity f (for constant f) and soundness and completeness s, c is likewise defined as: $\bar{f} = f/\lg(c/s)$.

Similar to the notation $\mathrm{PCP}[r, q]$ we denote PCP classes parameterized in terms of their free bit complexity as $\mathrm{FPCP}_{c,s}[r, f]$ for the class of languages which have PCP verifiers with soundness and completeness s, c and use f free bits and $O(r)$ random coin tosses.

Adaptive Vs. Non-adaptive PCPs: We now discuss another subtlety in the definition of a PCP. Note that the definition of the PCP allows for the verifier to be *adaptive*, i.e to make future queries depending upon the values it reads for its previous queries. The only restriction is that, for each input x and random string R, it reads at most q bits of the proof and then decides to accept or reject. One could imagine requiring the verifier to be *non–adaptive* i.e to decide the (at most) q locations of the proof it is going to read all at once depending upon x, R alone, and then query them and finally accept or reject. A PCP whose verifier is constrained to be non–adaptive is called a *non-adaptive* PCP and is denoted by naPCP. Whenever we want to make the non-adaptivity of a PCP construction explicit, we will always use this notation.

One can also define a notion of non-adaptivity associated with the free-bit complexity of a PCP verifier. The definition we gave for free-bit complexity above corresponds to the adaptive case. We say a verifier has *non-adaptive free-bit complexity* f, if for any input x and any fixed random string R, its acceptance predicate is a boolean function that has at most 2^f satisfying assignments. It is easy to see that a verifier using f non-adaptive free bits also uses only f (adaptive) free bits; the non-adaptive free-bit complexity of a verifier could, however, be much higher than its adaptive free-bit complexity. For example if the acceptance predicate φ of the verifier for each fixed x, R is of the form $\varphi = (a \wedge b) \vee (\bar{a} \wedge c)$ for some variables a, b, c, then its free-bit complexity is only 1 while its non-adaptive free-bit complexity is 2 as there are four assignments to (a, b, c) which satisfy φ. PCP classes parameterized in terms of their non-adaptive free bit complexity are denoted using naFPCP.

1.3 PCP Constructions: A Brief History

We now begin sketching a brief history of results in the area of PCPs leading up to our work.[2] The study of PCPs has been a subject of active research in the past few years. A central and important result early in the area was the result of Babai, Fortnow and Lund [5], who, stating in the above terminology, proved that NEXP $\subseteq$ PCP[poly, poly]. This important result was scaled down to the NP level by two independent works. Babai, Fortnow, Levin and Szegedy [6] proved that there exist PCPs for NP in which the verifier runs in polylogarithmic time. The definition of PCPs as above was implicit in the seminal work of Feige, Goldwasser, Lovász, Safra and Szegedy [11] who observed a remarkable connection between PCP systems and the hardness of approximating Max-Clique. They proved that NP $\subseteq$ PCP[$\log n \log\log n, \log n \log\log n$], and as a consequence of their connection to ap-

[2]We stress that this area is rich in deep and beautiful results and it is impossible to cite each piece of important work; we have done our best to cite and mention all results that closely relate to our line of study.

proximating clique deduced that Max-Clique cannot be approximated within any constant factor unless NP $\subseteq$ Dtime($n^{\log\log n}$).

The purpose of proving stronger hardness results for approximating Max-Clique motivated much of the initial investigation on PCPs. Arora and Safra [2] proved that NP $\subseteq$ $\mathrm{PCP}_{1,1/2}[\log, o(\log)]$, this proved that Max-Clique is NP-hard to approximate within any constant factor. They made the definition of PCPs and of the class PCP$[r, q]$ explicit, and identified query complexity as an important parameter of PCP constructions. Their work also introduced and used the idea of recursive proof checking (also known as proof composition) and this has played an important role in all future developments in the area. Arora, Lund, Motwani, Sudan and Szegedy [1] showed how to reduce the query complexity to a constant while preserving logarithmic randomness, yielding the celebrated PCP theorem that states

$$\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[\log, O(1)] \ .$$

The above, in addition to being an astonishing and inherently interesting statement about proof checking itself, also had remarkable applications to proving hardness of approximation problems: for example NP-hardness of approximating Max-Clique within N^{ε} for some $\varepsilon > 0$, and the NP-hardness of approximating MAX 3-SAT within some constant factor.

The appearance of the PCP theorem spurred a lot of research devoted to finding quantitative strengthening of it, with improved trade-offs between the different parameters arising in the proof-checking procedure. One main motivation behind this line of research is to get improved, and sometimes tight, hardness results for approximately solving various classical optimization problems like Max-Clique and MAX 3-SAT.

The work in [2, 1] distilled the role of "recursive proof checking" as an important tool in PCP constructions, and the construction of constant-prover one-round proof systems became an important step in the latter steps of the recursion as it is instrumental in getting PCP systems that make only a constant number of queries. In a constant-prover one-round proof system, a probabilistic polynomial time verifier makes one probe to each one of a number p (which is $O(1)$) of provers each of which returns possibly many bits as an answer. The verifier then decides to accept or reject depending upon the answers it receives. As in PCPs, there are several parameters of interest here like the number of provers, the answer sizes of the provers, the randomness complexity of the prover, to name a few. Following

[1], the task of constructing PCPs bifurcated into the tasks of constructing good constant-prover proof systems (which will serve as what have been called *outer verifiers*) and good *inner verification* procedures (these notions will be formalized and discussed in detail in the next Chapter).

PCP constructions ideally require constant-prover proof systems whose error can be made an arbitrarily small constant while keeping the randomness and answer sizes small, and the number of provers fixed, preferably two. Early constructions of constant-prover proof systems due to Lapidot and Shamir [18] and Feige and Lovász [12] used polylogarithmic randomness and answer sizes. The randomness complexity was reduced to logarithmic in [1] at the expense of increasing the number of provers to a constant much larger than 2. Bellare et al. [8] achieve the same logarithmic randomness while keeping the number of provers bounded (they needed 4 provers) and also achieve sub-logarithmic answer sizes. Significant progress was made when Feige and Kilian [13] constructed constant-prover proof systems with logarithmic randomness, constant answer size and only 2 provers. Bellare and Sudan [9] identified extra features of two-prover one-round proof systems that could be exploited to construct better PCPs; the 2-prover proof systems of [13], however, did not have such properties, so instead [9] worked with a modification of the proof systems in [18, 12]. A breakthrough came in the form of Raz's parallel repetition theorem [21], which gave a construction of a 2-prover 1-round proof system with logarithmic randomness and constant answer size, and in addition had the properties which [9] needed. The proof system of Raz has been the "outer verifier" in all future PCP constructions, beginning with the ones in [7].

This is a good point to remark that constant-prover proof systems, in addition to being of importance to PCP constructions, are of intrinsic interest and also have direct applications in proving hardness of approximations; for instance in the inapproximability results for Set Cover [19, 10]. There still remain interesting questions pertaining to constant-prover proof systems themselves, and there has been recent work in [22, 3] obtaining constant-prover proof systems with sub-constant error-probability while using logarithmic randomness and answer sizes. Our work will not focus on results on constant-prover proof systems; we will just use the construction of Raz [21] as an "outer verifier" in order to get our PCP constructions.

The sequence of works [8, 13, 9, 7] identified the important parameters for PCPs (like query bits, free bits, amortized free bits, soundness, etc.), and gave better and better PCP constructions for NP (all using logarithmic randomness) with improved parameters using the above constructions of constant-prover proof systems together with some other ideas to perform efficient "inner verification". The notion of free bits was implicit in [13] and was formalized in [9]. Amortized free-bits were introduced in [9] and formalized better in [7]. PCPs with low free bit complexity have applications to proving hardness of approximating Vertex Cover, while amortized free bits turns out to be the right measure to optimize for inapproximability results for Max-Clique [9, 7].

A PCP construction with amortized free-bit complexity $2 + \varepsilon$ was given in [7]. This paper introduced the *Long code* as an error-correcting code to be used in proof composition, and this important discovery has been the standard for subsequent constructions. They also showed that $\mathrm{NP} \subseteq \mathrm{PCP}_{1,0.85+\varepsilon}[\log, 3]$ and $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,0.7935+\varepsilon}[\log, 2]$. They also considered the question of determining the minimum q for which $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[\log, q]$ and showed that $q = 11$ would suffice.[3] Håstad [15], in a major breakthrough, constructed PCPs with amortized free-bit complexity ε for arbitrary $\varepsilon > 0$. Trevisan [27] and Sudan and Trevisan [24] construct PCPs with amortized query complexity $1.5 + \varepsilon$ for any $\varepsilon > 0$.

This sequence of developments [5, 4, 11, 2, 1, 8, 13, 9, 21, 7, 15] culminated in the striking results of Håstad [16] showing that every language in NP has a PCP verifier querying 3 bits and having error probability arbitrarily close to 1/2. This characterization of NP is tight in the sense that no verifier querying 3 bits could achieve an error strictly smaller than $\frac{1}{2}$ [29]. Håstad in his papers [15, 16] actually describes a general machinery for analyzing Long code based PCP constructions using Discrete Fourier analysis, and in principle these methods could yield a tight analysis of *any* given verifier.

1.4 Current Research Directions and Our Main Results

Current results in better quantitative improvements to existing PCP constructions (for NP) can be classified into (broadly) four kind of endeavors depending upon the complexity measure they are trying to optimize:

[3]This question was addressed in initial works in order to prove the hardness of approximating Max-Clique. While we now know amortized free-bits is the right measure for this purpose, the query complexity of a proof system is still a most natural measure, and it is an intriguing question how many query bits one needs in order to achieve a certain error probability.

1. Amortized free bits: This is the correct parameter to optimize for applications to inapproximability results for Max-Clique, as shown by [9, 7]. A tight result achieving ε amortized free bits for any $\varepsilon > 0$ has been shown by Håstad [15].

2. Free bits: The goal here is to achieve the best soundness possible for PCPs that use a small number of free bits; free-bit complexity is the right measure to optimize for obtaining inapproximability results for Vertex Cover. The best known construction for free bits is again by Håstad [16] and achieves near-perfect completeness, soundness 1/2 using only two free bits.

3. Amortized query bits: This parameter specifies the precise rate at which the error of a PCP goes down with the number of queries it makes, and also has applications to the approximability of the Boolean constraint satisfaction problem with constraints involving at most k variables, usually called MAX k-CSP. A construction achieving $1.5+\varepsilon$ amortized query bits is known [27, 24], and recently [23] announced an optimal PCP construction achieving $1+\varepsilon$ amortized query bits for any $\varepsilon > 0$ (a lower bound of 1 holds for the amortized query complexity of any PCP system powerful enough to capture NP, assuming P $\neq$ NP).

4. Query bits: The goal here is to achieve the best soundness possible for PCPs that use a small number of query bits. The query complexity being such a natural parameter of the proof-checking procedure, this is a fundamental question in the understanding of PCPs, and it also has applications to proving hardness results for approximating fundamental optimization problems like MAX 3-SAT, MAX CUT, MAX 2-SAT, MAX k-CSP for small values of k, to name a few.

Since tight results have been obtained for the case of amortized query and free bits, our work focuses on obtaining improved PCP constructions for the parameters of query bits, and also as corollaries, free bits.

A "tight" PCP construction in terms of query bits follows from the work of Håstad, namely his 3-query PCP construction with error-probability 1/2 [16]. In the process of proving this tight characterization of NP in terms of 3-query PCPs, Håstad's work exposes some of the previously unexplored subtleties in the definition of a PCP system. Recall that such a proof system is described by an (r, q)-restricted PCP verifier, and also the soundness and completeness parameters s, c. Notice that while the definition allows for the verifier to make *two-sided* error i.e have completeness less than 1, most PCP constructions to date restricted their attention to verifiers making one-sided error (i.e having perfect completeness). There are several reasons for this: (1) It was not known how to exploit the power of two-sided error, and (2) Verifiers with one-sided error work better in proof composition and in many applications to inapproximability results. Moreover, there is an aesthetically pleasing element to PCP proof systems with one-sided error when looked from the perspective of proof-checking: There is a short counterexample to any rejected proof and verifier explicitly exhibits a flaw in any proof it rejects; and in the case of 3 query verifiers, the flaw is in the 3 bits queried. In fact, the original definition of PCPs in [2, 1] required the proof system to have perfect completeness.

Håstad's construction, however, yields a verifier making two-sided error. Specifically, his construction has near-perfect completeness and when $x \in L$ the verifier makes an arbitrarily small but non-zero error, i.e he proves $\mathrm{NP} \subseteq \mathrm{PCP}_{1-\varepsilon,1/2}[\log, 3]$. The near-perfect completeness is inherent in his construction and leaves open the question: What is the lowest error that can be achieved in a 3-query PCP for NP having perfect completeness? In light of the newly acquired ability to perform (at least in principle) a tight analysis of almost any PCP verifier through Håstad's work, it seems feasible to examine this question: The only challenge seems to be in designing the right PCP verifier. Yet, the best previous construction of a PCP verifier that queries three bits and has perfect completeness only achieves an error probability arbitrarily close to 3/4 [16].

Trevisan [26] and Zwick [29] show a fundamental barrier to this quest: They show that any PCP verifier making 3 *non-adaptive* queries to the proof oracle, and having perfect completeness and soundness less than 5/8 can only recognize languages in P. This brings up another subtlety in the definition of PCPs: as we mentioned in Section 1.2 the definition actually allows the queries of the verifier to be generated *adaptively*. Most previous PCP constructions do not use adaptivity. However, to get a tight answer to the 3-query question, it seems necessary to build an adaptive verifier; and the only construction of an adaptive PCP verifier in the literature is a construction of Bellare et al. [7]. Thus this entire area seems relatively unexplored.

We build a new *adaptive* 3-query verifier for NP. This verifier is based on a combination of the adaptive verifier of Bellare et al. [7] and the non-adaptive verifier with perfect completeness of Håstad [16]. We perform a tight analysis of this verifier and obtain a somewhat surprising result:

Theorem 1.1 *For every* $\varepsilon > 0$, $\mathrm{NP} = \mathrm{PCP}_{1,\frac{1}{2}+\varepsilon}[\log, 3]$.

The theorem is tight since, as pointed out earlier, any 3-query verifier for NP must make an error with probability at least $\frac{1}{2}$ [29]. This theorem, therefore, resolves a central question relating to PCPs by obtaining a *tight characterization of NP* in terms of a 3-query PCP making one-sided error. [4] The surprising element of the result above is that it shows that an adaptive verifier can achieve a lower error than any non-adaptive verifier — thereby establishing a separation between adaptive and non-adaptive PCPs. Prior to this result there was no evidence indicating that such a separation might exist. In fact, on the contrary, Trevisan [25] points out that adaptive and non-adaptive PCPs actually have the same power for PCPs with *two-sided* error. Of technical interest is that we extend (in retrospect, quite easily) the Fourier analysis method of Håstad to the case of adaptive PCPs. While our proof of Theorem 1.1 borrows lots of ideas from [16], our presentation here is self-contained and also different from the one in [16] which does not make proof composition and the notion of inner verification explicit.

We move on to examine PCPs with slightly higher number of queries. One motivation for addressing this question is to determine the smallest number of queries required for a (non-adaptive) PCP to have soundness strictly less than 1/2. This question has the intriguing aspect of answering what the constant in the PCP Theorem (stated with completeness 1 and soundness 1/2 and requiring non–adaptivity) actually is. This quest was initiated in [8] and pursued further in [13, 9, 21, 7] and the best bound prior to our work was 9 and was implicit in the work of [16]. Another motivation is the following question: Is it true that every additional query increases the power of PCPs? (I.e., is $\mathrm{PCP}_{1,s}[\log, q] \subseteq \mathrm{PCP}_{1,s'}[\log, q+1]$, for some $s' < s$?[5]) It is easy to see that 3 additional bits certainly reduce the error. Yet, for one additional bit we do not know the answer. In fact, prior to this paper, it was not even known if there exists a non-trivial q ($q \geq 3$) for which this statement is true. To answer such questions, we prove some more new (but not necessarily tight) PCP characterizations

[4] At this stage almost every question relating to NP and PCP with 3-queries seems to be tightly resolved, except for the case of a non-adaptive PCP with perfect completeness where the best soundness achievable lies somewhere between 5/8 and 3/4.

[5] This question was posed to us by Oded Goldreich.

of NP (recall that the notation naPCP below stands for non-adaptive PCP).

Theorem 1.2 *For every $\varepsilon > 0$, the following hold:*

(1) **(4 non-adaptive queries)** $\mathrm{NP} = \mathrm{naPCP}_{1,\frac{1}{2}+\varepsilon}[\log, 4]$.

(2) **(5 adaptive queries)** $\mathrm{NP} = \mathrm{PCP}_{1,\frac{1}{4}+\varepsilon}[\log, 5]$.

(3) **(5 non-adaptive queries)** $\mathrm{NP} = \mathrm{naPCP}_{1,\frac{7}{16}+\varepsilon}[\log, 5]$.

(4) **(6 non-adaptive queries)** $\mathrm{NP} = \mathrm{naPCP}_{1,\frac{1}{4}+\varepsilon}[\log, 6]$.

Part (2) of result is where the main technical work is done. Parts (1) and (4) are immediate corollaries of the adaptive protocols in Theorem 1.1 and Part (2) of the theorem above: The non-adaptive verifier is obtained by reading all possible bits that may be read by the adaptive verifier. It is interesting to observe that that for several choices of q, the best known non-adaptive PCP verifier is obtained by starting from an adaptive one. Part (3) above requires some modification of the verifier of Part (2), and shows that at most 5 non-adaptive queries are required to get soundness below 1/2 (improving the bound of 9 known earlier).

The above results together yield some partial answer to the question posed earlier, since they exhibit a non-trivial value of q for which $q + 1$ queries give more power than q: For non-adaptive PCPs, 4 queries are stronger than 3. (This follows from Theorem 1.2 (1) and the fact that $\mathrm{naPCP}_{1,s}[\log, 3] \subseteq \mathrm{P}$ for any $s < 5/8$ [26, 29].) We will observe some more such results in Chapter 7.

Our PCP constructions also have implications for optimizing free bits. For instance, our 3-query construction also proves that $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,1/2+\varepsilon}[\log, 2]$, thereby answering an open question raised in [7]. We also obtain a few other such results, and also note a result which hints at potential difficulties which have to be surmounted in order to get PCP constructions that significantly improve the inapproximability results for Vertex Cover. These results are described in detail in Chapter 5. We also obtain some results showing that certain PCP classes (with perfect and near-perfect completeness) and certain soundness to query complexity trade-offs in fact collapse to P; these results will be mentioned in Chapter 6.

A preliminary version of the results presented in this thesis appeared in the paper [14].

1.5 Organization of the Thesis

We give some background on the methodology of PCP constructions and describe in detail the notion of proof composition as needed for our constructions in Chapter 2. We describe the 3-query PCP construction that proves Theorem 1.1 in Chapter 3. Chapter 4 describes the PCP constructions with higher number of queries and proves all the parts in Theorem 1.2 above. PCP constructions optimizing free bits that follow from our constructions in Chapters 3 and 4, together with a few others, are described in Chapter 5. Chapter 6 proves some results that establish limitations on the power of PCPs that make a small number of queries and have perfect or near-perfect completeness. Finally, in Chapter 7 we make some concluding remarks and list a few open questions and directions for future research.

Chapter 2

Proof Methodology and Background

2.1 Proof Composition: The general paradigm

Our PCP constructions rely on the proof-composition methodology introduced by Arora and Safra [2] and then refined in [1, 8, 9, 7, 16]. The main idea behind proof composition is to construct two proof systems, that optimize different efficiency parameters, and then combine them to build a composed proof system which is efficient under all the parameters. In this methodology in its current, most convenient form, one composes an *outer verifier* V_{out} that is typically a 2-Provers 1-Round (2P1R) protocol (in which the verifier makes one query to each of two provers and accepts if their answers are "consistent") and an *inner verifier* V_{in}. The verifier of the composed system V_{comp} expects as proof the entry-wise encoding of the proof of V_{out} using an error correcting code. V_{comp} starts of by simulating V_{out}, choosing two entries of the proof as V_{out} would, and then calls as a subroutine V_{in} whose job is to determine, given two strings, whether they are the valid encodings of two possible consistent answers that would have been accepted by V_{out}(see Figure 2-1). Since this requirement on V_{in} is too strong, we relax it to testing if the two strings are *close to* valid encodings of consistent answers. The power of this methodology in obtaining very good parameters for PCP constructions comes from the fact that V_{comp} inherits the query complexity and error probability of the inner verifier V_{in}, and even if V_{in} requires a large amount of randomness, the composed verifier will still only use logarithmic randomness.

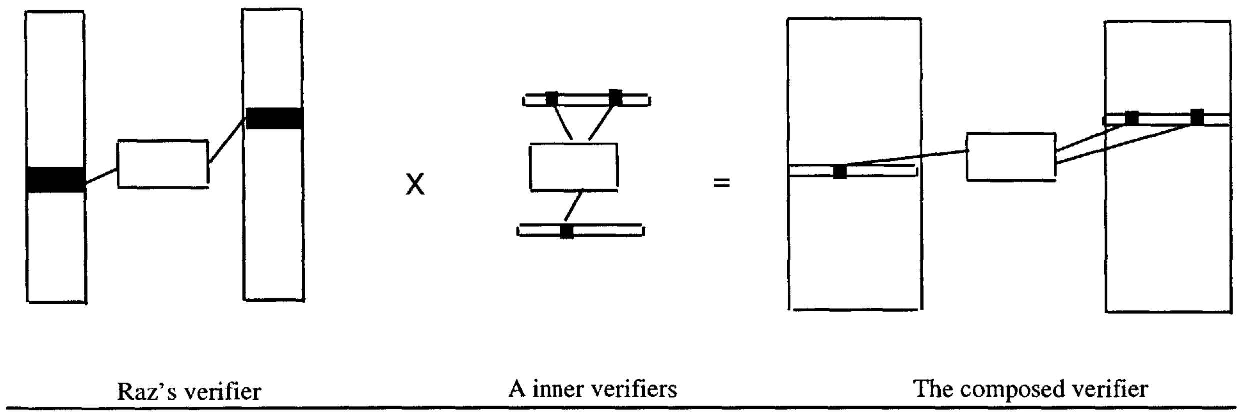

Figure 2-1: Proof composition

Moreover, as we shall see shortly, an outer verifier that is essentially the best possible one for our purposes has been given by Raz [21], and therefore resorting to proof composition reduces the task of designing PCPs to the simpler task of constructing inner verifiers.

Let us look a little more closely at the properties of the inner verifier. It knows the acceptance criterion of V_{out}, and, given oracle access to two strings which are allegedly encodings of answers that V_{out} would have accepted, it job is to test whether this is the case. In order to design a PCP with completeness c and soundness s, we would like to say that: (a) Whenever all the conditions are satisfied, V_{in} accepts with probability at least c, and, stating the soundness condition in the contrapositive form, (b) Whenever V_{in} accepts with probability more than s, the strings it is accessing are close to being valid encodings of answers that would have caused V_{out} to accept.

The second property above turns out to be a tricky one to formalize. One possible approach is to require that in the case when V_{in} accepts with probability more than s, there is a way to "decode" the strings it is accessing independently into possible consistent answers for the outer 2P1R protocol. This requirement is however too stringent. It turns out that a weaker requirement that will still suffice for purposes of composition, is to allow the decoding procedures to be randomized and require that if V_{in} accepts with probability more than $s + \delta$, then the decoding procedure, applied independently to the two strings which the inner verifier accesses, produces a pair of consistent answers (for V_{out} in the outer protocol) with probability at least $\gamma > 0$, for some γ which depends only on δ. Such a decoding procedure was made explicit in [16] and was implicit in the work of [7].

We now proceed to explicitly describe the outer verifier and give more details on the composition scheme the way we define and use it.

2.2 The Outer Verifier

For our outer verifier we use the 2-Prover 1-Round proof system due to Raz [21] which achieves perfect completeness, constant answer size and small soundness (the soundness can be made an arbitrarily small constant by letting the answer size be a sufficiently large constant). We now describe the construction of such a 2P1R protocol, and later abstract the details of the Raz verifier to give the final general form of our outer verifier, as will be most useful to us.

As a consequence of the PCP theorem [2, 1], together with a reduction due to Papadimitriou and Yannakakis [20] , there exists a polynomial time reduction that maps an instance φ of 3SAT to an instance φ' of 3SAT in which each variable occurs in the *same* constant number of clauses, with the property that if φ is satisfiable then so is φ', and if φ is not satisfiable then *every* assignment satisfies less than a fraction $(1-\psi)$ of the clauses of φ', where $\psi > 0$ is an *absolute constant.*

This transformation yields the following 2P1R system for ascertaining membership in 3SAT. Given a formula φ, we first use the above transformation to produce a formula φ' with N variables and M clauses. The verifier of the 2P1R protocol has oracle access to two tables (provers) $P : [N] \to \{0,1\}$ and $Q : [M] \to [7]$ which allegedly encode the same satisfying assignment for φ' (for an integer n we use $[n]$ to denote the set $\{1, 2, \dots, n\}$). Specifically, for every variable x, $P(x)$ contains the truth value of x in the assignment. The Q table supposedly contains, for every clause C, the values of the three variables occurring in C according to the same assignment as defined by P, and encoded as a number between 1 and 7, which represents, say, the index of the assignment in the lexicographic order among the satisfying assignments of C.

The verifier operates as follows: It picks at random a clause C in φ' and then one of the three variables (say the ith variable x for some $i \in \{1,2,3\}$) occurring in C again at random. It then gets answers $a = P(x)$ and $b = Q(c)$. The verifier now determines b_1, b_2, b_3 such that $b \in [7]$ encodes the satisfying assignment b_1, b_2, b_3 of C, and then accepts if $b_i = 1$. Due to the "gap" that exists in the number of satisfiable clauses of φ', it is easy to show

```
Verifier $V^u_{\text{out}}(\varphi; P, Q)$
  (First computes a formula $\varphi'$ with $N$ variables and $M$ clauses
  and which has a "gap" in the number of satisfiable clauses.)
    Randomly pick $q \in [M]^u$.
    Pick a projection function $\pi : [7]^u \to [2]^u$ according to $\mathcal{D}(q)$.
    Compute the query $p = f(q, \pi)$.
    Get answers $a = P(p) \in [2]^u$ and $b = Q(q) \in [7]^u$.
    accept iff $\pi(b) = a$.
```

Figure 2-2: A description of the outer 2-Provers 1-Round Protocol.

that this 2P1R protocol has perfect completeness and soundness $(1 - \psi/3) < 1$.

Our final outer 2P1R will be obtained by driving down the soundness of the above protocol by iterating it several times in *parallel*. For example the u-parallel version of this protocol does the following: The verifier picks at random u clauses $C_1, C_2, \ldots, C_u$ of φ' (possibly with repetitions), and for each C_i it picks a variable x_i occurring in it at random. The prover P is now expected to encode a satisfying assignment to φ', and for every u-tuple of variables P must return the value of this assignment restricted to the variables in that tuple. The prover Q is expected to return, for every u-tuple of clauses, the value of the same assignment (that P encodes) restricted to the variables occurring in the u-clauses (encoded in the obvious way as an element of $[7]^u$). The verifier now gets answers $(a_1, a_2, \ldots, a_u) = P(x_1, x_2, \ldots, x_u)$ and $(b_1, b_2, \ldots, b_u) = Q(C_1, C_2, \ldots, C_u)$ and checks that $\pi(b_1, b_2, \ldots, b_u) = (a_1, a_2, \ldots, a_u)$ where $\pi : [7]^u \to \{0,1\}^u$ is the appropriate *projection function* that extracts the values of $x_1, x_2, \ldots, x_u$ from the values of all the variables occurring in $C_1, C_2, \ldots, C_u$.

The parallel repetition theorem of Raz [21] proves that the above u-parallel version of the basic protocol has soundness c^u for some ***absolute constant*** $c < 1$. It is clear that the 2P1R has perfect completeness and uses answer size of $O(u)$.

We are now ready to abstract the above verifier into a form that will be useful to us and that will also aid us in our presentation. We will use the following description of it (see Figure 2-2): The verifier is parameterized by an integer u and it has oracle access to two tables P, Q (with $P : [N]^u \to [2]^u$ and $Q : [M]^u \to [7]^u$; the exact values of N, M will not be important, we only need N, M to be polynomial in the size of φ). (We have also identified $\{0,1\}^u$ with $[2]^u$ in the obvious manner for clarity of presentation.) The verifier then picks, uniformly at random, an entry q in table Q to query and then picks a projection function

$\pi : [7]^u \to [2]^u$ randomly according to a distribution $\mathcal{D}(q)$ that is determined by q. It then decides the query p into P as a deterministic function f of q, π, and then reads $a = P(p)$ and $b = Q(q)$ from the tables and finally accepts iff $\pi(b) = a$. By the parallel repetition theorem, we obtain:

Propostion 2.1 *For every $\gamma > 0$, the 2P1R using the verifier V_{out}^u for $u = O(\log 1/\gamma)$ has perfect completeness, soundness at most γ and answer size $O(\log 1/\gamma)$.*

A restriction on the distribution $\mathcal{D}(q)$: We now slightly compromise the generality of the description of our outer verifier of Figure 2-2 by placing a restriction on the possible distributions $\mathcal{D}(q)$ that the verifier V_{out}^u is allowed to use. For this purpose we first make the following definition:

Definition 2 *A distribution D on functions $f : S \to T$ is called σ-distinguishing if for every fixed $x \neq y \in S$,*

$$\mathop{\mathbf{Pr}}_{f \in D}[f(x) \neq f(y)] \geq \sigma.$$

We require that each distribution $\mathcal{D}(q)$ be the *product* of distributions $D_1(q), D_2(q), \ldots, D_u(q)$ where each $D_i(q)$ is a σ-distinguishing distribution on functions $\pi_i : [7] \to \{0,1\}$ (for an absolute constant $\sigma > 0$; in fact we may take $\sigma = 1/3$). We refer to such valid distributions $\mathcal{D}(q)$ as *smooth distributions.*

Definition 3 *A distribution D on functions $f : S^n \to T^n$ is called* smooth *if it is the product of n σ-distinguishing distributions $D_1, D_2, \ldots, D_n$ (on functions $g : S \to T$).*

The key property of smooth distributions $\mathcal{D}(q)$ is that a random function π chosen according to $\mathcal{D}(q)$, with good probability, has the property that the image of a "large" set under π is also quite "large". We show this by proving the following lemma, which was shown to us by Luca Trevisan.

Lemma 2.2 (Trevisan) *Let Σ, R be arbitrary finite sets, and D be a smooth distribution over functions $\pi : \Sigma^n \to R^n$. Then, for every k and every set $S \subseteq \Sigma^n$,*

$$|S| \geq |\Sigma|^k \Rightarrow \mathop{\mathbf{Pr}}_{\pi \in D}[|\pi(S)| < \sigma k/2] \leq e^{-k\sigma/8}$$

Proof: Let S be a set such that $|S| \geq |\Sigma|^k$. We define a subset $T \subseteq S$ and we prove that $\mathbf{Pr}[|\bar{\pi}(T)| < \sigma k/2] \leq e^{-k\sigma/8}$.

Towards this goal, we will construct a set $T = \{s_1, \ldots, s_{k+1}\}$ and indices $i_1, \ldots, i_{k+1}$, such that for every $j = 1, \ldots k$ it is the case that

$$s_j[i_j] \neq s_{j+1}[i_j] = \cdots = s_{k+1}[i_j]$$

We prove that such a set T and associated indices $i_1, \ldots, i_{k+1}$ must always exist. We proceed by induction on k. (The proof will also define a recursive algorithm to find the set and the indices — this algorithmic aspect is irrelevant for the sake of the proof.) For the base case $k = 0$ we can take i_1 to be an arbitrary index and T to be a singleton set consisting of an arbitrary element of S. For larger values of k, we let i_1 be an index where not all the elements of S are equal. Let a be the most popular value in the i_1-th entry of the elements of S; let s_1 be an element of S such that $s_1[i_1] \neq a$ and let $S' = \{s \in S : s[i_1] = a\}$; observe that $|S'| \geq |S|/|\Sigma| \geq |\Sigma|^{k-1}$. Apply the inductive hypothesis to S' and get a set $T' \subseteq S'$ of k elements and k indices $i_2, \ldots, i_{k+1}$ with the required property. Then $T' \cup \{s_1\}$ and $i_1, \ldots, i_{k+1}$ have the required property for S.

Let $T = \{s_1, \ldots, s_{k+1}\} \subseteq S$ and $i_1, \ldots, i_{k+1}$ as above. Let $a_j = s_j[i_j]$ and let b_j be the common value of $s_{j+1}[i_j], \ldots, s_{k+1}[i_j]$. Define the random variable X_j for $j = 1, \ldots, k$ whose value is 1 when $\pi_{i_j}[a_j] \neq \pi_{i_j}[b_j]$ and 0 otherwise where π_{i_j} is drawn according to the σ-distinguishing distribution D_{i_j}. Then

1. X_j are mutually independent 0/1 random variables, each one with expectation at least σ.

2. $\sum_j X_j$ is a random variable that lower bounds the number of distinct elements of $\bar{\pi}(T)$ (consider $\{s_j : X_j = 1\}$; then $\bar{\pi}$ maps this elements to different binary strings.)

By the fact that $T \subseteq S$, by part (2) above, and by Chernoff bounds and part (1) above, we have

$$\mathbf{Pr}[|\bar{\pi}(S)| < \sigma k/2] \leq \mathbf{Pr}[|\bar{\pi}(T)| < \sigma k/2] \leq \mathbf{Pr}[\sum_j X_j < \sigma k/2] \leq e^{-k\sigma/8} . \qquad \Box$$

As a consequence of the above lemma and the requirement of *smoothness* we placed on the distributions $\mathcal{D}(q)$, we get

Lemma 2.3 *For every $\beta \subseteq [7]^u$, the distribution $\mathcal{D}(q)$ employed by our outer verifier satisfies:*

$$\mathop{\mathbf{Pr}}_{\pi \in \mathcal{D}(q)}[|\pi(\beta)| \geq \frac{\sigma \log_7 |\beta|}{4}] \geq 1 - e^{-\sigma \log_7 |\beta| / 16} .$$

Proof: Follows from the previous Lemma with choice of $k = \lfloor \log_7 |\beta| \rfloor$. □

2.3 The Inner Verifier

We now need to define a suitable inner verifier that can be used for composition in conjunction with the outer verifier of Figure 2-2. As was briefly discussed in Section 2.1, the inner verifier will be given the acceptance predicate π used by the outer verifier and will have oracle access to supposed encodings A and B of two strings (that lie in $[7]^u$ and $[2]^u$) as per some error-correcting code. The inner verifier needs to check that the two strings A and B are *close to* valid encodings of elements $a \in [2]^u$ and $b \in [7]^u$ which satisfy the acceptance criterion of the outer verifier viz., $\pi(b) = a$, and for the purpose of getting composed PCP verifiers that have small query complexity, the inner verifier needs to do this while making only few queries into the tables A and B. The hope is that by using very redundant encodings A and B, we will be able to reject invalid A and B with good probability even though we make only a small number of queries.

In order to be able to define our inner verifiers, we first need to do two things: (a) define the method of encoding answers a and b of the outer verifier into strings A and B, and (b) specify a (randomized) decoding procedure, that decodes A and B independently into a and b such that whenever the inner verifier having access to tables A and B accepts with large enough probability, a and b satisfy $\pi(b) = a$ with "good" probability (recall the discussion in Section 2.1).

2.3.1 The Long Code

The current standard for the encoding method used in proof composition is the *Long code* introduced by Bellare, Goldreich and Sudan [7] and this code has since been instrumental in all powerful PCP constructions. The Long code of an element x in a domain D consists of the evaluation $f(x)$ for all the $2^{|D|}$ boolean functions $f : D \to \{0, 1\}$. For our purposes, we will only be using domains D which are either $[2]^u$ or $[7]^u$ for a positive integer u, though we still give the presentation in this section for a general domain D. From now on Boolean functions will be defined with values in $\{1, -1\}$ rather than $\{0, 1\}$. The association is that -1 stands for 1 (or true) and 1 stands for 0 (or false). This system is a lot easier to work with as it has the nice property that multiplication in $\{1, -1\}$ acts as Boolean xor in $\{0, 1\}$.

We now establish some notation that will be used throughout and also describe the machinery relating to Long codes that will be necessary and useful for our purposes. For a domain D, we denote by $\mathcal{F}_D$ the set of all functions $f : D \to \{1, -1\}$. The operator $\circ$ denotes composition of functions, and if $f \in \mathcal{F}_R$ and $\pi : D \to R$ then the function $(f \circ \pi) \in \mathcal{F}_D$ is defined as $(f \circ \pi)(b) = f(\pi(b))$ for any $b \in D$. For two sets α, β, we denote by $\alpha \Delta \beta = (\alpha \cup \beta) \setminus (\alpha \cap \beta)$ their symmetric difference.

We say that a function $A : \mathcal{F}_D \to \{-1, 1\}$ is *linear* iff $A(f)A(g) = A(fg)$ for all $f, g \in \mathcal{F}_D$. There are $2^{|D|}$ linear functions l_α, one for each set $\alpha \subseteq D$; it is defined as

$$l_\alpha(f) = \prod_{a \in \alpha} f(a) \ .$$

(By convention, we say that a product ranging over the empty set equals 1.) Note that the Long code is the set of linear functions whose support is a singleton, i.e. $\mathrm{LONG}_D = \{l_{\{x\}} : x \in D\}$, and $l_{\{x\}}$ is the Long code of x.
We will be using the following three standard properties of linear functions:

$$\begin{aligned} l_\alpha(f) l_\alpha(g) &= l_\alpha(fg) \\ l_\alpha(f) l_\beta(f) &= l_{\alpha\Delta\beta}(f) \\ \mathop{\mathbf{E}}_f l_\alpha(f) &= \begin{cases} 1 & \text{If } \alpha = \emptyset \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.1}$$

(In the third equation above f is picked uniformly at random from $\mathcal{F}_D$.)

It is useful to view a function $A : \mathcal{F}_D \to \{-1, 1\}$ as a real-valued function $A : \mathcal{F}_D \to \mathcal{R}$. The set of functions $A : \mathcal{F}_D \to \mathcal{R}$ is a vector space over the reals of dimension $2^{|D|}$. We can define a scalar product between functions as

$$A \cdot B = \frac{1}{2^{|D|}} \sum_{f \in \mathcal{F}_D} A(f)B(f) = \mathop{\mathbf{E}}_f [A(f)B(f)] \ .$$

The set of linear functions is easily seen to form an orthonormal basis for the set of functions $A : \mathcal{F}_D \to \mathcal{R}$ (with respect to the above dot product). This implies that for any such function A we have the *Fourier expansion*

$$A(f) = \sum_{\alpha} \hat{A}_{\alpha} l_{\alpha}(f), \tag{2.2}$$

where for $\alpha \subseteq D$, $\hat{A}_{\alpha} = A \cdot l_{\alpha}$ is the *Fourier coefficient* of A w.r.t α. Equation (2.2) above is an important one and will be used repeatedly in analyzing inner verifiers based on long codes. Observe that for a function $A : \mathcal{F}_D \to \{1, -1\}$, we have $-1 \leq \hat{A}_{\alpha} \leq 1$ for any α. The following proves another useful property of the Fourier coefficients of any $A : \mathcal{F}_D \to \{1, -1\}$.

Propostion 2.4 *For any function $A : \mathcal{F}_D \to \{1, -1\}$ we have*

$$\sum_{\alpha} \hat{A}_{\alpha}^2 = 1 \ .$$

Proof: We have

$$\sum_{\alpha} \hat{A}_{\alpha}^2 = A \cdot A = \mathop{\mathbf{E}}_{f}[A(f)^2] = 1$$

since $A(f) = \pm 1$. □

Folding of Long Codes: We will also need the notion of *folding* as introduced in [7]. Observe that for a long code $A = l_{\{a\}}$, $A(f) = f(a) = -(-f(a)) = -A(-f)$ for any $f \in \mathcal{F}_D$. If A satisfies $A(f) = -A(-f)$ for any $f \in \mathcal{F}_D$, we say that A is *folded*; thus all long codes are folded. For any function $A : \mathcal{F}_D \to \{1, -1\}$, we can define a new function A', called the *folding of A*, that satisfies such a property. Specifically A' can be defined as follows: fix an element $a_0 \in D$, say the smallest element as per some ordering of elements in D, and then define:

$$A'(f) = \left\{ \begin{array}{c|l} A(f) & \text{if } f(a_0) = 1 \\ -A(-f) & \text{if } f(a_0) = -1. \end{array} \right. \tag{2.3}$$

Note that A' clearly satisfies $A'(-f) = -A'(f)$. We stress that for any f, $A'(f)$ can be obtained using just one query into A, and that A' is equal to A if A a valid Long code of some element a. We therefore assume without loss of generality from now on that the strings (which are purported long codes), to which the inner verifier is given oracle access, are all folded.

We now note the following key property of folded functions $A : \mathcal{F}_D \to \{1, -1\}$ which is

```
Decoding Procedure Decode($A$);
  /* $A : \mathcal{F}_D \to \{-1, 1\}$, $A$ folded. Returns an element of $D$. */
    Choose $\alpha \subseteq D$ with probability $\hat{A}_\alpha^2$.
    Pick an $x \in \alpha$ uniformly at random and return $x$.
```

Figure 2-3: The Decoding Procedure

proved in [16].

Lemma 2.5 *If $A : \mathcal{F}_D \to \{1, -1\}$ is folded, then $\hat{A}_S = 0$ for all $S \subseteq D$ with $|S|$ even. In particular we have $\hat{A}_\emptyset = 0$.*

Proof: Recall the definition

$$\hat{A}_S = 2^{-|D|} \sum_{f \in \mathcal{F}_D} A(f) \prod_{x \in S} f(x) \ .$$

Since $A(f) = -A(-f)$ and $l_S(f) = l_S(-f)$ when $|S|$ is even, the two terms corresponding to f and $-f$ cancel and hence we get $\hat{A}_S = 0$. □

With this Fourier machinery we are now ready to describe the decoding procedure to be used with our inner verifier.

2.3.2 The Decoding Procedure

The decoding procedure takes as input $A : \mathcal{F}_D \to \{1, -1\}$ (which we assume, without loss of generality, is folded) and returns an element in D. The description of the decoding procedure is given in Figure 2-3. We remark that the procedure is well-defined since Parseval's identity implies that $\hat{A}_\alpha^2$ in fact defines a probability distribution, and because the procedure will never get stuck by picking $\alpha = \emptyset$ as the assumption that A is folded guarantees $\hat{A}_\emptyset = 0$ (by Lemma 2.5).

2.3.3 Formalizing "Good" Inner Verifiers

We are now ready to define precisely the requirements on our inner verifiers.

Definition 4 (Inner Verifier) *An inner verifier is a randomized oracle program that is given input a positive integer u and a projection function $\pi : [7]^u \to [2]^u$, and has oracle access to folded functions $A : \mathcal{F}_{[2]^u} \to \{1, -1\}$ and $B : \mathcal{F}_{[7]^u} \to \{1, -1\}$.*

Definition 5 (Good Inner Verifier) *An inner verifier V_{in} is (c, s, q)-*good *if the following properties hold:*

- [NUMBER OF QUERIES] *For all u, $V_{\text{in}}^{A,B}(u, \pi)$ makes a total of q (possibly adaptive) queries into the oracles A and B.*

- [COMPLETENESS] *For all u, if A is the Long code of a and B is the Long code of b and $\pi(b) = a$, then*
$$\mathbf{Pr}\,[V_{\text{in}}^{A,B}(u, \pi) \text{ accepts}] \geq c$$
(Here the probability is over the internal randomness of V.)

- [SOUNDNESS] *For any constant $\delta > 0$, there is a constant $\gamma > 0$ and a positive integer u_0 such that for all $u \geq u_0$, if π is drawn according to a distribution that is smooth, then*
$$\mathbf{Pr}\,[V_{\text{in}}^{A^\pi,B}(u, \pi) \text{ accepts}] \geq s + \delta \Longrightarrow \mathbf{Pr}\,[\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] \geq \gamma$$
where the probabilities are taken over the distribution of π and over the internal coin tosses of $V_{\text{in}}(u, \cdot)$ and Decode. *(For each B, we let $A = A^\pi$ to depend on π as we vary π according to its underlying smooth distribution.)*

Remark: A non-adaptive (c, s, q)-good inner verifier is a (c, s, q)-good inner verifier that makes at most q queries non-adaptively for every u, π, A, B.

The above definition is based on the standard definition of an inner verifier (e.g. from [7]) but it incorporates the possibility that the decoding procedure be randomized[1] and also allows the soundness condition to be averaged over the choices of π. The latter is a technicality and it makes the definition less elegant as the restriction that π be drawn randomly from smooth distributions has to be placed. With the precise definition of an inner verifier in place we are now ready to state and prove the Composition Theorem.

[1]Bellare et al. [7] used a deterministic procedure that returned a list of candidates, and this was conceptually similar to the randomized decoding idea that first appeared in [16] which however did not formalize the notion of an inner verifier explicitly. A definition of inner verifier with respect to a randomized decoding procedure is explicit in [27].

```
Verifier $V^u_{\text{comp}}(\varphi; LP, LQ)$
  (First computes a formula $\varphi'$ with $N$ variables and $M$ clauses
  and which has a "gap" in number of satisfiable clauses.)
    Randomly pick $q \in [M]^u$.
    Pick a projection function $\pi : [7]^u \to [2]^u$ according to $\mathcal{D}(q)$. ($\mathcal{D}(q)$ is smooth.)
    Compute the query $p = f(q, \pi)$.
    Let $A = LP(p)$ and $B = LQ(q)$. (Assume $A, B$ are folded.)
    Run $V^{A,B}_{\text{in}}(u, \pi)$.
```

Figure 2-4: A composed verifier that uses an inner verifier V_{in}

2.4 The Composition Theorem

Theorem 2.6 (Composition Theorem) *If there exists a (c, s, q)-good inner verifier, then for any $\varepsilon > 0$,* NP = PCP$_{c,s+\varepsilon}[\log, q]$. *Moreover, if there exists a non-adaptive (c, s, q)-good inner verifier, then, for any $\varepsilon > 0$,* NP = naPCP$_{c,s+\varepsilon}[\log, q]$.

Proof: The proof follows by composing together (using the method that was briefly sketched in Section 2.1) the outer verifier described in Figure 2-2 together with a (c, s, q)-good inner verifier. Let $\varepsilon > 0$ be fixed. The PCP verifier V_{comp} we are claiming to exist will expect as a proof a pair of tables LP and LQ that are the entry-wise Long code of a valid pair of proof oracles P and Q for the outer verifier of Figure 2-2 (i.e. for each query p (q) into P (Q), $LP(p)$ (resp. $LQ(q)$) will be the Long code of $P(p)$ (resp. $Q(q)$)). As discussed earlier, we may assume that all entries of LP and LQ are folded. We will use the outer verifier V^u_{out} with answer size u, perfect completeness and soundness η (where η, u will be specified later, but these will be constants depending only on ε).

The composed verifier V_{comp} is described in Figure 2-4. It is parameterized by an integer u; we assume that u is large enough so that the outer verifier V^u_{out} of Figure 2-2 has soundness η where $\eta > 0$ is a constant that depends only on ε and will be specified later in the proof. V^u_{comp} starts off by simulating the outer verifier V^u_{out} and picks queries p, q and the projection π exactly as V^u_{out} does, and then it executes the inner verification procedure (using a (c, s, q)-good inner verifier V_{in}) on input u, π and with oracle access to the the two supposed Long codes $LP(p)$ and $LQ(q)$.

Clearly the number of queries that V^u_{comp} makes is the same as that made by V_{in}, which by the assumption of (c, s, q)-goodness is at most q. Also, if the inner verifier makes its q

queries non-adaptively, then so does the composed verifier, and hence the statement about non-adaptive PCP constructions will also follow. Since u is a constant, and M, N are polynomial in the size of φ, the randomness complexity of V^u_{comp} is dominated by that of the outer verifier and is clearly $O(\log n)$ (where n is the size of the formula φ).

It is also easy to see that the composed verifier has completeness c. Indeed when φ is satisfiable, let LP and LQ are valid entry-wise Long codes of tables P and Q which will always cause the outer verifier to accept (such tables P, Q exist because the outer verifier V^u_{out} has perfect completeness). The input that is passed to V_{in} in this case satisfies the completeness condition of V_{in}. Therefore V_{in} accepts with probability at least c over its coin tosses for *every* possible choice of coin tosses of the outer verifier. This implies that V^u_{comp} accepts with probability at least c.

The only thing that remains to be proven is that the composed verifier has soundness $s+\varepsilon$. We show this by proving that if V^u_{comp} accepts its proofs LP and LQ with probability at least $s+\varepsilon$, then φ is satisfiable. Using the soundness condition for the outer verifier, in order to prove that φ is satisfiable it suffices to exhibit proofs P, Q that would make V^u_{out} accept with probability at least η, and this what we do next to complete the proof.

By an averaging argument, if V^u_{comp} accepts with probability at least $s+\varepsilon$, then for at least an $\varepsilon/2$ fraction of the random choices of q (or equivalently of $B = LQ(q)$), we have

$$\Pr_{\pi \in_R \mathcal{D}(q)} [V_{\text{in}}^{A^\pi, B}(u, \pi) \text{ accepts}] \geq s + \frac{\varepsilon}{2} \, . \tag{2.4}$$

(Note that once q is picked, the choice of π fixes p and therefore also fixes $A = A^\pi = LP(p)$.) The probability in the above equation is also taken over V_{in}'s internal coin tosses. We denote by G the set of such "good" choices q for which Equation (2.4) holds.

By the soundness condition of the inner verifier V_{in} and the fact that $\mathcal{D}(q)$ is a smooth distribution, Equation (2.4) implies that there is a constant $\gamma = \gamma_{\varepsilon/2}$, such that for all large enough $u \geq u_0$, for every $q_0 \in G$ it is the case that

$$\Pr_{\text{coin tosses of } \mathsf{Decode}, \pi \in_R \mathcal{D}(q_0)} [\mathsf{Decode}(LP(p^\pi)) = \pi(\mathsf{Decode}(LQ(q_0)))] \geq \gamma \tag{2.5}$$

Now imagine constructing proofs P, Q for the outer verifier by using the randomized decoding strategy Decode (on the proof LP, LQ given to the composed verifier) as follows:

- Independently for each p set $P(p) = \mathsf{Decode}(LP(p))$, and
- Independently for each q set $Q(q) = \mathsf{Decode}(LQ(q))$.

(Remember that Decode is a randomized algorithm; in the above definition of P, Q the executions of D have to be independent each time.)

We now mention how to fix the parameter u of V^u_{comp}; pick u such that V^u_{out} has soundness at most $\eta \overset{\text{def}}{=} \gamma\varepsilon/4$ ($\gamma = \gamma_{\varepsilon/2}$ is fixed in Equation (2.5) and by Proposition 2.1 $u = \Omega(\log 1/\eta)$ will suffice for this purpose) and also such that $u \geq u_0$ (u_0 was once again fixed in Equation 2.5).

We now claim that the probability that V^u_{out} accepts P, Q constructed as above (expected over the way P, Q are chosen) is more than η. This will clearly imply that there *exist* proofs P, Q that are accepted by V^u_{out} with probability more than η, and by the soundness condition of V^u_{out} this will imply that φ was satisfiable, as desired. It remains therefore to lower bound this probability.

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{P,Q,q,\pi \in_R \mathcal{D}(q)} [P(p) = \pi(Q(q))] &\geq \mathop{\mathbf{Pr}}_{q}[q \in G] \cdot \mathop{\mathbf{Pr}}_{P,Q,\pi \in \mathcal{D}(q_0)} [P(p) = \pi(Q(q_0)) | q_0 \in G] \\
&\geq \frac{\varepsilon}{2} \cdot \gamma \qquad (2.6) \\
&> \eta \,.
\end{aligned}$$

where Step (2.6) follows since at least an $\varepsilon/2$ fraction of the choices of q fall in G, and for each $q \in G$, we can use Equation (2.5) and the construction of P, Q to conclude that we will have $\pi(Q(q)) = P(p)$ with probability at least γ. This completes the proof of the Composition Theorem. $\square$

Chapter 3

The Adaptive 3-Query PCP Construction

3.1 Intuition behind the construction

Among the tasks of designing and analyzing PCP verifiers, the latter used to be the more challenging one, but the wide applicability of the new analysis techniques for inner verifiers based on Long codes (which were was described in the previous chapter) is shifting much of the difficulty to the design phase. We will spend much of this section motivating the intuition behind our tight 3-query PCP construction (which proves Theorem 1.1) and describing the ideas that lead to it.

As discussed in the previous Chapter, the inner verification problem is: given u, π and given oracle access to *folded* strings A and B (with $A : \mathcal{F}_{[2]^u} \to \{1, -1\}$ and $B : \mathcal{F}_{[7]^u} \to \{1, -1\}$), test whether there exists a $b \in [7]^u$ such that B is the long code of b and A is the long code of $\pi(b)$. Equivalently, we want to test whether for every f, g_1 and g_2, the following properties hold: (1) $A(f) = B(f \circ \pi)$; (2) $B(g_1 \wedge g_2) = B(g_1) \wedge B(g_2)$; (3) $B(g_1 g_2) = B(g_1)B(g_2)$. Properties (2) and (3), together with the fact that B is folded, ensure that B is a legal long code (of say b), and then Property (1) makes sure that A is the long code of a which satisfies $a = \pi(b)$. We will call A and B *consistent* if they satisfy the above properties. The goal is therefore to test if A and B are consistent (or "nearly" consistent) while only making 3 queries in all to A and B.

Assume A and B are consistent, and suppose $A(f) = 1$: then $B(f \circ \pi) = 1$ and also

```
Inner Verifier MBC$^{A,B}$ $(u,\pi)$
    Choose uniformly at random $f \in \mathcal{F}_{[2]^u}$, $g,h \in \mathcal{F}_{[7]^u}$
    if $A(f) = 1$ then accept iff $B(g) = B(g(f \circ \pi \wedge h))$
    if $A(f) = -1$ then accept iff $B(g) = B(g(-f \circ \pi \wedge h))$
```

Figure 3-1: The Inner Verifier based on the Monomial Basis Check of Bellare et al. [7].

for any h $B(f \circ \pi \wedge h) = B(f \circ \pi) \wedge B(h) = 1 \wedge B(h) = 1$. Similarly, if $A(f) = -1$ then $B(-f \circ \pi \wedge h) = 1$ for every h. So far we used only the first two properties of consistent strings A and B. Using also the third, we deduce that if A and B are consistent, then for any $f \in \mathcal{F}_{[2]^u}$ and any $g, h \in \mathcal{F}_{[7]^u}$.

$$\begin{array}{rlll} & A(f) = 1 & \text{implies} & B(g) = B(g(f \circ \pi \wedge h)) \\ \text{and} & A(f) = -1 & \text{implies} & B(g) = B(g(-f \circ \pi \wedge h)) \ . \end{array} \tag{3.1}$$

Checking this condition is essentially the *Monomial Basis Check* (MBC) of Bellare et al. [7], with the minor twist of looking at both A and B (Bellare et al. [7] would instead look only at B, and then test separately whether A is consistent with B). As a first proposal, we consider the test of Figure 3-1, which checks the Condition (3.1) for f, g and h chosen uniformly at random from their domain. Notice that this inner verifier has perfect completeness by construction and also makes only 3 (adaptive) queries. It is possible to prove that the soundness is $3/4$ and therefore the MBC inner verifier is $(1, 3/4, 3)$-good. We omit the (not too hard) analysis, that is based on the techniques of [16]. The following argument shows that the analysis is tight: if A and B are inconsistent long codes then the MBC verifier accepts with probability $3/4$, and our decoding procedure will have success probability zero when working with two inconsistent Long codes.

Since the worst case for the MBC verifier arises when A and B are individually correct but inconsistent, we try to patch the MBC test by adding another test that handles this case well. A good candidate for this "patching" role is the Projection Test, where f is taken uniformly from $\mathcal{F}_{[2]^u}$, g is taken uniformly from $\mathcal{F}_{[7]^u}$, and the test accepts iff $A(f) = B(g)B(g(f \circ \pi))$. Indeed, one can verify that if A and B are inconsistent long codes, then with probability $1/2$ over the choices of f and g the Projection Test rejects. Combining the two verification procedures, we define the BGS$_p$ verifier (see Figure 3-2) that is very similar to one used in [7]. This performs the MBC test with probability p and the Projection test

```
Inner Verifier $\mathrm{BGS}_p{}^{A,B}\ (u,\pi)$
  Choose uniformly at random $f \in \mathcal{F}_{[2]^u}$, $g,h \in \mathcal{F}_{[7]^u}$
  With probability $p$ do
    if $A(f)=1$ then accept iff $B(g)=B(g(f\circ\pi\wedge h))$
    if $A(f)=-1$ then accept iff $B(g)=B(g(-f\circ\pi\wedge h))$
  With probability $1-p$ do
    accept iff $A(f)=B(g)B(g(f\circ\pi))$
```

Figure 3-2: The Inner Verifier that combines Monomial Basis Check and Projection Test.

```
Inner Verifier $\text{B-MBC}_p{}^{A,B}\ (u,\pi)$
  Choose uniformly at random $f \in \mathcal{F}_{[2]^u}$, $g \in \mathcal{F}_{[7]^u}$
  Choose at random $h \in \mathcal{F}_{[7]^u}$ such that $\forall b \in [7]^u$, $\mathbf{Pr}[h(b)=1]=p$
  if $A(f)=1$ then accept iff $B(g)=B(g(f\circ\pi\wedge h))$
  if $A(f)=-1$ then accept iff $B(g)=B(g(-f\circ\pi\wedge h))$
```

```
Inner Verifier $\mathrm{IV3}_\delta{}^{A,B}\ (u,\pi)$
  Set $t=\lceil 1/\delta\rceil$, $\varepsilon_1=\delta^2$ and $\varepsilon_i=\varepsilon_{i-1}^{2c/\varepsilon_{i-1}}$
  Choose $p \in \{\varepsilon_1,\ldots,\varepsilon_t\}$ uniformly at random
  Run $\text{B-MBC}_p{}^{A,B}\ (u,\pi)$.
```

Figure 3-3: The B-MBC$_p$ verifier, a version of the MBC verifier where h is biased, and our final Inner Verifier IV3$_\delta$.

with probability $(1-p)$. It turns out that one can show (and this time the calculations are pretty hard) that it is best to set $p=1/3$ and that $\mathrm{BGS}_{1/3}$ is a $(1,2/3,3)$-good verifier, i.e. the soundness is $2/3$. Again, the analysis can be shown to be tight: no setting of p can result in a verifier with soundness less than $2/3$.

3.2 Obtaining a better 3-query construction

A second look at the BGS$_p$ verifier reveals that the two possible tests to be executed are very related: if we pick $h\equiv -1$ instead that according to the uniform distribution, then the Projection Test coincides with the MBC test[1]. Therefore, we can view BGS$_p$ in the following equivalent way: it first chooses to pick h according to one out of two possible distributions (i.e. either the uniform distribution on $\mathcal{F}_{[7]^u}$ or the deterministic choice of setting $h(b)=-1$

[1]Recall that B is folded — otherwise the claim would not be true.

for every b), then it picks f and g uniformly and performs the MBC test. An alternative approach, that turns out to be much better, is to "shuffle" the distributions, and to skew the distribution of h point-wise. This gives rise to the definition of the B-MBC$_p$ verifier (Figure 3-3, top). The analysis of the BGS$_p$ verifier suggests that it would be good to set $p = 1/6$ in B-MBC$_p$. Instead, it turns out that it is better to have p much smaller. We now see some details of the analysis which will guide us to the right inner verifier construction.

Let A, B, π and p be fixed, and let $X = X_{A,B,\pi,p}$ be the random variable whose value is 1 when B-MBC$_p{}^{A,B}(u,\pi)$ accepts and 0 otherwise[2]. The acceptance probability of B-MBC$_p{}^{A,B}(u,\pi)$ is $\mathbf{E}[X]$, which equals

$$\mathop{\mathbf{E}}_{f,g,h}\left[\left(\frac{(1+A(f))}{2}\right)\left(\frac{(1+B(g)B(g(f\circ\pi\wedge h)))}{2}\right)\right.$$
$$\left.+\left(\frac{(1-A(f))}{2}\right)\left(\frac{(1+B(g)B(g(-f\circ\pi\wedge h)))}{2}\right)\right]$$

Since A is folded, we always have $-A(f) = A(-f)$. Using the linearity of expectation and the fact that f and $-f$ are identically distributed in the uniform distribution, we get $\mathbf{E}[X]$ equals

$$\begin{aligned}
& 2\mathop{\mathbf{E}}_{f,g,h}\left[\left(\tfrac{(1+A(f))}{2}\right)\left(\tfrac{(1+B(g)B(g(f\circ\pi\wedge h)))}{2}\right)\right] \\
= \quad & \tfrac{1}{2}+\tfrac{1}{2}\mathop{\mathbf{E}}_{f,g,h}[B(g)B(g(f\circ\pi\wedge h))] \\
& +\tfrac{1}{2}\mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g(f\circ\pi\wedge h))]
\end{aligned} \tag{3.2}$$

(there would also be a term $\frac{1}{2}\mathop{\mathbf{E}}_{f}[A(f)]$ that we omit since it is zero owing to the foldedness of A).

The expressions arising in (3.2) are extremely hard to bound, but we are fortunate that their analysis already appeared in the literature! Indeed they are the *same* expressions arising in a verifier that Håstad [16] constructs in order to prove a (tight) non-approximability result for satisfiable instances of MAX 3SAT.[3] An equivalent description of the verifier of Håstad is the following: it asks the queries $A(f)$, $B(g)$ and $B(-g(f\circ\pi\wedge h))$, where f, g, h are generated as in B-MBC$_p$, then

[2]The sample space of $X_{A,B,\pi,p}$ is given by the possible choices of f, g, and h.

[3]We are able to provide slightly simpler analyses of these terms than [16] since we allow the projections π to be picked from a broad class of distributions rather than the specific one which [16] uses. But the overall structure of the analysis is the same as that of [16].

- if $A(f) = 1$ it accepts iff $B(g) = -B(-g(f \circ \pi \wedge h))$;
- if $A(f) = -1$ it accepts no matter what is the value of $B(g)$ and $B(-g(f \circ \pi \wedge h))$.

Our goal is to prove that, by choosing p small enough, B-MBC$_p$ will be a $(1, 1/2 + \varepsilon, 3)$-good inner verifier for $\varepsilon > 0$ as small as we desire and then by using the composition theorem we will be done. In order to prove this, we would like to prove that, for any B, the expectation of $B(g)B(g(f \circ \pi \wedge h))$ can be upper bounded by some arbitrarily small constant by choosing p correspondingly small, and that whenever the expectation of $A(f)B(g)B(g(f \circ \pi \wedge h))$ is non-negligible, then the probability of success of the decoding procedure (described in Section 2.3.2) is non-negligible. In order to bound the expectation of $B(g)B(g(f \circ \pi \wedge h))$, however, one cannot fix a particular p, but one has to pick p according to an appropriate distribution; also the bounds hold only if the expectation is also taken over π. Indeed there is a counterexample showing that the expressions of (3.2) cannot be bounded without going through such additional complications.[4] Our final verifier IV3$_\delta$, described in Figure 3-3, is the same as B-MBC$_p$ except for the choice of p. The strange distribution of p is the particular one for which we will be able to bound the expressions of (3.2). The constant c used in the definition of $\varepsilon_1, \ldots, \varepsilon_t$ is an absolute constant which will be left unspecified but can be easily calculated from our proofs.

3.3 Analysis of the 3-Query Protocol

In this section we will prove the following result.

Theorem 3.1 *For any $\delta > 0$, IV3$_\delta$ is a $(1, 1/2 + 3\delta/2, 3)$-good inner verifier.*

Using the above Theorem together with the Composition Theorem 2.6 gives us the main theorem of this chapter:

Theorem 3.2 *For any $\varepsilon > 0$,* $\mathrm{NP} = \mathrm{PCP}_{1,1/2+\varepsilon}[\log, 3]$.

Corollary 3.3 *For any $\varepsilon > 0$,* $\mathrm{NP} = \mathrm{naPCP}_{1,1/2+\varepsilon}[\log, 4]$.

Proof: Observe that IV3$_\delta$ can be viewed as making 4 non-adaptive queries. The result now follows using the Composition Theorem. $\square$.

[4]We thank Johan Håstad for showing us such an example.

3.3.1 Technical Lemmas

We now state and prove the two main lemmas (Lemmas 3.4 and 3.5 below) that will be required in the proof of Theorem 3.1 (which will be presented in Section 3.3.2). Our proofs are based on the proofs of these lemmas in [16] but we rework them as we allow the projections π from a wide class of distributions as opposed to the very specific distribution which [16] uses and it turns out that this in fact simplifies the presentation somewhat. Recall the definition of the decoding procedure Decode(A) from Section 2.3.2: it takes as input a folded string A; it picks a set α with probability $\hat{A}_\alpha^2$ and returns an element of the set α picked uniformly at random.

Lemma 3.4 ([16]) *If $t = \lceil \delta^{-1} \rceil$, $\varepsilon_1 = \delta^2$ and $\varepsilon_i = \varepsilon_{i-1}^{2c/\varepsilon_{i-1}}$ for $1 < i \leq t$, and $p \in \{\varepsilon_1, \cdots, \varepsilon_t\}$ is chosen uniformly at random, then for large enough positive integers u, for all folded $B : \mathcal{F}_{[7]^u} \to \{-1, 1\}$,*

$$\left| \mathop{\mathbf{E}}_{p,\pi,f,g,h} [B(g)B(g \cdot (f \circ \pi \wedge h))] \right| \leq 2\varepsilon_1^{1/2} + \frac{1}{t} \leq 3\delta$$

assuming π is picked according to a smooth distribution. (The parameter p is implicitly used in bias of the random choice of the function h.)

Lemma 3.5 ([16]) *For every $\delta, p > 0$, there exists a constant $\gamma = \gamma_{\delta,p} > 0$, such that for all large enough u, for all folded strings $B : \mathcal{F}_{[7]^u} \to \{1, -1\}$ and $\{A^\pi : \mathcal{F}_{[2]^u} \to \{1, -1\}\}_{\pi \in_R \mathcal{D}(B)}$, if*

$$\left| \mathop{\mathbf{E}}_{\pi,f,g,h} [A^\pi(f)B(g)B(g \cdot (f \circ \pi \wedge h))] \right| \geq \delta,$$

then $\mathbf{Pr}[\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] \geq \gamma$ where the probability is taken over the choice of π from the smooth distribution $\mathcal{D}(B)$ and the coin tosses of Decode. *(Note that the parameter p is implicitly used in bias of the random choice of the function h.)*

The following fact will be very useful in the proofs of the above two lemmas.

Lemma 3.6 *Let $B : \mathcal{F}_{[7]^u} \to \{1, -1\}$ be a folded string and let $\mathcal{D}(B)$ be a smooth distribution on projection functions $\pi : [7]^u \to [2]^u$. Then, for any p, $0 < p < 1$,*

$$\mathop{\mathbf{E}}_{\pi \in_R \mathcal{D}(B)} \Big[\sum_{\beta : |\beta| \geq K} \hat{B}_\beta^2 (1-p)^{|\pi(\beta)|} \Big] \leq \delta$$

provided $K \geq 2^{\Omega(p^{-1}\ln\delta^{-1})}$.

Proof: Since π is picked according to a smooth distribution, we can use Lemma 2.3 and therefore conclude:

$$\begin{aligned}\mathop{\mathbf{E}}_{\pi}\Big[\sum_{\beta:|\beta|\geq K}\hat{B}_\beta^2(1-p)^{|\pi(\beta)|}\Big] &\leq e^{-\sigma\log_7 K/16}+(1-p)^{\sigma\log_7 K/4}\\ &\leq \delta\end{aligned}$$

provided $K \geq 7^{16\sigma^{-1}p^{-1}\ln(2/\delta)} = 2^{\Omega(p^{-1}\ln\delta^{-1})}$ (recall that σ is an absolute constant). □

Proof of Lemma 3.4: Using the Fourier expansions of $B(g)$ and $B(g\cdot(f\circ\pi\wedge h))$ as in Equation (2.2), the properties of linear functions (2.1), and using linearity of expectation, we transform the given expectation, for each fixed p, into

$$\sum_{\beta_1,\beta_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}\mathop{\mathbf{E}}_{\pi,f,g,h}\left[l_{\beta_1\Delta\beta_2}(g)l_{\beta_2}(f\circ\pi\wedge h)\right].$$

Since g is picked uniformly and independently at random, the inner expectation is 0 unless $\beta_1=\beta_2=\beta$. Let us take expectation fixing π also for the moment. For $x\in\pi(\beta)$, we define $\beta_x=\{y\in\beta:\pi(y)=x\}$. We need to compute, for each β with $|\beta|$ odd (we only worry about such β as $\hat{B}_\beta=0$ otherwise by Lemma 2.5)

$$\begin{aligned}\mathop{\mathbf{E}}_{f,g,h}\Big[\prod_{y\in\beta}(f(\pi(y))\wedge h(y))\Big] &= \prod_{x\in\pi(\beta)}\mathop{\mathbf{E}}_{f,h}\Big[\prod_{y\in\beta_x}(f(x)\wedge h(y))\Big]\\ &= \prod_{x\in\pi(\beta)}\Big(\frac{1}{2}+\frac{1}{2}\cdot(2p-1)^{|\beta_x|}\Big)\\ &= \prod_{x\in\pi(\beta)}(-1)^{|\beta_x|}\cdot\Big(\frac{(-1)^{|\beta_x|}}{2}+\frac{(1-2p)^{|\beta_x|}}{2}\Big)\\ &= -\prod_{x\in\pi(\beta)}\Big(\frac{(-1)^{|\beta_x|}}{2}+\frac{(1-2p)^{|\beta_x|}}{2}\Big).\end{aligned}$$

as $|\beta|=\sum_{x\in\pi(\beta)}|\beta_x|$ is odd. For each fixed p the (absolute value of the) expectation we need to estimate thus becomes

$$\left|\mathop{\mathbf{E}}_{\pi}\Big[\sum_{\beta}\hat{B}_\beta^2\prod_{x\in\pi(\beta)}\Big(\frac{(-1)^{|\beta_x|}}{2}+\frac{(1-2p)^{|\beta_x|}}{2}\Big)\Big]\right|. \tag{3.3}$$

One hopes to estimate this sum as a function of p tending to 0 with p. Unfortunately such a estimate does not exist (and one can construct examples where such a statement would be false). It turns out, however, that it is easy to bound the expectation of Equation (3.3) above for small β and large β and this is very useful as it will allow us to vary p as per some distribution so that one can bound (3.3) when we take expectations over p as well. Let $c > 1$ be a small absolute constant to be determined. We have

Claim 1 *For each fixed $p > 0$ and B,*

$$\left| \mathop{\mathbf{E}}_{\pi \in_R \mathcal{D}(B), f, g, h} [B(g)B(g \cdot (f \circ \pi \wedge h))] \right| \leq 2p^{1/2} + \sum_{\beta | p^{-1/2} < |\beta| < p^{-c/p}} \hat{B}_\beta^2 .$$

Proof: We split the sum of Equation (3.3) into three parts depending upon which of the three disjoint intervals $[1, p^{-1/2}]$, $(p^{-1/2}, p^{-c/p})$ and $[p^{-c/p}, \infty)$, $|\beta|$ lies in. The middle interval need not be estimated as it appears on the right hand side of the estimate stated in the Claim.

Let us now consider β with $|\beta|$ small, i.e $|\beta| \leq p^{-1/2}$. Since $|\beta|$ is odd, there must exist $x \in \pi(\beta)$ with $|\beta_x|$ odd (also $|\beta_x| \leq |\beta| \leq p^{-1/2}$). For such an x

$$0 \geq \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2}\right) \geq \frac{1}{2}\left(-1 + (1 - 2p|\beta_x|)\right) = -p|\beta_x| \geq -p^{1/2} . \tag{3.4}$$

Hence, when $|\beta| \leq p^{-1/2}$,

$$\left| \prod_{x \in \pi(\beta)} \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2}\right) \right| \leq p^{1/2}$$

(using (3.4) and the fact that all factors are bounded by 1 in absolute value), and so

$$\sum_{\beta : |\beta| \leq p^{-1/2}} \hat{B}_\beta^2 \prod_{x \in \pi(\beta)} \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2}\right) \leq \sum_{\beta : |\beta| \leq p^{-1/2}} \hat{B}_\beta^2 p^{1/2} \leq p^{1/2} . \tag{3.5}$$

Let us now consider the β's with $|\beta| \geq p^{-c/p}$. For these the relevant expectation to bound is:

$$\mathop{\mathbf{E}}_{\pi}\Big[\sum_{\beta : |\beta| \geq p^{-c/p}} \hat{B}_\beta^2 \prod_{x \in \pi(\beta)} \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2}\right)\Big] \leq \mathop{\mathbf{E}}_{\pi}\Big[\sum_{\beta : |\beta| \geq p^{-c/p}} \hat{B}_\beta^2 (1-p)^{|\pi(\beta)|}\Big]$$

$$\begin{aligned}
&= \sum_{\beta:|\beta|\geq p^{-c/p}} \hat{B}_\beta^2 \mathop{\mathbf{E}}_{\pi}[(1-p)^{|\pi(\beta)|}] \\
&\leq \sum_{\beta:|\beta|\geq p^{-c/p}} \hat{B}_\beta^2 p^{1/2} \\
&\leq p^{1/2}\,. \qquad (3.6)
\end{aligned}$$

where the last but one step follows using Lemma 3.6 together with an appropriate choice of the (absolute) constant $c > 1$.

The statement of our Claim now follows from Equations (3.5) and (3.6). □ (*Claim 1*)

All that remains to be done is to now take expectations over the choice of p as well. Recall that p is drawn uniformly at random from $\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_t\}$ for $t = \lceil \delta^{-1} \rceil$ and where $\varepsilon_i = \varepsilon_{i-1}^{2c/\varepsilon_{i-1}}$ for $1 < i \leq t$. Hence using Claim 1, we have

$$\begin{aligned}
\left| \mathop{\mathbf{E}}_{p,\pi,f,g,h} [B(g)B(g\cdot(f\circ\pi\wedge h))] \right| &\leq \mathop{\mathbf{E}}_{p}[2p^{1/2} + \sum_{\beta:p^{-1/2}<|\beta|<p^{-c/p}} \hat{B}_\beta^2] \\
&\leq 2\varepsilon_1^{1/2} + \frac{1}{t},
\end{aligned}$$

where the last step follows because $\varepsilon_1 > \varepsilon_i$ for $1 < i \leq t$ and because the ranges of $|\beta|$ for the different values of p are disjoint. Since $\varepsilon_1 = \delta^2$ and $1/t \leq \delta$, we have

$$\left| \mathop{\mathbf{E}}_{p,\pi,f,g,h} [B(g)B(g\cdot(f\circ\pi\wedge h))] \right| \leq 3\delta$$

the Lemma follows. □ (*Lemma 3.4*)

Proof of Lemma 3.5: Using the Fourier expansions of $A^\pi(f)$, $B(g)$ and $B(g\cdot(f\circ\pi\wedge h))$ as in Equation (2.2), the properties of linear functions (2.1), and using linearity of expectation, we transform the given expectation, for each fixed π, into

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} \mathop{\mathbf{E}}_{f,g,h} \left[l_\alpha(f) l_{\beta_1 \Delta \beta_2}(g) l_{\beta_2}(f\circ\pi\wedge h) \right].$$

(We have omitted the superscript π on A for notational convenience.)

Since g is picked uniformly and independently at random, the inner expectation is 0 unless $\beta_1 = \beta_2 = \beta$ (say). Since f is also picked uniformly and independently at random, we can also conclude $\alpha \subseteq \pi(\beta)$. Using this, our expression simplifies, once a π is fixed, to

$$\begin{aligned} \left| \mathop{\mathbf{E}}_{f,g,h} \left[A(f)B(g)B(g \cdot (f \circ \pi \wedge h)) \right] \right| &= \left| \sum_{\beta,\ \alpha \subseteq \pi(\beta)} \hat{A}_\alpha \hat{B}_\beta^2 \mathop{\mathbf{E}}_{f,h} \left[l_\alpha(f) l_\beta(f \circ \pi \wedge h) \right] \right| \\ &\le \left| \sum_{\beta,\ \alpha \subseteq \pi(\beta)} |\hat{A}_\alpha| \hat{B}_\beta^2 \mathop{\mathbf{E}}_{f,h} \left[l_\alpha(f) l_\beta(f \circ \pi \wedge h) \right] \right| \qquad (3.7) \end{aligned}$$

We proceed to simplify this expression further:

$$\begin{aligned} \sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} |\hat{A}_\alpha| \hat{B}_\beta^2 \mathbf{E} \left[l_\alpha(f) l_\beta(f \circ \pi \wedge h) \right] &= \sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} |\hat{A}_\alpha| \hat{B}_\beta^2 \prod_{x \in \alpha} \mathbf{E} \left[f(x) \prod_{y \in \beta_x} (f(x) \wedge h(y)) \right] \\ &\qquad \prod_{x \in \pi(\beta) \setminus \alpha} \mathbf{E} \left[\prod_{y \in \beta_x} (f(x) \wedge h(y)) \right] \\ &= \sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} |\hat{A}_\alpha| \hat{B}_\beta^2 \prod_{x \in \alpha} \left(\frac{1}{2} + \frac{-(2p-1)^{|\beta_x|}}{2} \right) \\ &\qquad \prod_{x \in \pi(\beta) \setminus \alpha} \left(\frac{1}{2} + \frac{(2p-1)^{|\beta_x|}}{2} \right) \\ &= -\sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} \hat{B}_\beta^2 |\hat{A}_\alpha| \prod_{x \in \alpha} \left(\frac{(-1)^{|\beta_x|}}{2} - \frac{(1-2p)^{|\beta_x|}}{2} \right) \\ &\qquad \prod_{x \in \pi(\beta) \setminus \alpha} \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2} \right) \qquad (3.8) \end{aligned}$$

where the last step follows since $|\beta| = \sum_{x \in \pi(\beta)} |\beta_x|$ is odd if $\hat{B}_\beta \neq 0$. Define $h(\alpha, \beta)$ to be the quantity

$$\prod_{x \in \alpha} \left(\frac{(-1)^{|\beta_x|}}{2} - \frac{(1-2p)^{|\beta_x|}}{2} \right) \prod_{x \in \pi(\beta) \setminus \alpha} \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2} \right)$$

It is not difficult to see that

$$\begin{aligned} \sum_{\alpha \subseteq \pi(\beta)} h^2(\alpha, \beta) &= \prod_{x \in \pi(\beta)} \left[\left(\frac{(-1)^{|\beta_x|}}{2} - \frac{(1-2p)^{|\beta_x|}}{2} \right)^2 + \left(\frac{(-1)^{|\beta_x|}}{2} + \frac{(1-2p)^{|\beta_x|}}{2} \right)^2 \right] \\ &\le (1-p)^{|\pi(\beta)|} \qquad . \qquad (3.9) \end{aligned}$$

The last step follows from the fact that if $|a|, |b| \le 1-p$ and $|a|+|b| = 1$, then $a^2+b^2 \le (1-p)$. It is also easy to see that

$$\sum_{\alpha \subseteq \pi(\beta)} |h(\alpha,\beta)| = 1. \tag{3.10}$$

Hence for each fixed π and p, we have

$$\begin{aligned}
\left|\mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g(f\circ\pi\wedge h))]\right| &\le \left|\sum_{\beta,\ \alpha\subseteq\pi(\beta)} |\hat{A}_\alpha|\hat{B}_\beta^2 \mathop{\mathbf{E}}_{f,h}\left[l_\alpha(f)l_\beta(f\circ\pi\wedge h)\right]\right| \text{ (using (3.7))}\\
&\le \sum_{\substack{\beta\\ \alpha\subseteq\pi(\beta)}} \hat{B}_\beta^2\Big(|\hat{A}_\alpha||h(\alpha,\beta)|\Big) \text{ (using (3.8))}\\
&\le \sum_{\beta:|\beta|\ge K} \hat{B}_\beta^2\Big(\sum_{\alpha\subseteq\pi(\beta)}\hat{A}_\alpha^2\Big)^{1/2}\Big(\sum_{\alpha\subseteq\pi(\beta)} h^2(\alpha,\beta)\Big)^{1/2}+\\
&\quad + \sum_{\substack{\beta:|\beta|\le K\\ \alpha\subseteq\pi(\beta),|\hat{A}_\alpha|\le\delta/4}} \hat{B}_\beta^2|\hat{A}_\alpha||h(\alpha,\beta)|+\\
&\quad + \sum_{\substack{\beta:|\beta|\le K\\ \alpha\subseteq\pi(\beta),|\hat{A}_\alpha|\ge\delta/4}} \hat{B}_\beta^2|\hat{A}_\alpha||h(\alpha,\beta)|\\
&\le \sum_{\beta:|\beta|\ge K} \hat{B}_\beta^2\Big(\sum_{\alpha\subseteq\pi(\beta)} h^2(\alpha,\beta)\Big)^{1/2} + \frac{\delta}{4}\sum_{\beta:|\beta|\le K}\hat{B}_\beta^2 +\\
&\quad +\frac{4}{\delta}\sum_{\substack{\beta:|\beta|\le K\\ \alpha\subseteq\pi(\beta),|\hat{A}_\alpha|\ge\delta/4}} \hat{A}_\alpha^2\hat{B}_\beta^2 \text{ (using Equation 3.10))}\\
&\le \sum_{\beta:|\beta|\ge K}\hat{B}_\beta^2(1-p)^{|\pi(\beta)|/2} + \frac{\delta}{4} + \frac{4}{\delta}\sum_{\substack{\beta:|\beta|\le K\\ \alpha\subseteq\pi(\beta)}}\hat{A}_\alpha^2\hat{B}_\beta^2 \qquad (3.11)
\end{aligned}$$

where the last step follows using (3.9). We now take expectations over the projection π. Using Lemma 3.6 we conclude

$$\mathop{\mathbf{E}}_{\pi}\Big[\sum_{\beta:|\beta|\ge K}\hat{B}_\beta^2(1-p)^{|\pi(\beta)|/2}\Big] \le \frac{\delta}{4}, \tag{3.12}$$

provided $K = 2^{\Omega(p^{-1}\ln\delta^{-1})}$. Such a choice of K is possible provided $7^u \ge K$, and we assume that u is large enough for this purpose. Now, combining (3.11) and (3.12), together with the hypothesis of the Lemma that $\left|\mathop{\mathbf{E}}_{\pi,f,g,h}[A^\pi(f)B(g)B(g\cdot(f\circ\pi\wedge h))]\right| \ge \delta$, we get

$$\mathop{\mathbf{E}}_{\pi}\Big[\sum_{\substack{\beta:|\beta|\le K\\ \alpha\subseteq\pi(\beta)}}\hat{A}_\alpha^2\hat{B}_\beta^2\Big] \ge \frac{\delta}{4}\cdot\frac{\delta}{2} = \frac{\delta^2}{8}\,. \tag{3.13}$$

We now estimate the probability of success of the decoding strategy.

$$\begin{aligned}\Pr_{\pi,\text{coin tosses of Decode}}[\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] &\geq \mathop{\mathbf{E}}_{\pi}[\sum_{\substack{\alpha,\beta \\ \alpha\cap\pi(\beta)\neq\emptyset}} \hat{A}_\alpha^2 \hat{B}_\beta^2 \frac{1}{|\alpha|}\frac{1}{|\beta|}] \\ &\geq \mathop{\mathbf{E}}_{\pi}[\sum_{\substack{\beta:\beta\leq K \\ \alpha\subseteq\pi(\beta)}} \hat{A}_\alpha^2 \hat{B}_\beta^2 \frac{1}{K^2}] \,. \quad (3.14)\end{aligned}$$

(We have used $\hat{A}_\emptyset = 0$ in the last step above.)

Combining (3.13) and (3.14) we get

$$\Pr_{\pi,\text{coin tosses of Decode}}[\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] \geq \frac{\delta^2}{8K^2}. \quad (3.15)$$

Recalling that $K = 2^{\Omega(p^{-1}\ln\delta^{-1})}$, the success probability depends only on p, δ, and the proof is complete. □ (*Lemma 3.5*)

3.3.2 Proof of Theorem 3.1

The Proof: The statements about query complexity and completeness are clear, we verify the soundness claim. Recall that the probability of acceptance of $\mathrm{IV3}_\delta$ is

$$\frac{1}{2} + \frac{1}{2}\mathop{\mathbf{E}}_{p,f,g,h}[B(g)B(g(f\circ\pi\wedge h))] + \frac{1}{2}\mathop{\mathbf{E}}_{p,f,g,h}[A(f)B(g)B(g(f\circ\pi\wedge h))].$$

By Lemma 3.4,

$$\left|\mathop{\mathbf{E}}_{\pi,p,f,g,h}[B(g)B(g\cdot(f\circ\pi\wedge h))]\right| \leq 3\delta$$

and hence if $\mathbf{Pr}[V(A^\pi, B, \pi)\text{ accepts}] \geq 1/2 + 3\delta/2 + \gamma$, we must have

$$\left|\mathop{\mathbf{E}}_{\pi,p,f,g,h}[A(f)B(g)B(g(f\circ\pi\wedge h))]\right| \geq 2\gamma \quad (3.16)$$

and invoking Lemma 3.5, we can conclude that $\Pr_{\pi}[\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] \geq \eta$ for some $\eta > 0$ for all large enough u. (Note that in (3.16) the expectation is also taken over p, while Lemma 3.5 works with a fixed p. But clearly if (3.16) holds there exists a p such that the condition for Lemma 3.5 is satisfied and we can then work with that particular p.) This verifies the soundness condition of $\mathrm{IV3}_\delta$ and concludes the proof that $\mathrm{IV3}_\delta$ is a $(1, 1/2 + 3\delta/2, 3)$-good inner verifier. □

Chapter 4

The 5-Query PCP Constructions

In this chapter we describe extensions of our 3-query PCP construction for slightly higher number of queries. We will first describe the construction of an adaptive 5-query inner verifier and then obtain some non-adaptive PCP constructions based on it.

4.1 The Inner Verifier Construction

We now construct an inner verifier that makes 5 queries. The way to exploit the additional queries at our disposal is to query the purported long code A at two functions $f_1, f_2 \in \mathcal{F}_{[2]^u}$ instead of just a single $f \in \mathcal{F}_{[2]^u}$. In the 3 query protocol, the rationale used was that if $f(a) = 1$ then $(f \wedge h)(a) = 1$ for any h and similarly for the case when $f(a) = -1$. Similarly if $f_1(a) = f_2(a) = 1$, then we must have $(f_1 \wedge h)(a) = 1$ and $(f_2 \wedge h)(a) = 1$ for any h, and these two tests can be performed (instead of just one test $(f \wedge h)(a) = 1$ as was done in the 3 query protocol). One might expect that such a test will achieve a soundness of $(1/2)^2 = 1/4$ by making 5 adaptive queries, and indeed this can be shown to be the case using the same ananlysis technqiues presented in the previous Chapter. construction follows the same idea of recycling one bit between two repetitions of a basic test as in [27]). We, however, now present and analyze a different 5-query test that gives no improvement (in soundness) over the above-mentioned test for the case of 5 adaptive queries, but which has other applications (in construction of non-adaptive verifiers) that the originally suggested one does not. We will use as basis for our test the following fact: if $f_1(a) = f_2(a) = 1$, then $(f_1 \wedge f_2 \wedge h)(a) = (f_1 \wedge -f_2 \wedge h)(a) = (-f_1 \wedge f_2 \wedge h)(a)$ for any h. Thus, we will able to perform two equality tests while reading five bits, so one expects

```
Inner Verifier B-V5_p^{A,B} (u, π)
  Choose uniformly at random $f_1, f_2 \in \mathcal{F}_{[2]^u}$, $g \in \mathcal{F}_{[7]^u}$
  Choose at random $h \in \mathcal{F}_{[7]^u}$ such that $\forall b \in [7]^u$ $\mathbf{Pr}[h(b) = 1] = p$
  Let
    $G_1 = g \cdot (f_1 \circ \pi \wedge f_2 \circ \pi \wedge h)$,
    $G_2 = g \cdot (f_1 \circ \pi \wedge -f_2 \circ \pi \wedge h)$,
    $G_3 = g \cdot (-f_1 \circ \pi \wedge f_2 \circ \pi \wedge h)$,
    $G_4 = g \cdot (-f_1 \circ \pi \wedge -f_2 \circ \pi \wedge h)$.
  if $A(f_1) = 1$ and $A(f_2) = 1$ accept iff $B(G_1) = B(G_2) = B(G_3)$
  if $A(f_1) = 1$ and $A(f_2) = -1$ accept iff $B(G_1) = B(G_2) = B(G_4)$
  if $A(f_1) = -1$ and $A(f_2) = 1$ accept iff $B(G_1) = B(G_3) = B(G_4)$
  if $A(f_1) = -1$ and $A(f_2) = -1$ accept iff $B(G_2) = B(G_3) = B(G_4)$
```

```
Inner Verifier IV5_δ^{A,B} (u, π)
  Set $t = \lceil \delta^{-1} \rceil$, $\varepsilon_1 = \delta^2$ and $\varepsilon_i = \varepsilon_{i-1}^{2c/\varepsilon_{i-1}}$ for $1 < i \le t$.
  Choose $p \in \{\varepsilon_1, \cdots, \varepsilon_t\}$ uniformly at random.
  Run B-V5_p^{A,B} (u, π)
```

Figure 4-1: The adaptive 5-query inner verifier with a fixed p and the final inner verifier.

that the soundness should be $(1/2)^2 = 1/4$ and indeed we shall prove that to be the case. In the actual protocol, however, f and h will belong to different spaces and this will be handled using the *projection function* π as in the construction of our 3-query inner verifier, and moreover, since we would like all queried bits (functions) to be unbiased, we will also xor these functions with an unbiased function g. This yields the inner verifier of Figure 4-1. As in the case of the 3 query protocol, we once again need to pick the bias p at random from a set of different possibilities for the analysis to work. This gives us the final inner verifier $\mathrm{IV5}_\delta$ of Figure 4-1.

4.2 Analysis of the soundness

We now analyze the soundness of the inner verfier $\mathrm{IV5}_\delta$. Let Y denote the indicator random variable for the acceptance of $\mathrm{IV5}_\delta$, clearly we have

$$\begin{aligned} Y &= \Big(\frac{1+A(f_1)}{2}\Big)\Big(\frac{1+A(f_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_3)}{2}\Big) \\ &\quad +\Big(\frac{1+A(f_1)}{2}\Big)\Big(\frac{1-A(f_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_4)}{2}\Big) \end{aligned}$$

$$+\Big(\frac{1-A(f_1)}{2}\Big)\Big(\frac{1+A(f_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_3)}{2}\Big)\Big(\frac{1+B(G_1)B(G_4)}{2}\Big)$$
$$+\Big(\frac{1-A(f_1)}{2}\Big)\Big(\frac{1-A(f_2)}{2}\Big)\Big(\frac{1+B(G_2)B(G_3)}{2}\Big)\Big(\frac{1+B(G_2)B(G_4)}{2}\Big)$$

The probability that B-V5$_p$ accepts, given input (u, π) and oracle access to (A, B), is simply the expectation of Y over the choices of f_1, f_2, g, h, and this expectation equals:

$$\begin{aligned}\mathop{\mathbf{E}}_{f_1,f_2,g,h}[Y] &= \frac{1}{4}+\frac{1}{8}\mathbf{E}\Big[B(G_1)B(G_2)+B(G_1)B(G_3)+B(G_1)B(G_4)+B(G_2)B(G_3)+\\ &\qquad +B(G_2)B(G_4)+B(G_3)B(G_4)\Big]\\ &\quad +\frac{1}{8}\mathbf{E}\Big[A(f_1)B(G_1)B(G_2)-A(f_1)B(G_3)B(G_4)+A(f_2)B(G_1)B(G_3)-\\ &\qquad -A(f_2)B(G_2)B(G_4)\Big]\\ &\quad +\frac{1}{8}\mathbf{E}\Big[A(f_1)A(f_2)B(G_2)B(G_3)-A(f_1)A(f_2)B(G_1)B(G_4)\Big] \qquad (4.1)\end{aligned}$$

The following sequence of (simple) lemmas will now help us simplify the above expression.

Lemma 4.1 *If f_1, f_2, f are picked uniformly at random from $\mathcal{F}_{[2]^u}$, g u.a.r from $\mathcal{F}_{[7]^u}$ and $h \in \mathcal{F}_{[7]^u}$ is chosen at random so that* $\Pr[h(b) = 1] = p\ \forall b \in [7]^u$, *then*

$$\mathop{\mathbf{E}}_{f_1,f_2,g,h}[B(G_1)B(G_2)] = \mathop{\mathbf{E}}_{f,g,h}[B(g)B(g\cdot(f\circ\pi\wedge h))]$$

Proof: Note that G_1 is unbiased (i.e $\Pr[G_1(b) = 1] = 1/2\ \forall b \in [7]^u$), and $G_2 = G_1 \cdot (f_1 \circ \pi \wedge f_2 \circ \pi \wedge h) \cdot (f_1 \circ \pi \wedge -f_2 \circ \pi \wedge h) = G_1 \cdot (f_1 \circ \pi \wedge h)$ (after some boolean algebra). Thus the distribution of G_1 and G_2 when f_1, f_2, g, h are picked as specified is the same as the distribution of g and $g \cdot (f \circ \pi \wedge h)$ when f, g, h picked randomly as specified. Hence the above expectations are equal. □

For the same reason, we also have

$$\mathop{\mathbf{E}}_{f_1,f_2,g,h}[B(G_1)B(G_3)] = \mathop{\mathbf{E}}_{f_1,f_2,g,h}[B(G_2)B(G_4)] = \mathop{\mathbf{E}}_{f_1,f_2,g,h}[B(G_3)B(G_4)] = \mathop{\mathbf{E}}_{f,g,h}[B(g)B(g\cdot(f\circ\pi\wedge h))] \qquad (4.2)$$

Lemma 4.2 *If f_1, f_2, f are picked uniformly at random from $\mathcal{F}_{[2]^u}$, g u.a.r from $\mathcal{F}_{[7]^u}$ and*

$h \in \mathcal{F}_{[7]^u}$ *is chosen at random so that* $\Pr[h(b) = 1] = p\ \forall b \in [7]^u$, *then*

$$\mathop{\mathbf{E}}_{f_1,f_2,g,h}[B(G_1)B(G_4)] = \mathop{\mathbf{E}}_{f_1,f_2,g,h}[B(G_2)B(G_3)] = \mathop{\mathbf{E}}_{f,g,h}[B(g)B(g \cdot (f \circ \pi \wedge h))]$$

Proof: We argue about the expectation of $B(G_1)B(G_4)$, the other expectation being clearly the same as this one. As before G_1 is an unbiased function on $[7]^u$ and has the same distribution as g, and $G_4 = G_1 \cdot (f_1 \circ \pi \wedge f_2 \circ \pi \wedge h) \cdot (-f_1 \circ \pi \wedge -f_2 \circ \pi \wedge h) = G_1 \cdot (-(f_1 f_2) \circ \pi \wedge h)$. Now f_1, f_2 are picked u.a.r from $\mathcal{F}_{[2]^u}$, so the function $-f_1 \cdot f_2$ also is a random function from $\mathcal{F}_{[2]^u}$ therefore G_4 has the same distribution as $g \cdot (f \circ \pi \wedge h)$, and hence $B(G_1)B(G_4)$ has the same distributions as $B(g)B(g \cdot (f \circ \pi \wedge h))$. $\square$

Lemma 4.3 *If* f_1, f_2, f *are picked uniformly at random from* $\mathcal{F}_{[2]^u}$, g *u.a.r from* $\mathcal{F}_{[7]^u}$ *and* $h \in \mathcal{F}_{[7]^u}$ *is chosen at random so that* $\Pr[h(b) = 1] = p\ \forall b \in [7]^u$, *then*

- $\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)B(G_1)B(G_2)] = \mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_2)B(G_1)B(G_3)] = \mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g \cdot (f \circ \pi \wedge h))]$
- $\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)B(G_3)B(G_4)] = \mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_2)B(G_2)B(G_4)] = \mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g \cdot (-f \circ \pi \wedge h))]$

Proof: We consider the expectation of $A(f_1)B(G_1)B(G_2)$, the other expectations can be handled similarly. The proof is similar to those of the previous two lemmas: f_1 and G_1 are unbiased and $G_2 = G_1 \cdot (f_1 \circ \pi \wedge h)$ and hence the distribution of f_1, G_1, G_2 is the same as the distribution of $f, g, (f \circ \pi \wedge h)$, and the result follows. $\square$

Since $A : \mathcal{F}_{[2]^u} \to \{1, -1\}$ is folded, $A(-f_1) = -A(f_1)$ and hence

$$\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_2)B(G_3)] = -\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_1)B(G_4)], \tag{4.3}$$

and for a similar reason

$$\mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g \cdot (-f \circ \pi \wedge h))] = -\mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g \cdot (f \circ \pi \wedge h))]. \tag{4.4}$$

Now Equation (4.1) together with (4.3) and (4.4) and Lemmas 4.1, 4.2, and 4.3, implies

$$\mathop{\mathbf{E}}_{f_1,f_2,g,h}[Y] \;=\; \frac{1}{4} + \frac{3}{4}\mathop{\mathbf{E}}_{f,g,h}[B(g)B(g \cdot (f \circ \pi \wedge h))] + \frac{1}{2}\mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g \cdot (f \circ \pi \wedge h))]$$

$$-\frac{1}{4}\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(g\cdot(f_1\circ\pi\wedge f_2\circ\pi\wedge h))B(g\cdot(-f_1\circ\pi\wedge -f_2\circ\pi\wedge h))] \qquad (4.5)$$

The second and third terms above are ones which we recognize from the analysis of the 3-query protocol (Equation (3.2), and we handled them using Lemmas 3.4 and 3.5 respectively). We now proceed to handle the last term by stating and proving a lemma very similar to Lemma 3.5.

Lemma 4.4 *For every $\delta, p > 0$, there exists a constant $\gamma = \gamma_{\delta,p} > 0$, such that for all large enough u, for all folded strings $B : \mathcal{F}_{[7]^u} \to \{1,-1\}$ and $\{A^\pi : \mathcal{F}_{[2]^u} \to \{1,-1\}\}_{\pi\in_R\mathcal{D}(B)}$, if*

$$\left|\mathop{\mathbf{E}}_{\pi,f_1,f_2,g,h}[A^\pi(f_1)A^\pi(f_2)B(G_1)B(G_4)]\right| \geq \delta,$$

then $\mathbf{Pr}[\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] \geq \gamma$ where the probability is taken over the choice of π from the smooth distribution $\mathcal{D}(B)$ and the coin tosses of Decode. *(Note that the parameter p is implicitly used in the bias in the random choice of the function h.)*

Proof: The proof closely follows that of Lemma 3.5, and in fact after a certain point we will complete our proof by just resorting to the proof of Lemma 3.5. Using the Fourier expansions of $A(f_i)$ and $B(G_j)$ as in Equation 2.2 and the properties of linear functions (2.1), we can transform the given expectation, for each fixed π, into

$$\sum_{\alpha_1,\alpha_2,\beta_1,\beta_2}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}\mathop{\mathbf{E}}_{f_1,f_2,g,h}\Big[l_{\alpha_1}(f_1)l_{\alpha_2}(f_2)l_{\beta_1\Delta\beta_2}(g)l_{\beta_1}(f_1\circ\pi\wedge f_2\circ\pi\wedge h)l_{\beta_2}(-f_1\circ\pi\wedge -f_2\circ\pi\wedge h)\Big].$$

Since g is picked uniformly and independently at random, the inner expectation is 0 unless $\beta_1 = \beta_2 = \beta$ (say). Since f_1 and f_2 are also picked uniformly and independently at random, we can also conclude $\alpha_1, \alpha_2 \subseteq \pi(\beta)$. Some boolean algebra yields $(f_1\circ\pi\wedge f_2\circ\pi\wedge h)\cdot(-f_1\circ\pi\wedge -f_2\circ\pi\wedge h) = (-f_1f_2\circ\pi\wedge h)$. Incorporating all this, our expression simplifies to

$$\sum_{\substack{\beta\\ \alpha_1,\alpha_2\subseteq\pi(\beta)}}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_\beta^2\mathop{\mathbf{E}}_{f_1,f_2,h}\Big[l_{\alpha_1}(f_1)l_{\alpha_2}(f_2)l_\beta((-f_1f_2)\circ\pi\wedge h)\Big]\,. \qquad (4.6)$$

Since $f_1(x)$ and $f_2(x)$ are chosen independently for different values of x, the inner expectation can be written as a product of expectations, one for each $x \in \pi(\beta)$. If there exists $x_0 \in \alpha_1 - \alpha_2$, then the expectation

$$E_{x_0} = \mathop{\mathbf{E}}_{f_1,f_2,h}\left[f_1(x_0) \prod_{y \in \beta_{x_0}} (-f_1(x_0)f_2(x_0) \wedge h(y))\right]$$

occurs as one of the factors of the inner expectation, and since this expectation E_{x_0} can easily be seen to be 0, the original expectation will be 0 as well. This implies that all non-zero terms in the summation (4.6) have $\alpha_1 \subseteq \alpha_2$, and similarly we must have $\alpha_2 \subseteq \alpha_1$ as well. Thus we have $\alpha_1 = \alpha_2$ whenever the inner expectation in (4.6) is non-zero. This further simplifies our expression to:

$$\sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} \hat{A}_\alpha^2 \hat{B}_\beta^2 \mathop{\mathbf{E}}_{f_1,f_2,h} \left[l_\alpha(f_1 f_2) l_\beta(-f_1 f_2 \circ \pi \wedge h)\right] . \tag{4.7}$$

Now since $-f_1 \cdot f_2$ is a uniformly distributed function in $\mathcal{F}_{[2]^u}$, and also $|\alpha|$ is odd for $\hat{A}_\alpha \neq 0$, the above simplifies to

$$- \sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} \hat{A}_\alpha^2 \hat{B}_\beta^2 \mathop{\mathbf{E}}_{f,h} \left[l_\alpha(f) l_\beta(f \circ \pi \wedge h)\right] . \tag{4.8}$$

Hence for each fixed π we have

$$\begin{aligned}
\left| \mathop{\mathbf{E}}_{f_1,f_2,g,h} \left[A^\pi(f_1)A^\pi(f_2)B(G_1)B(G_4)\right] \right| &= \left| - \sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} \hat{A}_\alpha^2 \hat{B}_\beta^2 \mathop{\mathbf{E}}_{f,h} \left[l_\alpha(f) l_\beta(f \circ \pi \wedge h)\right] \right| \\
&\leq \left| \sum_{\substack{\beta \\ \alpha \subseteq \pi(\beta)}} |\hat{A}_\alpha| \hat{B}_\beta^2 \mathop{\mathbf{E}}_{f,h} \left[l_\alpha(f) l_\beta(f \circ \pi \wedge h)\right] \right| .
\end{aligned}$$

since $|\hat{A}_\alpha| \leq 1$ implies $\hat{A}_\alpha^2 \leq |\hat{A}_\alpha|$. Comparing with Equation (3.7) of the proof of Lemma 3.5, we are now in the same position as we were in the proof of Lemma 3.5, and we can complete the proof using exactly the same approach as used there. □

We are now ready to prove the claimed soundness for the inner verifier AIV5_δ.

Theorem 4.5 *For any $\delta > 0$,* AIV5_δ *is a $(1, \frac{1}{4} + \frac{9\delta}{4}, 5)$-good inner verifier.*

Proof: The statements about query complexity and completeness are clear, we verify the soundness claim. By Lemma 3.4,

$$\left| \mathop{\mathbf{E}}_{\pi,p,f,g,h} [B(g)B(g \cdot (f \circ \pi \wedge h))] \right| \leq 3\delta$$

and now using equation (4.5), we have

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\pi,p,f_1,f_2,g,h} [V(A^\pi, B, \pi)\text{accepts}] &\leq \frac{1}{4} + \frac{9\delta}{4} + \frac{1}{2} \left| \mathop{\mathbf{E}}_{\pi,p,f,g,h} [A(f)B(g)B(g \cdot (f \circ \pi \wedge h))] \right| \\ &\quad + \frac{1}{4} \left| \mathop{\mathbf{E}}_{\pi,p,f,g,h} [A(f_1)A(f_2)B(G_1)B(G_4)] \right| \end{aligned}$$

and hence if $\mathop{\mathbf{Pr}}_{\pi,\text{random tosses of } V} [V(A^\pi, B, \pi)\text{accepts}] \geq 1/4 + 9\delta/4 + \gamma$, then either

$$\left| \mathop{\mathbf{E}}_{\pi,p,f,g,h} [A(f)B(g)B(g \cdot (f \circ \pi \wedge h))] \right| \geq \gamma ,$$

or

$$\left| \mathop{\mathbf{E}}_{\pi,p,f,g,h} [A(f_1)A(f_2)B(G_1)B(G_4)] \right| \geq 2\gamma .$$

Therefore we can use Lemmas 3.5 and 4.4 to conclude that $\mathop{\mathbf{Pr}}_{\pi} [\mathsf{Decode}(A^\pi) = \pi(\mathsf{Decode}(B))] \geq \eta$, where $\eta = \eta_{\gamma,\delta} > 0$ for all large enough u. This establishes the soundness property of the verifier AIV5_δ, completing the proof that AIV5_δ is a $(1, 1/4 + 9\delta/4, 5)$-good inner verifier.
□

Theorem 4.6 *For any $\varepsilon > 0$,*

$$\mathrm{PCP}_{1,1/4+\varepsilon}[\log, 5] = \mathrm{NP} .$$

Proof: Follows from the previous Lemma and the Composition Theorem 2.6. □

4.3 Constructions of Non-adaptive PCPs

We now consider the problem of constructing non–adaptive PCPs with soundness strictly less than 1/2. We have already seen in Theorem 3.3 that a soundness arbitrarily close to 1/2 can be achieved by PCPs that make 4 non–adaptive queries. We are going to prove in this section that 5 non–adaptive queries suffice to achieve a soundness strictly less than 1/2. The previous best known construction implicit in the work of Håstad [16] required 9 queries.

If we only require near-perfect completeness, a construction making 5 non–adaptive queries and having soundness 1/4 is known [27]; proving a similar result with perfect completeness turns out to be more complicated. It is known that three queries do not suffice to achieve a soundness less than 1/2 (even for PCPs with near-perfect completeness); it reamins unknown whether four queries will suffice or whether our bound of five queries is in fact optimal.

4.3.1 A construction with 6 non–adaptive queries

Theorem 4.7 *For any $\varepsilon > 0$,*

$$\mathrm{naPCP}_{1,1/4+\varepsilon}[\log, 6] = \mathrm{NP} \ .$$

Proof: Observe that by reading all the four bits $B(G_i)$ for $1 \le i \le 4$, the inner verifer $\mathrm{AIV5}_\delta$ reads only 6 bits non–adaptively. Hence $\mathrm{AIV5}_\delta$ is a $(1, 1/4 + 9\delta/4, 6)$-good non–adaptive inner verifer, and appealing to the Composition Theorem once again, we obtain the desired PCP. □

Remark: The above theorem implies that, for any $\varepsilon > 0$, NP has PCPs with $3+\varepsilon$ amortized non–adaptive query bits even with perfect completeness. The best known bound prior to our work was $3/\lg(4/3) + \varepsilon = 7.2282625 + \varepsilon$ amortized query bits implicit in [16]. Note that Theorem 3.3 itself proves that $4 + \varepsilon$ amortized query bits are achievable. If one does not require perfect completeness, a construction achieving $1.5 + \varepsilon$ amortized query bits has been obtained recently [27, 24]. Our result implies that approximating *satisfiable instances* of k-CSP to within a factor of $2^{\lfloor k/3 \rfloor - \varepsilon}$ is NP-hard, for any $\varepsilon > 0$.

4.3.2 A construction with 5 non–adaptive queries

We modify the inner verifer $\mathrm{B\text{-}V5}_p$ so that it does not read the bit $B(G_4)$ and does not perform any test that involves $B(G_4)$. Let us call the new inner verifier $\mathrm{B\text{-}NAV5}_p$, which is shown in Figure 4-2.

In the three cases when either $A(f_1) \neq 1$ or $A(f_2) \neq 1$ (or both), $\mathrm{B\text{-}NAV5}_p$ performs only one test, while when $A(f_1) = A(f_2) = 1$ it performs two tests. This implies that the soundness of $\mathrm{B\text{-}NAV5}_p$ is at least $3/4 \times 1/2 + 1/4 \times 1/4 = 7/16$ (which is the probability of accepting random, but inconsistent, long codes). We will now show, using the same

```
Inner Verifier B-NAV5$_p$ $^{A,B}$ $(u,\pi)$
    Choose uniformly at random $f_1, f_2 \in \mathcal{F}_n$, $g \in \mathcal{F}_m$
    Choose at random $h \in \mathcal{F}_m$ such that $\forall b \in \{-1,1\}^m. \mathbf{Pr}[h(b)=1]=p$
    Set
        $G_1 = g \cdot (f_1 \circ \pi \wedge f_2 \circ \pi \wedge h)$
        $G_2 = g \cdot (f_1 \circ \pi \wedge -f_2 \circ \pi \wedge h)$
        $G_3 = g \cdot (-f_1 \circ \pi \wedge f_2 \circ \pi \wedge h)$
        if $A(f_1)=1$ and $A(f_2)=1$ accept iff $B(G_1)=B(G_2)$ and $B(G_1)=B(G_3)$.
        if $A(f_1)=1$ and $A(f_2)=-1$ accept iff $B(G_1)=B(G_2)$.
        if $A(f_1)=-1$ and $A(f_2)=1$ accept iff $B(G_1)=B(G_3)$.
        if $A(f_1)=-1$ and $A(f_2)=-1$ accept iff $B(G_2)=B(G_3)$.
```

```
Inner Verifier NAIV5$_\delta^{A,B}$ $(u,\pi)$
    Set $t=\lceil \delta^{-1} \rceil$, $\varepsilon_1 = \delta^2$ and $\varepsilon_i = \varepsilon_{i-1}^{2c/\varepsilon_{i-1}}$ for $1 < i \leq t$.
    Choose $p \in \{\varepsilon_1, \cdots, \varepsilon_t\}$ uniformly at random.
    Run B-NAV5$_p$$^{A,B}$ $(u,\pi)$
```

Figure 4-2: The non-adaptive 5-query inner verifier with a fixed p and the final inner verifier NAIV5$_\delta$.

techniques as that of Section 4.2, that this bound is achieved, provided we use the inner verifier NAIV5$_\delta$ (see Figure 4-2) that runs B-NAV5$_p$ with p chosen according to a certain distribution.

Analysis of Soundness:

The arithmetization of the above test yields the following expression Z, which equals 1 if the test accepts, and equals 0 otherwise:

$$\begin{aligned} Z \;=\; & \Big(\frac{1+A(f_1)}{2}\Big)\Big(\frac{1+A(f_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_3)}{2}\Big) \\ & +\Big(\frac{1+A(f_1)}{2}\Big)\Big(\frac{1-A(f_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_2)}{2}\Big) \\ & +\Big(\frac{1-A(f_1)}{2}\Big)\Big(\frac{1+A(f_2)}{2}\Big)\Big(\frac{1+B(G_1)B(G_3)}{2}\Big) \\ & +\Big(\frac{1-A(f_1)}{2}\Big)\Big(\frac{1-A(f_2)}{2}\Big)\Big(\frac{1+B(G_2)B(G_3)}{2}\Big) \end{aligned}$$

The probability that this inner verifier B-NAV5$_p$ accepts, given input (u,π) and oracle access to (A,B), is therefore just the expectation of Z over the choices of f_1, f_2, g, h, and this expectation equals:

$$
\begin{aligned}
\mathop{\mathbf{E}}_{f_1,f_2,g,h}[Z] \quad = \quad & \frac{7}{16} - \frac{1}{16}\mathbf{E}\left[A(f_1) + A(f_2) + A(f_1)A(f_2)\right] \\
& + \frac{3}{16}\mathbf{E}\left[B(G_1)B(G_2) + B(G_1)B(G_3) + B(G_2)B(G_3)\right] \\
& + \frac{3}{16}\mathbf{E}\left[A(f_1)B(G_1)B(G_2) + A(f_2)B(G_1)B(G_3)\right] + \\
& + \frac{3}{16}\mathbf{E}\left[A(f_1)A(f_2)B(G_2)B(G_3)\right] \qquad (4.9) \\
& - \frac{1}{16}\mathbf{E}\Big[A(f_1)B(G_1)B(G_3) + A(f_1)B(G_2)B(G_3) + \\
& \qquad + A(f_2)B(G_1)B(G_2) + A(f_2)B(G_2)B(G_3)\Big] \\
& - \frac{1}{16}\mathbf{E}\left[A(f_1)A(f_2)B(G_1)B(G_2) - A(f_1)A(f_2)B(G_1)B(G_3)\right]
\end{aligned}
$$

Since A is folded, we have $\mathbf{E}[A(f_1)] = \mathbf{E}[A(f_2)] = 0$ and since f_1, f_2 are chosen independently, $\mathbf{E}[A(f_1)A(f_2)] = \mathbf{E}[A(f_1)] \times \mathbf{E}[A(f_2)] = 0$. Following the approach of Lemma 4.3, one can prove that

$$
\begin{aligned}
\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)B(G_1)B(G_3)] &= \mathop{\mathbf{E}}_{f_1,f_2,g',h}[A(f_1)B(g')B(g' \cdot (f_2 \circ \pi \wedge h))] \\
&= \mathop{\mathbf{E}}_{f_1}[A(f_1)] \times \mathop{\mathbf{E}}_{f_2,g',h}[B(g')B(g' \cdot (f_2 \circ \pi \wedge h))] \\
&= 0 .
\end{aligned}
$$

since f_1 is chosen independent of f_2, g', h. For exactly the same reason,

$$
\mathbf{E}[A(f_1)B(G_2)B(G_3)] = \mathbf{E}[A(f_2)B(G_1)B(G_2)] = \mathbf{E}[A(f_2)B(G_2)B(G_3)] = 0 .
$$

Arguing as in Lemma 4.3 again, we get

$$
\begin{aligned}
\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_1)B(G_2)] &= \mathop{\mathbf{E}}_{f_1,f_2,g',h}[A(f_1)A(f_2)B(g')B(g' \cdot (f_1 \circ \pi \wedge h))] \\
&= \mathop{\mathbf{E}}_{f_2}[A(f_2)] \mathop{\mathbf{E}}_{f_1,g',h} A(f_1)B(g')B(g' \cdot (f_1 \circ \pi \wedge h))] \\
&= 0,
\end{aligned}
$$

and similarly

$$
\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_1)B(G_3)] = 0 .
$$

Also, since A is folded, we clearly have:

$$\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_2)B(G_3)] = -\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_1)B(G_4)] \ .$$

Combining these observations with Lemmas 4.1, 4.2 and 4.3, we get, going back to Equation (4.9),

$$\begin{aligned}\mathop{\mathbf{E}}_{f_1,f_2,g,h}[Z] \ \leq \ & \frac{7}{16} + \frac{9}{16}\left|\mathop{\mathbf{E}}_{f,g,h}[B(g)B(g\cdot(f\circ\pi\wedge h))]\right| + \frac{6}{16}\left|\mathop{\mathbf{E}}_{f,g,h}[A(f)B(g)B(g\cdot(f\circ\pi\wedge h))]\right| \\ & + \frac{3}{16}\left|\mathop{\mathbf{E}}_{f_1,f_2,g,h}[A(f_1)A(f_2)B(G_1)B(G_4)]\right| \qquad (4.10)\end{aligned}$$

Consider now our final inner verifier NAIV5$_\delta$ defined in Figure 4-2. Using the above Equation (4.10) and Lemmas 3.4, 3.5 and 4.4, we can prove, exactly as we did in Lemma 4.5, that

Lemma 4.8 *For any* $\delta > 0$, NAIV5$_\delta$ *is a* $(1, 7/16 + 27\delta/16, 5)$*-good non-adaptive inner verifier.*

Theorem 4.9 *For any* $\varepsilon > 0$,

$$\text{naPCP}_{1,7/16+\varepsilon}[\log, 5] = \text{NP} \ .$$

Chapter 5

Some Results on Free Bit Complexity

Free bits are an extremely important parameter measuring the complexity of a PCP construction as it has direct applications to showing hardness of approximating Vertex Cover, as is formalized in the following Proposition:

Propostion 5.1 ([7]) *If* NP $\subseteq$ FPCP$_{c,s}[\log, f]$, *then approximating Vertex Cover up to a factor of* $\frac{2f-s}{2f-c} - \varepsilon$ *is* NP*-hard for any* $\varepsilon > 0$.

5.1 Free bits and our PCP constructions

5.1.1 The Vertex Cover hardness revisited

The best hardness of approximation result known for vertex cover is a factor of $7/6 - \varepsilon$, for any $\varepsilon > 0$, and is obtained using Proposition 5.1 together with the two free bit PCP construction of Håstad [16], which has completeness $1 - \varepsilon$ and soundness $1/2$, for any $\varepsilon > 0$. We now prove that we can achieve the same soundness while also guaranteeing perfect completeness, thereby answering in the affirmative a question raised in [7].

Theorem 5.2 *For any* $\varepsilon > 0$, *we have* NP $\subseteq$ FPCP$_{1,1/2+\varepsilon}[\log, 2]$.

Proof: We just observe that the inner verifier IV3$_\delta$ which we used to get our 3-query PCP construction uses just two free bits. This is because, once $A(f)$ and $B(g)$ are read, the verifier "knows" the values it expects for the other bits it reads. The theorem now follows using the soundness bound we proved for our 3-query PCP in Theorem 3.1. □

Plugging the above into Proposition 5.1, we are therefore able to match the best known hardness result for approximating Vertex Cover while only using perfect completeness, indicating that imperfect completeness is not inherent at least for the current best inapproximability result for Vertex Cover.

5.1.2 Some other constructions

Using the transformation of PCP systems as given Proposition 11.9 of [7], we also get the following corollary:

Theorem 5.3 *For any $\varepsilon > 0$, we have*

$$\mathrm{NP} \subseteq \mathrm{FPCP}_{1,3/4+\varepsilon}[\log, \lg 3].$$

Theorems 5.2 and 5.5 above can be contrasted with that of Trevisan [26] on PCP classes defined in terms of non-adaptive free bits collapsing to P, which states that, for all $\varepsilon > 0$,

$$\begin{aligned} \mathrm{naFPCP}_{1,1/2-\varepsilon}[\log, 2] &\subseteq \mathrm{P}, \text{ and} \\ \mathrm{naFPCP}_{1,3/4-\varepsilon}[\log, \lg 3] &\subseteq \mathrm{P}\ . \end{aligned}$$

The free bit complexity of the verifier $\mathrm{IV3}_\delta$ is 3 when viewed in the non-adaptive sense (there are at most 8 satisfying assignments to the boolean predicate tested by $\mathrm{IV3}_\delta$), and hence we also get:

Theorem 5.4 *For any $\varepsilon > 0$, we have* $\mathrm{NP} \subseteq \mathrm{naFPCP}_{1,1/2+\varepsilon}[\log, 3]$.

Finally we get the following result from our 5-query PCP construction:

Theorem 5.5 *For any $\varepsilon > 0$, we have* $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,1/4+\varepsilon}[\log, 3]$.

Proof: Follows from the fact that the inner verifier $\mathrm{IV5}_\delta$ uses only 3 free bits (in the adaptive sense). □

5.2 Difficulty of Proving Bound better than 3/2 for Vertex Cover

This section addresses the question of obtaining significantly better inapproximability results for Vertex Cover based on PCP constructions. In particular we hint at a potential difficulty that has to be overcome in extending current PCP constructions to prove a hardness factor of better than 3/2 for approximating Vertex Cover. Rather than as a negative result, we view this result as pointing out that new type of constructions should be conceived in order to further improve the inapproximability bounds for Vertex Cover, and also indicating what minimal property such a construction must possess. By Proposition 5.1, PCPs that use $\lg 3$ free bits can at best prove the hardness of approximating Vertex Cover within $3/2 - \varepsilon$ (this result can be attained by showing $\mathrm{NP} \subseteq \mathrm{FPCP}_{1-\varepsilon,\varepsilon}[\log, 2]$). To get better inapproximability results, one therefore needs to give good PCP constructions that use at most one free bit. To get the best bounds, by Proposition 5.1, it is conceivable that the PCP will have near-perfect completeness $(1 - \varepsilon)$, and we prove below that any such PCP must be significantly different from existing constructions, as its query complexity can not be a simple constant (as is the case with existing constructions exploiting near-perfect completeness), but, in fact, must grow at least as $\varepsilon^{-1/3}$.

Theorem 5.6 *For all $\varepsilon > 0$, assuming $\mathrm{P} \neq \mathrm{NP}$, any PCP system that captures NP that uses only one free bit and has completeness $1 - \varepsilon$, must make $\Omega(\varepsilon^{-1/3})$ queries if it must have any soundness s that is strictly bounded away from 1.*

Proof: Let a PCP verifier make at most k queries on any sequence of its random bits and decide membership in an NP-complete language L with completeness $(1 - \varepsilon)$ and soundness $s < 1$. We want to prove a lower bound of $\Omega(\varepsilon^{-1/3})$ on k. For any outcome R of its random coin tosses, it accepts if and only if a certain boolean formula f_R is satisfied, and since the verifier uses only one free bit, the formula f_R is the disjunction of two *inconsistent* terms T_R^1 and T_R^2 each of which has at most k literals (since the verifier makes at most k queries). Since T_R^1 and T_R^2 are inconsistent, there must exist a query x_i such that $x_i \in T_R^1$ and $\bar{x}_i \in T_R^1$. By the assumption about the parameters of the PCP, if $x \in L$, at least a $(1 - \varepsilon)$ fraction of the functions f_R are satisfiable, while if $x \notin L$, at most a fraction s of the functions f_R are satisfiable. We replace the function f_R by the set of constraints $S_R = S_R^1 \cup S_R^2$ where

$$S_R^1 = \left\{x_i \Rightarrow l : l \text{ is a literal in } T_R^1, l \neq x_i\right\},$$

$$S_R^2 = \left\{\bar{x}_i \Rightarrow l : l \text{ is a literal in } T_R^2, l \neq \bar{x}_i\right\}.$$

Assume, without loss of generality that S_R^1 and S_R^2 each have exactly k implication constraints (this may be achieved by repeating certain constraints or adding dummy variables and constraints which can always be satisfied). Clearly if f_R is satisfied, all the constraints in S_R are satisfied too, and even if f_R is not satisfied all constraints in either S_R^1 or S_R^2 can be satisfied. Thus, if a fraction $1 - \varepsilon$ of the functions f_R are satisfiable, then at least a fraction

$$\frac{2k(1-\varepsilon) + k\varepsilon}{2k} = 1 - \frac{\varepsilon}{2}$$

of the 2SAT constraints in $\bigcup_R S_R$ are satisfiable.

Now, consider running Zwick's algorithm [30] for almost-satisfiable 2SAT instances on the 2SAT instance comprising of clauses in $\bigcup_R S_R$. If a fraction $1 - \varepsilon/2$ of the clauses are satisfiable (as will be the case when $x \in L$), then the algorithm will find an assignment that satisfies at least a fraction $1 - O(\varepsilon^{1/3})$ of the 2SAT clauses. In the case when $x \notin L$, at most a fraction

$$\frac{s \cdot 2k + (1-s)(2k-1)}{2k} = 1 - \frac{(1-s)}{2k}$$

of the 2SAT clauses are simultaneously satisfiable. Thus, if $1 - \frac{(1-s)}{2k} < 1 - O(\varepsilon^{1/3})$, we can decide membership in L in polynomial time by running Zwick's algorithm on the 2SAT instance with clauses in $\cup_R S_R$, and accepting only those strings x for which the algorithm finds an assignment satisfying more than a $1 - \frac{(1-s)}{2k}$ fraction of the 2SAT constrains. Since L is an NP-complete language, this is impossible assuming P $\neq$ NP, and thus we must have

$$1 - \frac{(1-s)}{2k} \geq 1 - O(\varepsilon^{1/3}),$$

or,

$$k \geq \Omega(\frac{(1-s)}{2} \cdot \varepsilon^{-1/3}) = \Omega(\varepsilon^{-1/3})$$

implying that the PCP must read at least $\Omega(\varepsilon^{-1/3})$ bits. □

5.3 A gadget based PCP construction with $\lg 3$ free bits

We investigated PCPs that use only one free bit in the previous section. We now consider what can be achieved with $\lg 3$ free bits (which is the next higher number of free bits possible). We prove the following:

Theorem 5.7 *For any $\varepsilon > 0$,*

$$\mathrm{NP} \subseteq \mathrm{naFPCP}_{1-\varepsilon, 5/6}[\log, \lg 3].$$

Proof: The construction is obtained from Hastad's 3-query PCP [16] which simply checks if the parity of the 3 bits it reads is a certain value 0 or 1 (depending upon the bits it reads). This construction has free-bit complexity 2 because a 3-parity constraint has four satisfying assignments. We now give a "gadget" which will express a 3-parity constraint in terms of 1-ONE constraints (the 1-ONE constraint is a 3-ary boolean constraint defined by 1-ONE(p, q, r) is true iff *exactly* one of the literals p, q, r is true). The gadget is: replace the constraint $x \oplus y \oplus z = 1$ (here we treat boolean functions as 0-1 functions in the usual way) by the three clauses 1-ONE(x, a, b), 1-ONE(y, b, c), and 1-ONE(z, c, a), where a, b, c are auxiliary variables specific to this single parity constraint. It is easy to see that if the original parity constraint was satisfied, then all the three 1-ONE constraints can be satisfied by assigning appropriate values to a, b and c, where as if the parity constraint was violated, no assignment to a, b and c can satisfy more than two 1-ONE clauses. In the terminology of [28], this is a *3-gadget.*

Now consider again the proof system of Håstad [16] that has completeness $(1 - \varepsilon)$ and soundness $1/2$ (actually it has soundness $1/2 + \varepsilon$, but assume that the soundness is actually $1/2$ by unconditionally rejecting with a certain small probability), and whose verifier always checks the 3-parity of the three bits it reads. Now, assume that the proof of the above system is modified to include, in addition to the original proof, also the assignment to the auxiliary variables a, b, c above for *each* triple of bits in the original proof (note that this blows up the size of the proof only by a polynomial factor). Modify the verifier as follows: instead of reading three bits x_1, x_2 and x_3 and checking if $x_1 \oplus x_2 \oplus x_3 = 1$, it picks one of the three 1-ONE clauses of the above gadget (for the particular triple x_1, x_2, x_3) at random, reads the corresponding three bits and checks that the values indeed satisfy 1-ONE.

The query complexity of the new verifier is still 3, but it has free bit complexity only $\lg 3$ (even in the non–adaptive sense), as any constraint it checks is a 1-ONE constraint which has only three satisfying assignments. The completeness is at least $(1-\varepsilon)\cdot 1+\varepsilon\cdot\frac{2}{3} = 1-\varepsilon/3$, while a false proof is rejected with probability at least $\frac{1}{2}\cdot\frac{1}{3} = \frac{1}{6}$, since with probability $1/2$ a parity constraint that is violated is picked (because of the soundness of the original verifier), and with probability $1/3$ a violation is caught through the gadget. The proof system thus has soundness $5/6$. $\square$

Chapter 6

Weakness Results for PCPs

Our PCP constructions so far are powerful enough to capture all of NP while only making few queries into the proof and yet having very good soundness. In this chapter, we proceed to prove the complementary results that PCP classes with certain query complexity and error probability are *weak* in the sense that they can only capture languages in P. We achieve this goal by providing approximation algorithms for constraint satisfaction problems and then invoking the following results which connect the power of PCPs with the approximability of the MAX kCSP problem. Recall that MAX kCSP is the constraint satisfaction problem where all constraints are k-ary boolean functions, and the goal is to find a truth assignment to the underlying variables that maximizes the number of constraints satisfied.

Fact 1 ([1]) *If there exists a polynomial time factor r approximation algorithm for satisfiable instances of MAX kCSP, then $\mathsf{naPCP}_{1,s}[\log, k] \subseteq \mathrm{P}$ for any $s < r$.*

Fact 2 ([25]) *If there exists a polynomial time factor r approximation algorithm for MAX kCSP, then $\mathrm{PCP}_{c,s}[\log, k] \subseteq \mathrm{P}$ for any $s/c < r$.*

Existing approximation algorithms in [25, 26, 29] for various MAX kCSP problems therefore imply that $\mathrm{PCP}_{c,s}[\log, 3] \subseteq \mathrm{P}$ for any $s/c < 1/2$, $\mathsf{naPCP}_{1,s}[\log, 3] \subseteq \mathrm{P}$ for any $s < 5/8$, $\mathrm{PCP}_{c,s}[\log, k] \subseteq \mathrm{P}$ for any $s/c < 2^{1-k}$, and lastly $\mathsf{naPCP}_{1,s}[\log, k] \subseteq \mathrm{P}$ for any $s < (k+1)/2^k$.

We use the above two results connecting constraint satisfaction problems with limitations of PCPs to limit the the power of k-query PCPs for all values of k, for the special cases of perfect and near-perfect completeness.

6.1 Weakness of PCPs with perfect completeness

Theorem 6.1 *For an* $\varepsilon > 0$, $\mathrm{PCP}_{1,3/(2^k+1)-\varepsilon}[\log, k] \subseteq \mathrm{P}$.

Proof: Consider a PCP verifier V for language L that makes k (possibly adaptive) queries, has perfect completeness, has soundness less than $3/(2^k+1)$. Our goal is to show that L is in P. We prove this by constructing in polynomial time, for the case when $x \in L$, an explicit proof that will be accepted by V with probability at least $3/(2^k+1)$. Note that this yields the following polynomial time procedure to decide membership of x in L: construct the afore-mentioned proof in polynomial time (if construction of the proof fails, then we know that $x \notin L$, so reject x immediately) and then accept x iff V accepts this proof with probability at least $3/(2^k+1)$ (since V has logarithmic randomness this can be also be done in polynomial time by enumerating over all possible random strings of V).

The construction of such a proof when $x \in L$ (which is accepted by V with probability at least $3/(2^k+1)$) is achieved by considering a random proof and a proof constructed by finding a satisfying assignment to an appropriately constructed satisfiable 2SAT instance, and taking the "better" of the two proofs. The details of the construction follow.

For any particular choice R of the random bits of V, the computation of V can be modeled as a decision tree T_R of depth at most k. Let m_1 (respectively m_2) denote the fraction of random strings R for which T_R has exactly one accepting "leaf" (respectively exactly two accepting leaves). Let $m_3 = 1 - m_1 - m_2$ be the fraction of R's for which T_R has at least three accepting leaves.

It is easy to see that the probability that a decision tree T_R with exactly l accepting leaves accepts a random proof equals $\frac{l}{2^k}$, and hence the probability that V accepts a random proof is at least $\frac{(m_1+2m_2+3m_3)}{2^k}$. (This is because each decision tree of V has the property that configurations corresponding to two distinct leaves can never occur simultaneously: and hence the events corresponding to the various accepting leafs of a particular tree occurring are mutually exclusive.)

For the case when $x \in L$, consider the following proof: For each tree T_R with just one accepting leaf, create unary clauses consisting of the (at most k) literals which when set to true will make T_R accept (these literals will correspond to the variables on the path from the root to the unique accepting leaf in T_R). For each tree T_R with exactly two accepting leaves, there must exist a variable x such that both the left and right subtrees of the tree rooted at x have exactly one accepting leaf. Create unary clauses as before for the variables (other than x) which occur in the path from the root of T_R to x, and if $l_1, l_2, \ldots, l_p$ and $r_1, r_2, \ldots, r_q$ are the literals which when set to true will make T_R accept in the cases when $x = 0$ and $x = 1$ respectively, create the (at most $2k$) clauses $\bar{x} \Rightarrow l_1, \ldots, \bar{x} \Rightarrow l_p$ and $x \Rightarrow r_1, \ldots, x \Rightarrow r_q$.

Since V has perfect completeness and $x \in L$, the 2SAT formula $\mathcal{C}$ comprising all clauses so created must be satisfiable, and using the polynomial time algorithm for 2SAT, a satisfying assignment σ for $\mathcal{C}$ can be found. Now, let Π be the proof that agrees with σ on all variables (bits) occurring in the clauses of $\mathcal{C}$, and arbitrary elsewhere. Clearly, the probability that V accepts Π is at least the probability that V chooses an R such that T_R has at most 2 accepting leaves, which is $m_1 + m_2$.

Combining the bounds on acceptance probability for a random proof and Π, we get that, in the case when $x \in L$, we can, in polynomial time, construct a proof that is accepted by V with probability at least

$$\min_{m_1,m_2,m_3:m_1+m_2+m_3=1} \max\left\{m_1 + m_2, \frac{m_1 + 2m_2 + 3m_3}{2^k}\right\} = \frac{3}{2^k + 1} \,. \quad \square$$

6.2 Weakness of PCPs with near-perfect completeness

Theorem 6.2 *We have, for any $\varepsilon > 0$, $\mathrm{PCP}_{1-\varepsilon,s}[\log, k] \subseteq \mathrm{P}$, where $s = \frac{3}{2^k+2} - O(\frac{k\varepsilon^{1/3}}{2^k+2})$.*

Proof: The idea is the same as the above proof, the only difference is that the 2SAT formula $\mathcal{C}$ created as above will not in general be satisfiable since V no longer has perfect completeness. If V has completeness $1 - \varepsilon$, however, it is easy to check that at least a fraction $1 - \varepsilon/(m_1 + m_2)$ of the clauses in $\mathcal{C}$ will be satisfiable, and hence we can run the polynomial time algorithm of Zwick [29] for nearly satisfiable instances of 2SAT to find an assignment σ satisfying at least $1 - O((\varepsilon/(m_1 + m_2))^{1/3})$ of the clauses in $\mathcal{C}$. As before, considering a random proof and a proof Π constructed so that it agrees with σ on all its variables and is arbitrary elsewhere, will imply, after a few simple calculations, that the soundness of the verifier cannot be less than

$$\min_{m_1,m_2,m_3:m_1+m_2+m_3=1} \max\left\{m_1+m_2-O(k\varepsilon^{1/3}), \frac{m_1 + 2m_2 + 3m_3}{2^k}\right\} = \frac{3}{2^k + 2} - O(\frac{k\varepsilon^{1/3}}{2^k + 2}). \,. \quad \square$$

Chapter 7

Concluding Remarks

We considered the problem of getting the best possible soundness out of a PCP construction that makes a small number of queries and in addition has perfect completeness. The work of Håstad [16], in addition to providing some striking constructions of PCPs, also developed a general analysis technique based on Discrete Fourier Analysis to bound the soundness of PCP constructions, and in principle reduced the task of obtaining better PCPs to construction of good inner verifiers without worrying too much about how they can be analyzed tightly.[1] This work was the primary driving force behind our work.

One of our principal results is that perfect completeness and a soundness arbitrarily close to $1/2$ can be achieved by 3-query PCPs. This somewhat surprising result is tight in that one cannot achieve a better soundness, and also demonstrates the power of adaptive queries in PCP constructions. In fact it demonstrates that adaptivity gives strictly more power to PCP verifiers as one cannot achieve a soundness smaller than $5/8$ using 3 non-adaptive queries and having perfect completeness.

We also used our techniques to give improved (but not necessarily tight) PCP constructions for a slightly higher number of queries. For example we prove that with 5 queries one can achieve a soundness of $1/4 + \varepsilon$; the best previous construction achieved the same soundness [27], but as was the case with Håstad's 3-query PCP, it too resorted to near-perfect completeness. Thus, we are, once again, able to match the previously best known soundness, but in addition achieving perfect completeness by resorting to adaptive queries. This raised questions about the power of adaptivity in general, and in particular whether

[1]It is however not true that any inner verifier construction can be analyzed tightly now without difficulty, but at least in principle the techniques to do so seem to exist now while they were unknown earlier.

they can always be used to do away with near-perfect completeness in PCP constructions while still achieving the same (or even nearly same) soundness.

While our principal PCP constructions are adaptive, we are able to extract non-adaptive PCPs out of our constructions which are also better than previously known constructions. For example our main 3-query result also implies a soundness of $1/2 + \varepsilon$ can be attained by making 4 non-adaptive queries. We are also able to modify our 5-query construction slightly to conclude that a soundness of $7/16 + \varepsilon$ is achievable using just 5 non-adaptive queries. This result makes significant progress on the question of determining the smallest number of queries required by a non-adaptive PCP with perfect completeness to achieve a soundness strictly less than 1/2. Our result shows that 5 queries suffice, while the previous best bound was 9 queries implicit in the work of [16].

Our result also has some implications for PCP constructions with a small number of free bits. In particular, we obtain that $\mathrm{NP} \subseteq \mathrm{FPCP}_{1,1/2+\varepsilon}[\log, 2]$, which can be used to derive the (already known) hardness result of approximating Vertex Cover within $7/6 - \varepsilon$, but only resorting to perfect completeness. This shows that near-perfect completeness is not inherent at least to the current inapproximability result for Vertex Cover.

We are also able to deduce some non-trivial values of q for which $q+1$ queries are strictly more powerful than q queries. In order to be able to do this, we design an approximation algorithm for Max 4CSP, the constraint satisfaction problem where all constraints are boolean functions of (at most) four variables, using the semidefinite programming methodology of Karloff and Zwick [17, 29]. While the exact performance guarantee can be expressed in terms of the minimum of a complicated function over a certain range (the function involves the volume function of spherical tetrahedra), it is very hard to bound analytically. We have run numerical programs to estimate this minimum and based on the results are able to conclude the following containment result for 4-query PCPs.[2]

Theorem 7.1 *There is a polynomial time factor 0.33-approximation algorithm for* Max 4CSP.

Corollary 7.2 *For any c, s such that $s/c < 0.33$,* $\mathrm{PCP}_{c,s}[\log, 4] \subseteq \mathrm{P}$.

[2]The statement is not really a "theorem" since we do not have an analytical proof of it. We thank Uri Zwick for providing us with some of the programs we used to obtain this result.

Based on the above and our PCP constructions, we are able to exhibit some non-trivial values of q for which $q+1$ queries add more power to PCPs compared to q-query PCPs.

- For non-adaptive PCP, 4 queries are stronger than 3. (This follows from Corollary 3.3 and the fact that $\text{naPCP}_{1,s}[\log, 3] \subseteq \text{P}$ for $s < 5/8$ [26, 29].)
- For adaptive PCP, 5 queries are stronger than 4. (This follows from Theorem 4.6 and Corollary 7.2.)
- 5 non-adaptive queries are stronger than 4 adaptive queries. (This follows from Corollary 7.2 with $c = 1 - \varepsilon$ and the fact – proved in [27] – that $\text{NP} \subseteq \text{PCP}_{1-\varepsilon,1/4}[\log, 5]$ for any $\varepsilon > 0$.)

Open Questions: There are plenty of related questions which still remain unanswered and it is not clear how much current techniques will have to be modified or new techniques invented to answer some of them. We list some of the most prominent questions below:

1. What is the best soundness achievable by non-adaptive 3 query PCPs with perfect completeness? It is known that the soundness has to be at least $5/8$ and a construction with soundness $3/4 + \varepsilon$ is also known. It is not clear at this moment which of these two (or if any of the two) is tight, though our guess seems to be that one can achieve a soundness of $5/8 + \varepsilon$ with 3 non-adaptive queries and perfect completeness.
2. Can one achieve a soundness better than $1/2$ using 4 queries, even allowing for near-perfect completeness? What about the same question if in addition the queries are restricted to be non-adaptive?
3. Can one construct 2 free-bit PCPs with near-perfect completeness and soundness less than $1/2$? Such a construction appears difficult, but it will be a very important development as it would improve the current inapproximability result for Vertex Cover.
4. Asking questions of a more general nature, is it true that adaptive queries strictly add more power to q-query PCPs for all $q \geq 3$? Is it true that an additional query always helps, i.e are $q+1$ query PCPs more powerful than q query PCPs for all $q \geq 2$?

Bibliography

[1] S. ARORA, C. LUND, R. MOTWANI, M. SUDAN, AND M. SZEGEDY. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proceedings of FOCS'92.*

[2] S. ARORA AND S. SAFRA. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proceedings of FOCS'92.*

[3] S. ARORA AND M. SUDAN. Improved low degree testing and its applications. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997.

[4] L. BABAI, L. FORTNOW, L. LEVIN, AND M. SZEGEDY. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[5] L. BABAI, L. FORTNOW, AND C. LUND. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Preliminary version in *Proceedings of FOCS'90.*

[6] L. BABAI, L. FORTNOW, L. LEVIN AND M. SZEGEDY. Checking Computations in polylogarithmic time. *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, 1991, pp. 21-31.

[7] M. BELLARE, O. GOLDREICH, AND M. SUDAN. Free bits, PCP's and non-approximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. Preliminary version in *Proceedings of FOCS'95.*

[8] M. BELLARE, S. GOLDWASSER, C. LUND, AND A. RUSSELL. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th ACM*

Symposium on Theory of Computing, pages 294–304, 1993. See also the errata sheet in *Proceedings of STOC'94*.

[9] M. BELLARE AND M. SUDAN. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 184–193, 1994.

[10] U. FEIGE. A threshold of $\ln n$ for approximating Set Cover. *Journal of the ACM*, 52(4):634–652, 1998.

[11] U. FEIGE, S. GOLDWASSER, L. LOVÁSZ, S. SAFRA, AND M. SZEGEDY. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proceedings of FOCS'91*.

[12] U. FEIGE AND L. LOVÁSZ. Two-prover one round proof systems: Their power and their problems. *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 1992, pp. 733-744.

[13] U. FEIGE AND J. KILIAN. Two prover protocols - low error at affordable rates. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 172–183, 1994.

[14] V. GURUSWAMI, D. LEWIN, M. SUDAN AND L. TREVISAN. A tight characterization of NP with 3-query PCPs. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998. Also available as *ECCC Technical Report* TR98-034, 1998.

[15] J. HÅSTAD. Clique is hard to approximate within $n^{1-\varepsilon}$. *ECCC Technical Report TR97-038*. (Preliminary versions in *Proceedings of FOCS '96 and STOC'96*).

[16] J. HÅSTAD. Some optimal inapproximability results. Technical Report TR97-037, Electronic Colloquium on Computational Complexity, 1997. Preliminary version in *Proceedings of STOC'97*.

[17] H. KARLOFF AND U. ZWICK. A $(7/8-\varepsilon)$-approximation algorithm for MAX 3SAT? In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, 1997.

[18] D. LAPIDOT AND A. SHAMIR. Fully parallelized multi-prover protocols for NEXP-time. *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, 1991, pp. 13-18.

[19] C. LUND AND M. YANNAKAKIS. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960-981, 1994.

[20] C. H. PAPADIMITRIOU AND M. YANNAKAKIS. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43 (1991), pp. 425-440.

[21] R. RAZ. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. Preliminary version in *Proceedings of STOC'95.*

[22] R. RAZ AND S. SAFRA. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997.

[23] A. SAMORODNITSKY AND L. TREVISAN. A PCP construction with $1+\varepsilon$ amortized query bits. Personal Communication, February 1999.

[24] M. SUDAN AND L. TREVISAN. Probabilistically Checkable Proofs with low amortized query complexity. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998. Also available as *ECCC Technical Report* TR98-040, 1998.

[25] L. TREVISAN. Positive linear programming, parallel approximation, and PCP's. In *Proceedings of the 4th European Symposium on Algorithms*, pages 62–75. LNCS 1136, Springer-Verlag, 1996.

[26] L. TREVISAN. Approximating satisfiable satisfiability problems. In *Proceedings of the 5th European Symposium on Algorithms*, pages 472–485. LNCS 1284, Springer-Verlag, 1997.

[27] L. TREVISAN. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998.

[28] L. TREVISAN, G. SORKIN, M. SUDAN AND D. WILLIAMSON. Gadgets, Approximation, and Linear Programming. *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 1996.

[29] U. ZWICK. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[30] U. Zwick. Approximating almost-satisfiable formulas. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998.