Proof Verification and the Hardness of Approximation Problems

SANJEEV ARORA* CARSTEN LUND† RAJEEV MOTWANI‡
MADHU SUDAN§ MARIO SZEGEDY¶

Abstract

We show that every language in NP has a probablistic verifier that checks membership proofs for it using logarithmic number of random bits and by examining a *constant* number of bits in the proof. For strings in the language, there exists a proof such that the verifier always accepts. For instances not in the language, the verifier rejects every provided "proof" with probability at least 1/2. Our result builds upon and improves a recent result of Arora and Safra [5] whose verifiers examine a nonconstant number of bits in the proof (though this number is a very slowly growing function of the input length).

As a consequence we prove that no MAX SNP-hard problem has a polynomial time approximation scheme, unless NP=P. The class MAX SNP was defined by Papadimitriou and Yannakakis [76] and hard problems for this class include vertex cover, maximum satisfiability, maximum cut, metric TSP, Steiner trees and shortest superstring. We also improve upon the clique hardness results of Feige, Goldwasser, Lovasz, Safra and Szegedy [40], and Arora and Safra [5] and shows that there exists a positive ϵ such that approximating the maximum clique size in an N-vertex graph to within a factor of N^ϵ is NP-hard.

1 Introduction

Classifying optimization problems according to their computational complexity is a central endeavor in theoretical computer science. The theory of NP-completeness,

*arora@cs.princeton.edu. Part of this work was done when this author was a student at the University of California at Berkeley, supported by NSF PYI Grant CCR 8896202 and an IBM fellowship. Currently supported by an NSF CAREER award and an Alfred P. Sloan Fellowship.

†lund@research.att.com. AT&T Bell Labs, 600 Mountain Ave, Murray Hill, NJ 07974.

‡rajeev@cs.stanford.edu. Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by Alfred P. Sloan Research Fellowship, IBM Faculty Partnership Award, NSF grant CCR-9010517, NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

§madhu@watson.ibm.com. IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. Part of this work was done when this author was a student at the University of California at Berkeley, supported by NSF PYI Grant CCR 8896202.

¶ms@research.att.com. AT&T Bell Labs, 600 Mountain Ave, Murray Hill, NJ 07974.

developed by Cook [34], Karp [64] and Levin [70], shows that many decision problems of interest, such as satisfiability, are *NP-complete.* This theory also applies to the even more interesting class of optimization problems, such as the traveling salesman problem, bin packing, graph coloring, and maximum clique, by showing that natural decision versions of these problems are also NP-complete, implying that the optimization versions are *NP-hard.* If P $\neq$ NP, no polynomial-time algorithm can solve these problems optimally.

Given this evidence of intractability, researchers have attempted to discover polynomial-time *approximation algorithms* for the NP-hard optimization problems [51, 75]. An algorithm is said to *approximate* a problem *within a factor* c, where $c \geq 1$, if it computes, for every instance of the problem, a solution whose cost (or value) is within a factor c of the optimum.

While all NP-complete decision problems are polynomial-time equivalent, research over the past two decades [51, 75], starting with the papers of Johnson [61] and Sahni and Gonzales [86], suggests that NP-hard optimization problems differ vastly if we are interested in computing approximately optimal solutions. Some NP-hard problems, such as the knapsack problem [85, 59], have a *Fully Polynomial Time Approximation Scheme*: an algorithm that, for any given $\epsilon > 0$, approximates the problem within a factor $1 + \epsilon$ in time that is polynomial in the input size and $1/\epsilon$. The class of problems with such approximation schemes is called *FPTAS*. Some other NP-hard problems, such as the problem of scheduling processes on a multiple processor machine so as to minimize their makespan [57], have a *Polynomial Time Approximation Scheme*: an algorithm that, for any given $\epsilon > 0$, approximates the problem within a factor $1 + \epsilon$ in time that is polynomial in the input size (and could depend arbitrarily upon $1/\epsilon$). The class of problems with such approximation schemes is called *PTAS*. Most problems of interest are not known to be in PTAS. However, many of the latter problems, such as maximum cut, vertex cover, Euclidean Steiner tree, and Euclidean traveling salesman problem, have a constant-factor approximation algorithm, i.e., algorithms which for some fixed constant $c > 1$, are able to approximate the optimal solution to within a factor c in polynomial time. The class of such problems is called *APX*. It follows from the definitions that FPTAS $\subseteq$ PTAS $\subseteq$ APX.

Researchers have also tried to show that problems do not belong to some of the classes above. Garey and Johnson [50] invented the notion of *strong NP-completeness* to show that a large collection of problems are not in FPTAS if P $\neq$ NP. Sahni and Gonzalez [86] showed that the (unrestricted) traveling salesman problem is not in APX if P $\neq$ NP. But the status of many other problems remained open. For example, it was not known if the clique problem is in APX.

In a recent breakthrough, Feige, Goldwasser, Lovasz, Safra, and Szegedy [40] provided strong evidence that clique is not in APX. They showed that if it is possible to approximate the clique number to within any constant factor in polynomial time, then every NP problem can be solved in $n^{O(\log \log n)}$ time. More recently, Arora and Safra [5] improved this result to show that if clique is in APX, then P = NP. In other words, approximating clique within any constant factor is NP-hard. These results relied upon algebraic techniques from complexity theory and the theory of interactive proofs. Arora and Safra also used these techniques to give a new probabilistic characterization of NP which is of independent interest (and is described below).

However, there has been less progress in showing that APX problems are not in PTAS. The one important work in this direction is due to Papadimitriou and Yannakakis [76], who show that a large subset of APX problems are essentially equivalent in this regard: either all of them belong to PTAS, or none of them do so. They used 2nd order logic to define a class of NP optimization problems called MAX SNP that is contained within APX. (The inspiration to use 2nd order logic came from the work of Fagin [38] and Kolaitis and Vardi [68].) They also defined a notion of approximation-preserving reductions and, thereby, the notion of *completeness* and *hardness* for MAX SNP. We will not define these terms here, except to note that a *MAX SNP-complete* problem is in PTAS if and only if MAX SNP $\subseteq$ PTAS, and that a *MAX SNP-hard* problem is in PTAS only if MAX SNP $\subseteq$ PTAS. Many APX problems of real interest are MAX SNP-hard, e.g., MAX-3SAT, MAX-CUT, vertex cover, independent set, and metric traveling salesman problem. Note that to show that none of the MAX SNP-hard problems is in PTAS, it suffices to exhibit just *one* MAX SNP problem that is not in PTAS.

In this paper we show that if P $\neq$ NP, then the MAX SNP problem MAX-3SAT — the problem of computing the maximum number of simultaneously satisfiable clauses in a 3-CNF formula — is not in PTAS. Thus, it follows that if P $\neq$ NP, then all MAX SNP-hard problems are not in PTAS.

Our result, like those of Feige et al and Arora-Safra, involves constructing efficient verifiers that probabilistically check membership proofs for NP languages. As a consequence, we improve upon Arora-Safra's characterization of NP.

In the concluding section of the paper (Section 8) we discuss related work that has appeared since the circulation of the first draft of this paper.

1.1 Related Recent Work

As hinted above, a large body of work in complexity theory and the theory of interactive proofs forms the backdrop for our work. We briefly describe the more relevant developments.

1.1.1 Proof Verification

By definition, NP is the class of languages for which membership proofs can be checked in deterministic polynomial time. In other words, for every NP language L, there exists a polynomial time Turing Machine M, called a *verifier*, that takes pairs of strings as its input and behaves as follows: if a string $x \in L$, then there exists a string π such that M accepts the pair (x, π); conversely, if $x \notin L$, then for all strings π, M rejects (x, π).

Generalizing the above definition of NP leads to definitions of interesting new complexity classes, which have been the subject of intense research in the past decade.

Goldwasser, Micali and Rackoff [56] and Babai [8, 14] allowed the verifier to be a probabilistic polynomial-time Turing Machine that interacts with a "prover," which is an infinitely powerful Turing Machine trying to convince the verifier that the input x is in the language. A surprising recent result, due to Lund, Fortnow, Karloff and Nisan [72] and Shamir [88], has shown that every language in PSPACE — which is suspected to be a much larger class than NP — admits such "interactive" membership proofs. Another variant of proof verification, due to Ben-Or, Goldwasser, Kilian and Wigderson [22], involves a probabilistic polynomial-time verifier interacting with more than one mutually non-interacting provers. The class of languages with such interactive proofs is called MIP (for Multi-prover Interactive Proofs). Fortnow, Rompel and Sipser [44] gave an equivalent definition of MIP in terms of a probabilistic polynomial-time verifier that checks membership proofs (possibly of exponential length) for the language. This equivalent definition is described below, using the term *probabilistically checkable proofs* introduced by Arora and Safra.

Babai, Fortnow and Lund [11] recently showed that MIP is exactly NEXPTIME, the class of languages for which membership proofs can be checked deterministically in exponential time. This result is surprising because NEXPTIME is just the exponential analogue of NP, and its usual definition involves no notion of randomness or interaction. Therefore researchers tried to discover if the MIP = NEXPTIME result can be "scaled-down" to say something interesting about NP. Babai, Fortnow, Levin and Szegedy [12] introduced the notion of *transparent* membership proofs, namely, membership proofs that can be checked in polylogarithmic time. They showed that NP languages have such proofs. Feige et al [40] showed a similar result, but with a somewhat more efficient verifier. Arora and Safra [5] further improved the efficiency of checking membership proofs for NP languages. They also gave a surprising new characterization of NP. We describe their result below, and describe how we have improved upon it.

Arora and Safra define a hierarchy of complexity classes called PCP (for Probabilistically Checkable Proofs). This definition uses the notion of a "probabilistic oracle verifier" of Fortnow et al [44] and classifies languages based on how efficiently such a verifier can check membership proofs for them. The notion of "efficiency" refers to the number of random bits used by the verifier as well as the number of bits it reads in the membership proof. These parameters were first highlighted by the work of Feige et al.[40]. In fact, the definition of a class very similar to PCP was implicit in the work of Feige et al [40].

Definition 1 ([5]) *For functions $r, q : \mathcal{Z}^+ \to \mathcal{Z}^+$, a probabilistic polynomial-time verifier V is* $(r(n), q(n))$-restricted *if, for every input of size n, it uses at most $r(n)$ random bits and examines at most $q(n)$ bits in the membership proof while checking it.*

A language L is in PCP$(r(n), q(n))$ if there exists a $(r(n), q(n))$-restricted polynomial-time verifier that, for every input x, behaves as follows:

- *if $x \in L$, then there exists a membership proof π such that V accepts (x, π) with probability 1 (i.e., for every choice of its random bits);*

- *if $x \notin L$, then for any membership proof π, V accepts π with probability at most* $1/2$.

The PCP notation allows a compact description of many known results,

- NEXPTIME $= \cup_{c>0}$PCP(n^c, n^c) (Babai, Fortnow, and Lund [11]).
- NP $\subseteq \cup_{c>2}$PCP$(\log^c n, \log^c n,)$ (Babai, Fortnow, Levin, and Szegedy [12]).
- NP $\subseteq \cup_{c>0}$PCP$(c \log n \,\mathrm{loglog}\, n, c \log n \,\mathrm{loglog}\, n)$ (Feige, Goldwasser, Lund, Safra, and Szegedy [40]).
- NP $= \cup_{c>0}$PCP$(c \log n, c\sqrt{\log n})$ (Arora and Safra [5]).

Notice that the first and the last of the above relations are exact characterizations of the complexity classes NEXPTIME and NP, respectively. In this paper we improve upon the last result (see Theorem 4).

1.1.2 PCP and Non-approximability

A result due to Feige et al [40] implies that in order to prove the hardness of approximating the clique number, it suffices to show that every NP language is low in the PCP hierarchy. The following statement summarizes this result.

Theorem 2 ([40]) *Suppose that for some function $q : \mathcal{Z}^+ \rightarrow \mathcal{Z}^+$,*

$$NP \subset \cup_{c>0} PCP(c \log n, q(n)).$$

Then, there exists a constant k such that approximating the clique number in an N-vertex graph to within a factor of $N^{1/(k+q(N))}$ is NP-hard. ▮

Thus, as a consequence of the new characterization of NP due to Arora et al [5], it follows that approximating the clique number within a factor $2^{\theta(\sqrt{\log N})}$ is NP-hard.

The discovery of Theorem 2 inspired the search for other connections between probabilistic proof checking and non-approximability (Bellare [16], Bellare and Rogaway [20], Feige and Lovasz [43], and Zuckerman [93]). Another such connection is reported by Arora, Motwani, Safra, Sudan and Szegedy [4], which shows the connection between PCP's and the hardness of approximating MAX 3SAT. The following theorem summarizes this result; for a proof see Section 3.

Theorem 3 ([4]) *If $NP \subset \cup_{c>0} PCP(c \log n, q)$ for some positive integer q, then there exists a constant $\epsilon > 0$, such that approximating MAX-3SAT within a factor $1 + \epsilon$ is NP-hard.*

1.2 Our results

The main result of this paper is a new characterization of NP in terms of PCP, which improves the earlier characterization due to Arora et al [5].

Theorem 4 (Main) *There is a positive integer q such that $NP = \cup_{c>0} PCP(c \log n, q)$.*

The containment $\cup_{c>0}\mathrm{PCP}(c\log n, q) \subseteq \mathrm{NP}$ is trivial, since a verifier that uses $O(\log n)$ random bits can be replaced by a deterministic verifier. The nontrivial containment $\mathrm{NP} \subseteq \cup_{c>0}\mathrm{PCP}(c\log n, q)$ is a consequence of our Theorem 16 below.

We note that the characterization in Theorem 4 is probably optimal upto constant factors, since, as observed in [5], if $\mathrm{P} \neq \mathrm{NP}$, then $\mathrm{NP} \not\subseteq \mathrm{PCP}(o(\log n), o(\log n))$.

As an immediate consequence of our main theorem and Theorem 3, we have the following.

Theorem 5 *There exists a constant $\epsilon > 0$ such that approximating the maximum number of simultaneously satisfiable clauses in a 3CNF formula within a factor $1+\epsilon$ is NP-hard.*

As noted earlier, Theorem 5 implies that if $\mathrm{P} \neq \mathrm{NP}$ then no MAX SNP-hard problem is in PTAS. The class of MAX SNP-hard problems includes MAX 2SAT, MAX CUT, vertex cover [76], metric TSP [77], Steiner tree [25], shortest superstrings [26], MAX 3DM [62], and multiway cut [36]. Our theorem above implies that for every one of these problems Π, there exists a threshold ϵ_Π such that approximating Π within a factor $1+\epsilon_\Pi$ is NP-hard.

Also, as a consequence of our main theorem and Theorem 2, we can prove the following result about the non-approximability of clique number.

Theorem 6 *There exists a constant $\epsilon > 0$, such that approximating the maximum clique size in an N-vertex graph to within a factor of N^ϵ is NP-hard.*

The previous best result (in [5]) shows that approximating clique to within a factor of $2^{\sqrt{\log N}}$ is NP-hard.

Our techniques. Our proof of Theorem 4 uses techniques similar to those in recent results about PCPs (Babai, Fortnow, Levin, and Szegedy [12]; Feige, Goldwasser, Lovasz, Safra and Szegedy [40]; and Arora-Safra [5]). The concept of *Recursive Proof Checking* invented by Arora and Safra plays a particularly important role (see Section 2). We were also influenced by work on program checking and correcting, especially Blum, Luby, and Rubinfeld [29], and Rubinfeld and Sudan [84]. The influence of the former will be apparent in Section 5, while the latter work (together with a lemma of Arora and Safra [5]) is used in our analysis of the Low Degree Test described in Section 7.2. Finally, we were influenced by work on *constant prover 1-round interactive proof systems* [69, 43]. In fact, our definition of an outer verifier (Definition 8) may be viewed as a generalization of the definition of such proof systems, and our Theorem 4 provides the first known construction of a 2-prover 1-round proof system that uses logarithmic random bits and constant number of communication bits. In fact, prior to this paper, no construction of constant prover one round proof system using logarithmic random bits and $n^{o(1)}$ communication bits was known. Theorem 14, which presents such a proof system with logarithmic randomness and polylogarithmic communication bits, already improves upon the performance of known constructions and plays a central role in the proof of Theorem 4.

2 Proof of the Main Theorem: Overview

A key paradigm in the proof of Theorem 4 is the technique of *recursive proof checking* from [5]. The technique, described in Theorem 13, involves constructing two types of verifiers with certain special properties, and then *composing* them. If the first verifier queries membership proofs in $q_1(n)$ bits while checking it and the second in $q_2(n)$ bits, where n is the input size, then the composed verifier queries the proof in approximately $q_2(q_1(n))$ bits. Typically, q_1 and q_2 are sublinear functions of n (actually, they are closer to $\log n$), so the function $q_1 \circ q_2$ grows slower than either q_1 or q_2.

As already mentioned, the above composition is possible only when the two verifiers have certain special properties, which we describe below. The main contribution of this paper are some new techniques for constructing such verifiers, which will be described when we prove Theorems 14 and Theorems 15.

First we introduce some terminology. We define CKT-SAT, an NP-complete language. An *algebraic circuit* (or just *circuit* for short) is a directed acyclic graph with fan-in 2. Some of the nodes in the graph are *input* nodes; the others are *internal* nodes, each marked with one of the two symbols $+$ and $\cdot$ that are interpreted as operations over the field $GF(2)$. One special internal node is designated as the *output* node. We may interpret the circuit as a device that for every assignment of 0/1 values to the input nodes, produces a unique 0/1 value for each internal node, by evaluating each node in the obvious way according to the operation ($+$ or $\cdot$) labeling it. For a circuit C with n input nodes and an assignment of values $x \in \{0,1\}^n$ to the input nodes, the *value of C on x*, denoted $C(x)$, is the value produced this way at the output node. If $C(x) = 1$, we say x is a *satisfying assignment* to the circuit. The *size* of a circuit is the number of wires in it. The CKT-SAT problem is the following.

CKT-SAT
Given: *An algebraic circuit C.*
Question: *Does C have a satisfying assignment?*

Clearly, CKT-SAT $\in$ NP; furthermore, a 3-CNF formula can be viewed as a circuit, so 3-SAT is a special case of CKT-SAT. Hence it follows that CKT-SAT is NP-complete.

Oracles. All verifiers in this paper expect membership proofs to have a very regular structure — the proof consists of *oracles*. An oracle is a table whose each entry is a string from $\{0,1\}^a$, where a is some positive integer called the *answer size* of the oracle. The verifier can read the contents of any location in the table by writing its *address* on some special tape. In our constructions, this address is interpreted as an element of some algebraically defined domain, say D. (That is to say, the oracle is viewed as a function from D to $\{0,1\}^a$.) The operation of reading the contents of a location $q \in D$ is called as *querying the location* q.

2.1 Outer and Inner Verifiers and How They Compose

Now we define two special types of verifiers, the *outer* and *inner* verifiers. An outer verifier can be composed with an inner verifier and, under some special conditions, the resulting verifier is more efficient than both of them.

The notable feature of both types of verifiers is that their decision process has a succinct representation as a small circuit. (In our constructions, the size of the circuit is polylogarithmic in the input size, or less.) By this we mean that the verifier, after reading the input and the random string, computes a (small) circuit and a sequence of locations in the proof. Then it queries those locations in the proof, and accepts if and only if the concatenation of the entries in those locations is a satisfying assignment for the circuit's inputs.

Definition 7 *For functions $r, p, c, a\colon \mathcal{Z}^+ \to \mathcal{Z}^+$, an $(r(n), p(n), c(n), a(n))$* **outer verifier** *is a randomized Turing machine V which expects the membership proof for an input of size n to be an oracle of answer size $a(n)$. Given an input $x \in \{0,1\}^n$ and an oracle π of answer size $a(n)$, V runs in $\mathrm{poly}(n)$ time and behaves as follows:*

1. **Uses $r(n)$ random bits:** *V reads x and picks a string R uniformly at random from $\{0,1\}^{r(n)}$.*
2. **Constructs a circuit of size $c(n)$:** *V computes a circuit C of size $c(n)$.*
3. **Computes $p(n)$ locations in π:** *V uses x and R to compute $p(n)$ locations in π. Let $q_1, \ldots, q_{p(n)}$ denote these locations.*
4. **Queries the oracle:** *V queries the oracle π in locations $q_1, \ldots, q_{p(n)}$. For $i = 1, \ldots, p(n)$, let a_i denote the string in the location q_i of π.*
5. **Makes a decision:** *V outputs* accept *if $a_1 \circ a_2 \circ \cdots \circ a_{p(n)}$ is a satisfying assignment to circuit C (where $\circ$ denotes concatenation of strings), and otherwise it outputs* reject*. We denote this decision by $V^\pi(x; R)$.*

Remark: Note that the number of entries in the oracle has been left unspecified. But without loss of generality, it can be upper bounded by $2^{r(n)}p(n)$, since this is the maximum number of locations the verifier can query in its $2^{r(n)}$ possible runs.

Definition 8 *A language L is in* $\mathbf{RPCP}(r(n),\, p(n),\, c(n),\, a(n),\, e)$*, where* $r, p, c, a\colon \mathcal{Z}^+ \to \mathcal{Z}^+$ *and e is a positive fraction less than 1, if there exists a* $(r(n), p(n), c(n), a(n))$ *outer verifier which satisfies the following properties for every input x.*

Completeness *If $x \in L$, then there exists an oracle π such that*

$$\Pr_R[V^\pi(x; R) = \mathsf{accept}] = 1$$

.

Soundness *If $x \notin L$, then for all oracles π,*

$$\Pr_R[V^\pi(x; R) = \mathsf{accept}] \leq e.$$

In both cases, the probability is over the choice of R in $\{0,1\}^{r(|x|)}$.

We note that an $(r(n), p(n), c(n), a(n))$ outer verifier with $p(n) = O(1)$ is very similar to a *constant prover* 1 *round* interactive proof system [44]. Fairly efficient constructions of such verifiers are implicit in [69, 43]; for instance, it is shown that $\mathrm{NP} \subseteq \mathrm{RPCP}((\log)^{O(1)}n,\, O(1),\, (\log)^{O(1)}n,\, (\log)^{O(1)}n,\, 1/n)$. We also observe that the definition of RPCP generalizes that of PCP. In particular, $\mathrm{RPCP}(r(n),\, p(n),\, c(n),\, a(n),\, 1/2) \subseteq \mathrm{PCP}(r(n), p(n)a(n))$, since an $(r(n), p(n), c(n), a(n))$ outer verifier examines $p(n)a(n)$ bits in the oracle. So to prove Theorem 4, it suffices to show $\mathrm{NP} \subseteq \mathrm{RPCP}(r(n),\, p(n),\, c(n),\, a(n),\, 1/2)$, where $p(n)$ and $a(n)$ are some fixed constants and $r = O(\log n)$. Not knowing any simple way to achieve this, we give a 3-step construction (see the proof of Theorem 16). At each step, r remains $O(\log n)$, p remains $O(1)$, and e remains some fixed fraction. The only parameters that change are $c(n)$ and $a(n)$, which go down from $\mathrm{poly}(\log n)$ to $\mathrm{poly}(\log\log n)$ to $O(1)$.

First we define an inner verifier, a notion implicit in Arora and Safra [5]. To motivate this definition, we state informally how an inner verifier will be used during recursive proof checking. We will use it to perform Step 4 of an outer verifier — namely, checking that $a_1 \circ \cdots \circ a_{p(n)}$, the concatenation of the oracle's replies, is a satisfying assignment for the circuit C — without reading most of the bits in $a_1, \ldots, a_{p(n)}$. This may sound impossible — how can you check that a bit string is a satisfying assignment without reading every bit in it? But Arora and Safra showed how to do it by modifying a result of Babai et al [12] who had shown that if a bit-string is given to us in an encoded form (using a specific error-correcting code), then it is possible to check a proof that the string is a satisfying assignment, *without reading the entire string!* Arora and Safra show how to do the same check when the input, in addition to being encoded, is also split into many parts. This is explained further in the definition of an inner verifier.

First we define an encoding scheme, which is a map from strings over one alphabet to strings over another alphabet. We will think of an encoding scheme as a mapping from a string to an oracle.

Definition 9 *For $l, a \in \mathcal{Z}^+$, let $\mathcal{F}_{l,a}$ denote the family of functions* $\{f | f : [l] \to \{0,1\}^a\}$*. Equivalently, one can think of $\mathcal{F}_{l,a}$ as the family of l-letter strings over the alphabet* $\{0,1\}^a$.

Definition 10 (Valid Encoding/Decoding Scheme) *For functions $l, a : \mathcal{Z}^+ \to \mathcal{Z}^+$, an $(l(n), a(n))$-*encoding scheme *is an ensemble of functions $E = \{E_n\}_{n \in \mathcal{Z}^+}$, such that $E_n : \{0,1\}^n \to \mathcal{F}_{l(n),a(n)}$. An (l, a)-*decoding scheme *is an ensemble of functions $E^{-1} = \{E_n^{-1}\}_{n \in \mathcal{Z}^+}$, such that $E_n^{-1} : \mathcal{F}_{l(n),a(n)} \to \{0,1\}^n$. A pair of encoding/decoding schemes (E, E^{-1}) is* valid *if for all $x \in \{0,1\}^*$, $E_{|x|}^{-1}(E_{|x|}(x)) = x$.*

In other words, an encoding scheme maps an n bits string to an $l(n)$-letter string over the alphabet $2^{a(n)}$. A decoding scheme tries to reverse this mapping and is valid if and only if $E^{-1} \circ E$ is the identity function. Notice that the map E^{-1} could behave arbitrarily on $l(n)$-letter strings which do not have pre-images under E.

Now we define inner verifiers. Such verifiers check membership proofs for the language CKT-SAT, and expect the proof to have a very special structure.

Definition 11 *For functions $r, p, c, a : \mathcal{Z}^+ \to \mathcal{Z}^+$, positive integer k and fraction $e \in \mathcal{R}^+$, a $(k, r(n), p(n), c(n), a(n), e)$-***inner verifier system** *is a triple (V, E, E^{-1}), where V is a probabilistic Turing machine and (E, E^{-1}) is a valid $(l(n), a(n))$-encoding/decoding scheme, for some function $l : \mathcal{Z}^+ \to \mathcal{Z}^+$.*

The input to the verifier is a circuit. The number of input nodes of the circuit is a multiple of k. Let C be this circuit, let n be its size and km be the number of inputs to C. Then, V expects the membership proof to consist of $k+1$ oracles $X_1, \ldots, X_{k+1}$, each of answer size $a(n)$. For clarity, we use the shorthand π for X_{k+1}. The verifier runs in poly(n) *time and behaves as follows.*

1. **Uses $r(n)$ random bits:** *V reads C and picks a random string $R \in \{0,1\}^{r(n)}$.*
2. **Constructs a circuit of size $c(n)$:** *Based on C and R, V computes a circuit C' of size $c(n)$.*
3. **Computes $p(n)$ locations in π:** *V computes $p(n)$ locations $q_1, \ldots, q_{p(n)}$ in the oracles. Each location q_i is a pair (q_i^1, q_i^2) where $q_i^1 \in [k+1]$ denotes the oracle in which it is contained and q_i^2 denotes the position within that oracle.*
4. **Queries the oracles:** *V uses these $p(n)$ locations computed above to query the oracles. Let $a_1, \ldots, a_{p(n)}$ denote the strings received as answers, where each $a_i \in \{0,1\}^{a(n)}$.*
5. **Makes a decision:** *V outputs* accept *if $a_1 \circ \cdots \circ a_{p(n)}$ is a satisfying assignment for C'. Otherwise it outputs* reject*. We denote this decision by $V^{X_1,\ldots,X_k,\pi}(C; R)$.*

   Then the following properties hold.

   Completeness: *Let $x_1, \ldots, x_k$ be m-bit strings such that $x_1 \circ x_2 \cdots \circ x_k$ is a satisfying assignment for C. Then there exists an oracle π such that*

   $$\Pr_R[V^{X_1,\ldots,X_k,\pi}(C; R)] = \mathsf{accept}] = 1,$$

   where X_j is the encoding $E_m(x_j)$ for $1 \leq j \leq k$.

Soundness: *If for some oracles $X_1, \ldots, X_k, \pi$, the probability*

$$\Pr_R \left[V^{X_1,\ldots,X_k,\pi}(C; R) = \mathsf{accept} \right] \geq e,$$

then $E_m^{-1}(X_1) \circ \cdots \circ E_m^{-1}(X_k)$ is a satisfying assignment for C.

The previous definition is complicated. The following observation might clarify it a little, and the proof of Theorem 13 will clarify it further.

Proposition 12 *If $r, p, c, a\colon \mathcal{Z}^+ \to \mathcal{Z}^+$ are any functions, $k \in \mathcal{Z}^+$, $e < 1$ and there exists a $(k, r(n), p(n), c(n), a(n), e)$-inner verifier system, then CKT-SAT $\in$ RPCP$(r(n)$, $p(n)$, $c(n)$, $a(n)$, $e)$.*

Proof: The verifier in a $(k, r(n), p(n), c(n), a(n), e)$ inner verifier system uses $r(n)$ random bits, expects the proof to be an oracle of answer size $a(n)$ (i.e., we ignore the special structure of the proof as described in the definition, and think of the $k+1$ oracles in it as one long oracle), and queries $p(n)$ locations in the oracle. Its decision is represented by a circuit of size $c(n)$. Thus it is also an $(r(n), p(n), c(n), a(n))$ outer verifier.

Further, by definition, it can checks membership proofs for CKT-SAT: if the input circuit is satisfiable, then the completeness condition implies that there is an oracle which the verifier accepts with probability 1. Conversely, if the verifier accepts some oracle with probability e, then soundness implies that the circuit is satisfiable.

Hence CKT-SAT $\in$ RPCP$(r(n)$, $p(n)$, $c(n)$, $a(n)$, $e)$. ∎

The technique of recursive proof checking is described in the proof of the following theorem.

Theorem 13 ([5]) *Let $p \in \mathcal{Z}^+$ be such that there exists a $(p, r_1(n), p_1(n), c_1(n), a_1(n), e_1)$-inner verifier system for some functions $r_1, p_1, c_1, a_1\colon \mathcal{Z}^+ \to \mathcal{Z}^+$ and $0 < e_1 < 1$. Then, for all functions $r, c, a\colon \mathcal{Z}^+ \to \mathcal{Z}^+$ and a positive fraction e,*

RPCP$(r(n)$, p, $c(n)$, $a(n)$, $e) \subseteq$ RPCP$(r(n)+r_1(\tau)$, $p_1(\tau)$, $a_1(\tau)$, $c_1(\tau)$, $e+e_1-ee_1)$,

where τ is a shorthand for $c(n)$ and $r(n) + r_1(\tau)$ is a shorthand for $r(n) + r_1(c(n))$.

We will prove Theorem 13 at the end of this section. First, we show how to prove Theorem 4. The main ingredients are Theorems 14 and 15, whose proofs (in Sections 6 and 7.5 respectively) are the main technical contributions of this paper.

Theorem 14 *For every constant $k \in \mathcal{Z}^+$, there exist constants $c_1, c_2, c_3, p \in \mathcal{Z}^+$ and real number $e < 1$, such that for some functions $c(n) = O(\log^{c_2} n)$ and $a(n) = O(\log^{c_3} n)$ there exists a $(k, c_1 \log n, p, c(n), a(n), e)$ inner verifier system.*

Theorem 15 *For every constant $k \in \mathcal{Z}^+$, there exist constants $c_1, p \in \mathcal{Z}^+$ and a positive real number $e < 1$, such that there exists a $(k, c_1 n^2, p, 2^p, 1, e)$ inner verifier system.*

Now we prove Theorem 4, our main theorem. It is a simple consequence of the following theorem.

Theorem 16 *There exists a constant C, such that for every language $L \in NP$, there exists a constant c_L so that*

$$L \in \mathrm{RPCP}(c_L \log n,\, C,\, 2^C,\, 1,\, 1/2).$$

Proof: Since CKT-SAT is NP-complete, we are done if we can show that CKT-SAT $\in \mathrm{RPCP}(c_L \log n,\, C,\, 2^C,\, 1,\, 1/2)$ for some constants c_L, C.

The main idea is to use the verifier of Theorem 15 and use recursive proof checking to construct more efficient verifiers. First we use Theorem 15 with $k = 1$. The theorem guarantees us a $(1, c_1 \log n, p, c(n), a(n), 1/16)$-inner-verifier system where $a(n) = \log^{d_1} n$ and $c(n) = \log^{d_2} n$ for some constants d_1, d_2, and c_1, p are also constants. Then Proposition 12 implies that

$$\text{CKT-SAT } \in \mathrm{RPCP}(c_1 \log n,\, p,\, \log^{d_2} n,\, \log^{d_1} n,\, 1/16). \tag{1}$$

This verifier for CKT-SAT makes p queries to the oracle in the membership proof. Use this same constant p as the value of k in Theorem 15. We obtain constants c', c'', d' such that a $(p, c' \log n, c'', \log^{d'} n, \log^{d'} n, 1/16)$-inner verifier system exists.

The existence of this inner verifier allows us to apply Theorem 13 to (1) to obtain that

$$\text{CKT-SAT } \in \mathrm{RPCP}(c_1 \log n + c' d_1 \log\log n,\, c'',\, (d_1 \log\log n)^{d'},\, (d_1 \log\log n)^{d'},\, 1/8). \tag{2}$$

This verifier for CKT-SAT makes c'' queries to the oracle in the membership proof. Using the constant c'' as p in the statement of Theorem 15, we obtain constants g, h such that there is a $(c'', gn^2, h, 2^h, 1, 1/16)$ inner-verifier system.

The existence of this verifier allows us to apply Theorem 13 to the statement (2), to obtain

$$\text{CKT-SAT } \in \mathrm{RPCP}(c_1 \log n + c' d \log\log n + g(d \log\log n)^{2d'},\, h,\, 2^h,\, 1,\, 3/16). \tag{3}$$

Since every fixed power of $\log\log n$ is $o(\log n)$ and h, c_1 are fixed constants, our theorem has been proved. ▮

Now we prove Theorem 13.

Proof: (*of Theorem 13*) First we outline the main idea. Let L be a language in RPCP($r(n)$, p, $c(n)$, $a(n)$, e) and let V_1 be the outer verifier that checks membership proofs for it. Let (V_2, E, E^{-1}) be the $(p, r_1(n), p_1(n), c_1(n), a_1(n), e_1)$ inner verifier system mentioned in the hypothesis. Observe that once we fix the input and the random string for V_1, its decision to accept or reject is based upon the contents of only p locations in the oracle. Moreover, these p locations needs to satisfy a very simple condition: the concatenation of their respective entries should be a satisfying assignment to a certain (small) circuit. The main idea now is that verifier V_1 can use the inner verifier V_2 to check that this condition holds. The new verifier thus obtained turns out to be a new outer verifier, which we denote by V.

Now we fill in the above outline. Let $x \in \{0,1\}^n$ be any input. According to the hypothesis, the outer verifier V_1 expects a membership proof for x to be an oracle of answer size $a(n)$. Also, V_1 uses $r(n)$ bits. Let

$$Y \quad = \quad \text{Number of locations that } V_1 \text{ expects in an oracle for input } x. \tag{4}$$

Then the new verifier V that we are going to describe expects a membership proof for x to be an oracle containing $Y + 2^{r(n)}$ sub-oracles. Let this membership proof be denoted by π. Then each address in π is denoted by a pair $[s,t]$, where $s \leq Y + 2^{r(n)}$ is index of the sub-oracle, and t is the position within the sub-oracle. We let $\pi[s,\cdot]$ denote the suboracle of π whose index is s. (Note that we have not specified the number of locations within each suboracle. This is determined using the program of V_2, as will be clear in a minute.)

The new verifier V acts as follows. First it picks a random string $R_1 \in \{0,1\}^{r(n)}$. Then it simulates the outer verifier V_1 for Steps 1 through 3 described in Definition 7, while using R_1 as the random string. Note that these steps do not require any querying of the oracle. Let

$$\begin{aligned} C \quad &= \quad \text{the circuit computed by Step 2 of } V_1. && (5)\\ Q_1, \ldots, Q_p \quad &= \quad \text{the queries generated by Step 3 of } V_1. && (6) \end{aligned}$$

Note that C has size $c(n)$, and that each Q_i is an integer from 1 to Y.

Next, V picks a random string $R_2 \in \{0,1\}^{r_1(c(n))}$ and simulates the inner verifier V_2 on the input C and random string R_2. Note that V_2 expects a membership proof to contain $p+1$ oracles. The simulation uses the suboracles $\pi[Q_1,\cdot], \ldots, \pi[Q_p,\cdot], \pi[Y + R,\cdot]$ of π as these $p+1$ oracles. (Note that we are thinking of R as an integer between 1 and $2^{r(n)}$.) If this simulation of V_2 ends up producing accept as output, then V outputs accept and otherwise reject.

This finishes the description of V.

Complexity: It is clear from the above description that V uses $r(n)+r_1(c(n))$ random bits. All its other parameters are just those of V_2 when given an input of size $c(n)$. Hence we conclude that V queries the oracle in $p_1(c(n))$ locations, expects the oracle to have answer size $a_1(c(n)))$, and bases its decision upon whether the oracle entries constitute a satisfying assignment to a circuit of size $c_1(c(n))$. In other words, it is an $(r(n) + r_1(n),\ p_1(c(n)),\ c_1(c(n),\ a_1(c(n)))$ outer verifier.

Completeness and soundness: We have to show that V satisfies the completeness and soundness conditions for language L. For each $x \in \{0,1\}^n$, we prove these conditions separately.

Case: $x \in L$: The completeness condition for V_1 implies that there is an oracle π' which V_1 accepts with probability 1. We describe how to convert π' into an oracle π which V accepts with probability 1. First, replace the string in each location of π' with the encoding of that string using $E_{c(n)}$. (This encoding is an oracle of answer size $a_1(c(n))$.) Thus if π' had Y locations, we obtain a sequence of Y oracles. We make these oracles the first Y sub-oracles of π. Next, we construct the last $2^{r(n)}$ suboracles of π. For $R \in \{0,1\}^{r(n)}$, let C be the circuit generated by V_1 using R as the random string. Let $a_1, \ldots, a_p$ be the responses given by the oracle π' when the queries were generated by V_1 using random string R. The completeness condition for V_1 implies that $a_1 \ldots a_p$ is a satisfying assignment for C. Then the completeness condition for V_2 implies that there exists an oracle τ such that

$$\Pr_{R_2 \in \{0,1\}^{r_1(c(n))}}[V_2^{X_1,\ldots,X_p,\tau}(C; R_2) = \mathsf{accept}] \quad = \quad 1,$$

where each X_i is simply the encoding $E_{|a_i|}(a_i)$. We let this τ be the $Y + R$'th sub-oracle of π. This finishes the description of π. Our construction has ensured that

$$V^{\pi}(x; (R, R_2)) = \mathsf{accept} \quad \forall R \in \{0,1\}^{r(n)}, R_2 \in \{0,1\}^{r_1(c(n))}.$$

In other words, V accepts π with probability 1.

Case: $x \notin L$: We prove this part by contradiction. Assume that there is an oracle π such that

$$\Pr_{R \in \{0,1\}^{r(n)}, R_2 \in \{0,1\}^{r_1(n)}}[V^{\pi}(x; (R, R_2)) = \mathsf{accept}] > e + e_1 - ee_1.$$

We use π to construct an oracle π' such that

$$\Pr_{R \in \{0,1\}^{r(n)}}\left[V_1^{\pi'}(x; R) = \mathsf{accept}\right] > e,$$

thus contradicting the soundness condition for V_1.

To construct π', we take the first Y suboracles in π, and apply $E^{-1}_{a(n)}$ on each. (Of course, if π does not have some of these suboracles, we use an arbitrary string for the missing suboracles.) The last $2^{r(n)}$ suboracles are discarded. Thus we obtain an oracle π' with Y locations and answer size $a(n)$. Now we show that V_1 accepts π' with probability greater than e. Consider the set

$$\mathcal{R} = \{R \in \{0,1\}^{r(n)} | \Pr_{R_2 \in \{0,1\}^{r_1(n)}}[V^{\pi}(x; R, R_2) = \mathsf{accept}] > e_1\}.$$

Observe that $\mathcal{R}$ constitutes a fraction more than e of $\{0,1\}^{r(n)}$, since otherwise the probability that V accepts π would have been less than $e_1(1-e) + e =$

$e + e_1 - ee_1$. Hence it suffices for us to show that every $R \in \mathcal{R}$ satisfies $V_1^{\pi'}(x; R) = \mathsf{accept}$.

But this follows just by definition of V. Let C denote the circuit generated by V_1 on input x using $R \in \mathcal{R}$ as random string. Let $Q_1, \ldots, Q_p$ be the queries generated by V_1. Then the actions of V on input x and random string (R, R_2) just involve simulating the inner verifier V_2 on input C, random string R_2, and using the suboracles $\pi[Q_1, \cdot], \ldots, \pi[Q_p, \cdot], \pi[Y + R, \cdot]$. Since we know that

$$\Pr_{R_2 \in \{0,1\}^{r_1(n)}} [V^{\pi}(x; R, R_2) = \mathsf{accept}] > e_1,$$

we conclude (by the soundness condition for V_2) that the decoded string

$$E^{-1}_{a(n)}(\pi[Q_1, \cdot]) \circ \cdots \circ E^{-1}_{a(n)}(\pi[Q_1, \cdot])$$

is a satisfying assignment for C. But the corresponding locations in π' contain exactly these decodings. Therefore, $V_1^{\pi}(x; R) = \mathsf{accept}$. This completes the proof.

Thus we have proved that verifier V satisfies all properties claimed for it, and thus $L \in \mathrm{RPCP}(r(n) + r_1(c(n)),\, p_1(c(n)),\, a_1(c(n)),\, a_1(c(n)),\, e + e_2 - ee_2)$.

3 PCP and hardness of approximating MAX 3-SAT

In this section we define MAX 3-SAT and show how the existence of an $(O(\log n), O(1))$-restricted verifier for NP implies the hardness of approximating MAX 3-SAT.

Definition 17 (MAX 3-SAT) *An instance of this problem is a 3-CNF formula ϕ. Every assignment to the variables of the formula is a solution, whose value is the number of clauses of formula that are satisfied by the assignment. The goal is to find an assignment with maximum value.*

We are now ready to prove Theorem 3.

Proof of Theorem 3: Let L be a language in NP and V be a $(c_L \log n, q)$-restricted verifier for it, where c_L, q are constants. We give a polynomial-time reduction that, given input x of length n, produces a 3-SAT formula ϕ_x with at most $2^q 2^{c_L \log n}$ clauses. This formula satisfies the following conditions.

1. If $x \in L$, then ϕ_x is satisfiable.
2. If $x \notin L$, then every assignment fails to satisfy at least $2^{c_L \log n - 1}$ clauses in ϕ_x.

Without loss of generality, we assume that the answer size of the oracle is 1 (i.e., each oracle entry is a bit). The reduction goes as follows. For every potential query Q_i to the oracle π made by the verifier V, associate a boolean variable v_i. (Thus the set of possible assignments to the variables are in one-to-one correspondence with the set of possible oracles.) For every choice of the verifier's random string, do the following. Suppose the random string is $R \in \{0,1\}^{c_L \log n}$ and suppose the verifier V asks queries $Q_{i_1}, \ldots, Q_{i_q}$ using R. Let $\psi_{x,R}: \{0,1\}^q \to \{0,1\}$ be a boolean function such that $\psi_{x,R}(a_{i_1}, \ldots, a_{i_q})$ is 1 if and only if V accepts upon receiving $a_{i_1}, \ldots, a_{i_q}$ as the responses from the oracle. Express $\psi_{x,R}$ as a boolean formula in the variables $v_{i_1}, \ldots, v_{i_q}$. Then express it in the standard way (see e.g. [51]) as a 3-CNF formula, by using some auxiliary variables. (These auxiliary variables should be unique to R and should not be reused for any other random string.) Let $\phi_{x,R}$ denote this 3-CNF formula. Then

$$\phi_x = \bigwedge_R \phi_{x,R}.$$

Denote the number of clauses of ϕ_x by m. Note that m is at most $2^q 2^{c_L \log n}$. Clearly, if $x \in L$, then we can find a satisfying assignment to ϕ_x: take the bits in the oracle as an assignment to the v_i's, and assign to the auxiliary variables in the obvious way. Conversely, suppose $x \notin L$. Then every oracle is rejected by V with probability at least 1/2. In terms of the formula, this means that every assignment to the v_i's fails to satisfy 1/2 of the $\phi_{x,R}$'s (regardless of the values of the auxiliary variables). Each such $\phi_{x,R}$ provides an unsatisfied clause. Thus the number of unsatisfied clauses in ϕ_x is at least $2^{c_L \log n - 1}$. This concludes the description of the reduction.

We conclude by observing that if there exists an algorithm that can distinguish between the cases that the number of satisfiable formulae in ϕ_x is m and the case where the number of satisfiable formulae in ϕ_x is less than $(1-\epsilon)m$ for $\epsilon = 2^{-(q+1)}$, then this distinguishes between the cases $x \in L$ and $x \notin L$. Thus we conclude that no algorithm can approximate the number of satisfiable clauses in a 3-CNF formula to within a $1-\epsilon$ factor in polynomial time, unless P=NP. ▌

4 Terminology

Recall that an encoding/decoding scheme is central to the notion of an inner verifier. Such a scheme maps strings to oracles (in other words, it maps a string to a function from some domain D to some range R). The following definition is used to define the *distance* between the encodings of two strings.

Definition 18 *Let D and R be finite sets and f, g be functions from D to R. The* relative distance between f and g, *denoted $\Delta(f,g)$, is the fraction of inputs from D on which they disagree. If $\Delta(f,g) \leq \epsilon$, we say that f is ϵ-*close to g.

Everywhere in this paper, the domain and range are defined using finite fields. Let $F = GF(q)$ be a field and m be an integer. By specializing the above definition, the distance between two functions $f, g: F^m \to F$, denoted $\Delta(f,g)$, is the fraction of points in F^m they differ on.

$$\Delta(f,g) = \frac{1}{q^m} \left| \{x : \ f(x) \neq g(x)\} \right| .$$

A function $f: F^m \to F$ is a *degree k polynomial* if it can be described as follows. There is a set of coefficients $\left\{ a_{i_1,\ldots,i_m} \in F : i_1, \ldots, i_m \geq 0 \text{ and} \sum_{j \leq m} i_j \leq k \right\}$ such that for every $(x_1, \ldots, x_m) \in F^m$,

$$f(x_1, \ldots, x_m) = \sum_{i_1,\ldots,i_m} a_{i_1,\ldots,i_m} \prod_{j \leq m} x_j^{i_j} .$$

We let $F_m^{(d)}$ denote the set of m-variate polynomials of degree d. The following lemma is often attributed to Schwartz [87].

Lemma 19 *If $f, g \in F_m^{(d)}$, then $\Delta(f,g) \geq 1 - d/|F|$.*

A function $f: F^m \to F$ is *δ-close to $F_m^{(d)}$* (or just δ-close when d is understood from the context) if there is a polynomial $g \in F_m^{(d)}$ such that $\Delta(f,g) \leq \delta$. From the above lemma, it follows that this nearest polynomial g is unique if $\delta < 1/4$ and $d/|F| \leq 1/2$. This fact will be used many times.

5 The constant bit verifier: Ingredients

In this section, we prove Theorem 15 by constructing the appropriate inner verifier. We will use techniques developed in the context of program checking.

Linear Encoding/Decoding scheme. Through this section we work with vector spaces over GF(2) and all operations (additions and multiplications) are assumed to be over GF(2). We will use an encoding/decoding scheme that encodes bit-strings with linear functions over GF(2). We will describe algebraic procedures using which the verifier can check, after a few queries, that the provided function encodes a satisfying assignment to the input circuit (or at least is ϵ-close to the encoding of a satisfying assignment).

Think of n-bit strings as n-dimensional vectors over GF(2). For an n-bit string x, we let $x^{(i)}$ represent its ith coordinate. We encode this string by a linear function $L_x: \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ that maps y to $\sum_{i=1}^n x^{(i)} y^{(i)}$. Note that every linear function from $\mathrm{GF}(2)^n$ to GF(2) is the encoding of some n bit string.

Definition 20 (parity encoding) *The parity encoding scheme $E_n^{\oplus}$ maps the n-bit string x to L_x.*

The following fact is easy to verify

Proposition 21 *For $x \neq y$, $\Delta(E_n^{\oplus}(x), E_n^{\oplus}(y)) \geq 1/2$.*

We now define the corresponding decoding scheme.

Definition 22 (parity decoding) *Given a function $f : GF(2)^n \rightarrow GF(2)$, the parity decoding scheme $(E_n^{\oplus})^{-1}$ maps f to the string x which minimizes the distance $\Delta(E^{\oplus}(x), f)$. Ties are broken arbitrarily.*

Remark: If there exists a function f and string x such that $\Delta(f, E^{\oplus}(x)) < 1/4$, then $(E_n^{\oplus})^{-1}(f) = x$.

Testing/Correcting Given an oracle $f : \mathrm{GF}(2)^n \rightarrow \mathrm{GF}(2)$, we would like to determine very efficiently, probabilistically, if there exists a string $x \in \mathrm{GF}(2)^n$, such that $\Delta(L_x, f)$ is small. The following procedure due to Blum, Luby and Rubinfeld [29] achieves this with just 3 queries to the oracle f.

```
Linearity-test(f; n):
/* Expects an oracle f : GF(2)^n → GF(2).  */

   Pick x, y ∈_R GF(2)^n;
   If f(x) + f(y) ≠ f(x + y) then reject else accept.
```

The following theorem describes the effectiveness of this test.

Theorem 23 ([29])

1. *If $f = g$ for linear function g, then the test* Linearity-test$(f; n)$ *accepts with probability* 1.
2. *If the probability that the test* Linearity test *accepts is more than $1 - \delta$ for some $\delta < 2/9$ then there exists a linear function g such that $\Delta(f, g) \leq 2\delta$.*

Remark: The exact bound stated above in part 2 of Theorem 23 may not match the bound as stated in [29]. However this bound can be easily reconstructed from some of the subsequent papers, for instance, in the work of Gemmell et al [54]. In any case, the exact bound is unimportant for what follows.

Another useful aspect of the linear encoding is that one can obtain the value of the linear function at any point using few (randomly chosen) queries to any function that is very close to it. This procedure, described below, is also due to Blum, Luby and Rubinfeld [29].

```
Linear-self-corr(f, x; n):
/* Expects an oracle f : GF(2)^n → GF(2) and x ∈ GF(2)^n.   */

    Pick y ∈_R GF(2)^n;
    Output f(x + y) − f(y).
```

Proposition 24 ([29])

1. *If $f = g$ for linear function g, then the procedure* Linear-self-corr$(f, x; n)$ *outputs $g(x)$ with probability 1.*
2. *Given a function f that is δ-close to some linear function g, and any point $x \in GF(2)^n$, the procedure* Linear-self-corr$(f, x; n)$ *outputs $g(x)$ with probability at least $1 - 2\epsilon$.*

Concatenation It will be in our interest to construct a single oracle which represents the information content of many different oracles. We describe such a procedure next.

Definition 25 *For positive integers $n_1, \ldots, n_k$, if $f_1, \ldots, f_k$ are linear functions with $f_i : GF(2)^{n_i} \to GF(2)$, their* concatenated linear function linear-concat$_{f_1,\ldots,f_k}$ *is the function $f : GF(2)^{n_1+\cdots+n_k} \to GF(2)$ defined as follows:*

$$\textit{for } x_1 \in GF(2)^{n_1}, \ldots, x_k \in GF(2)^{n_k} \qquad f(x_1, \ldots, x_k) = \sum_{i=1}^{k} f_i(x_i).$$

We remark that every linear function $f : \mathrm{GF}(2)^{n_1+\cdots+n_k} \to \mathrm{GF}(2)$ is a concatenation of some k functions $f_1, \ldots, f_k : \mathrm{GF}(2)^{n_i} \to \mathrm{GF}(2)$ defined as

$$f_i(x) = f(0, \ldots, 0, x_i, 0, \ldots, 0) \qquad \forall x_i \in \mathrm{GF}(2)^{n_i},$$

where the x_i on the right hand side is the i'th argument of f.

Suppose we are given $f : \mathrm{GF}(2)^{n_1+\cdots+n_k} \to \mathrm{GF}(2)$ and wish to test if f_i in the above list is equal to some given linear function $f' : \mathrm{GF}(2)^{n_i} \to \mathrm{GF}(2)$. A randomized test suggests itself: pick $x_i \in \mathrm{GF}(2)^n$ at random and test if $f(0, \ldots, 0, x_i, \ldots, 0) = f'(x_i)$. We now show that a simple variant of this test works even if f and f' are not linear functions but only ϵ-close to some linear functions.

First we introduce some notation: For $i \in \{1, \ldots, k\}$ and $x \in \mathrm{GF}(2)^{n_i}$, the inverse projection of x is the vector $y = \pi_i^{-1}(x; n_1, n_2, \ldots, n_k) \in \mathrm{GF}(2)^{n_1+\cdots+n_k}$ which is 0 on all coordinates except those from $n_1 + \cdots n_{i-1} + 1$ to $n_1 + \cdots + n_i$, where it is equal to x_i.

Linear-concat-corr-test$(i, f, f'; n_1, \ldots, n_k)$:
/* /> Expects oracles $f : \mathrm{GF}(2)^{kn} \to \mathrm{GF}(2)$ and $f' : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$. */

Pick $x_i \in_R \mathrm{GF}(2)^n$;
If Linear-self-corr$(f, \pi_i^{-1}(x_i; n_1, \ldots, n_k); \sum_{i=1}^k n_i) = f'(x_i)$ then accept else reject.

Proposition 26

1. *If f and f' are linear functions such that f is the concatenation of k linear functions $f_1, \ldots, f_k$ where $f_j : GF(2)^{n_j} \to GF(2)$ with the ith function being f', then the procedure* Linear-concat-corr-test$(i, f, f'; n_1, \ldots, n_k)$ *accepts with probability* 1.
2. *If f and f' are ϵ close to linear functions g and g' respectively, for some $\epsilon < 1/4$, and the probability that the procedure* Linear-concat-corr-test$(i, f, f'; n_1, \ldots, n_k)$ *accept is greater than $1/2 + 3\epsilon$, then g is the concatenation of linear functions $g_1, \ldots, g_k$ where $g_j : GF(2)^{n_j} \to GF(2)$ with the ith function being g'.*

Proof: The proof of the completeness part is straightforward. For the second part assume for contradiction that g is not the concatenation of k linear functions with g' as the ith one. Then for a randomly chosen element x_i, $g(\pi_i^{-1}(x_i; n_1, \ldots, n_k))$ does not equal $g'(x_i)$ with probability 1/2. In order for the test to accept it must be the case that either x_i is such that $g(\pi_i^{-1}(x_i; n_1, \ldots, n_k)) = g'(x_i)$ or Linear-self-corr$(f, (\pi_i^{-1}(x_i; n_1, \ldots, n_k)); n_1 + \cdots + n_k) \neq g(\pi_i^{-1}(x_i; n_1, \ldots, n_k))$ or $g'(x_i) \neq f'(x_i)$. We upper bound the probability of these events by the sum of their probabilities. The middle event above happens with probability at most 2ϵ (by Proposition 24) and the final event with probability at most ϵ (since $\Delta(f', g') \leq \epsilon$). Thus the probability of any of these events happening is at most $1/2 + 2\epsilon + \epsilon$. The proposition follows. ∎

Quadratic Functions Before going on to describe the inner verifier of this section we need one more tool. Recall that the definition of L_x described it as a table (oracle) of values of a linear function at all its inputs. A dual way to think of L_x is as a table of the values of all linear functions at the input $x \in \mathrm{GF}(2)^n$. (Notice that there are exactly 2^n linear functions L_y from $\mathrm{GF}(2)^n$ to $\mathrm{GF}(2)$; and the value of the function L_y at x equals the value of L_x at y.) This perspective is more useful when trying to verify circuit satisfiability. For the sake of verifying circuit satisfiability it is especially useful to have a representation which allows one to find the values of all degree 2 functions at x. The following definition gives such a representation.

Definition 27 *For $x \in GF(2)^n$, the function* quad_x *is the map from $GF(2)^{n^2}$ to $GF(2)$ defined as follows: For the n^2 bit string $c = \{c^{(ij)}\}_{i=1,j=1}^{n,n}$,* $\mathsf{quad}_x(c) = \sum_{i=1}^n \sum_{j=1}^n c^{(ij)} x^{(i)} x^{(j)}$.

Observe that any quadratic function in x can be computed from the value of quad_x at one point, even if x is unknown. Suppose one is interested in the value of the polynomial $p(x) = \sum_{i \neq j} p^{(ij)} x^{(i)} x^{(j)} + \sum_i p^{(i)} x^{(i)} + p^{(00)}$. Then $p(x) = \mathsf{quad}_x(c) + p^{(00)}$, where $c^{(ij)} = p^{(ij)}$ if $i \neq j$, and $c^{ii} = p^i$. (Notice that since we are working over GF(2), the identity $x^2 = x$ holds for all $x \in \mathrm{GF}(2)$ and hence p will not contain any terms of the form $(x^{(i)})^2$.) Given any polynomial p of degree 2, weuse the notation coeff_p to denote the n^2 bit vector which satisfies $p(x) = \mathsf{quad}_x(\mathsf{coeff}_p)$ for all $x \in \mathrm{GF}(2)^n$.

The following observation will be useful in dealing with the quadratic representations.

Proposition 28 *For any $x \in GF(2)^n$, the function quad_x is linear.*

We now show how to check if two linear functions $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ and $f' : \mathrm{GF}(2)^{n^2} \to \mathrm{GF}(2)$ correspond to the linear and quadratic function representations respectively of the same string $x \in \mathrm{GF}(2)^n$. First we need some notation: Given $x, y \in \mathrm{GF}(2)^n$, $x@y$ is the n^2 bit vector with $(x@y)^{(ij)} = x^{(i)} y^{(j)}$.

Quadratic-consistency($f, f'; n$);
/* Expects $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ and $f' : \mathrm{GF}(2)^{n^2} \to \mathrm{GF}(2)$. */

  Pick $x, y \in_R \mathrm{GF}(2)^n$.
  If $f(x) \cdot f(y) = f'(x@y)$ then accept else reject.

The test above is based on Freivalds test [45] for matrix multiplication and its correctness is established as follows.

Proposition 29

1. *Given linear functions $f : GF(2)^n \to GF(2)$ and $f' : GF(2)^{n^2} \to GF(2)$, if for some $x \in GF(2)^n$, $f = L_x$ and $f' = \mathsf{quad}_x$, then the procedure* Quadratic-consistency($f, f'; n$) *accepts with probability* 1.
2. *For linear functions $f : GF(2)^n \to GF(2)$ and $f' : GF(2)^{n^2} \to GF(2)$, if the procedure* Quadratic consistency($f, f'; n$) *accepts with probability greater than 3/4, then there exists $x \in GF(2)^n$, such that $f = L_x$ and $f' = \mathsf{quad}_x$.*

Proof: Every linear function is of the form L_x for some $x \in \mathrm{GF}(2)^n$. Let x be such that $f = L_x$. Since f' is linear, there exists $b \in \mathrm{GF}(2)^{n^2}$, such that $f'(z) = \sum_{i=1}^n \sum_{j=1}^n b^{(ij)} z^{(ij)}$. Part (1) follows from the fact that if $b^{(ij)} = x^{(i)} x^{(j)}$ then $f'(z_1 @ z_2) = f(z_1) f(z_2)$. For part (2) we look at the $n \times n$ matrix $B = \{b^{(ij)}\}$. We would like to compare B with the $n \times n$ matrix $C = xx^T$. Assume for the sake of contradiction that the two matrices are not identical. Then Freivalds' probabilistic test [45] for matrix identity guarantees that for a randomly chosen bit vector z_2, the probability that the vectors Bz_2 and Cz^2 turn out to be distinct is at least 1/2. Now assume that $Bz_2 \neq Cz_2$ and consider their respective inner products with a randomly chosen vector z_1. We get

$$\Pr_{z_1 \in \mathrm{GF}(2)^n}\left[z_1^T Bz_2 \neq z_1^T Cz_2 | Bz_2 \neq Cz_2\right] \geq \frac{1}{2}.$$

By taking the product of the two events above we get

$$\Pr_{z_1, z_2 \in \mathrm{GF}(2)^n}\left[z_1^T Bz_2 \neq z_1^T Cz_2\right] \geq \frac{1}{4}.$$

But the left hand side in the above inequality is exactly $f'(z_1 @ z_2)$ and the quantity on the right hand side is $f(z_1)f(z_2)$. Thus if f' is not equal to quad_x, then the probability that $f'(z_1 @ z_2) \neq f(z_1)f(z_2)$ is at least 1/4. ▮

We now give an error-correcting version of the above test, when the functions f and f' are not linear but only close to linear functions.

```
Quadratic-corr-test(f, f'; n):
/* Expects oracle f : GF(2)^n → GF(2) and f' : GF(2)^(n^2) → GF(2).*/

   Pick z1, z2 ∈R GF(2)^n;
   If f(z1) · f(z2) ≠ Linear-self-corr(f', z1@z2; n)
         then reject
         else accept.
```

The following proposition can be proved as an elementary combination of Propositions 29 and 24.

Proposition 30

1. *Given linear functions $f : GF(2)^n \rightarrow GF(2)$ and $f' : GF(2)^{n^2} \rightarrow GF(2)$, if for some $x \in GF(2)^n$, $f = L_x$ and $f' = \mathsf{quad}_x$, then the procedure* Quadratic-corr-test$(f, f'; n)$ *accepts with probability* 1.
2. *Given functions $f : GF(2)^n \rightarrow GF(2)$, which is ϵ-close to some linear function g, and $f' : GF(2)^{n^2} \rightarrow GF(2)$, which is ϵ-close to some linear function g', if the procedure* Quadratic-corr-test$(f, f'; n)$ *accepts with probability greater than $3/4 + 4\epsilon$, then there exists $x \in GF(2)^n$, such that $g = L_x$ and $g' = \mathsf{quad}_x$.*

6 The Constant Bit Verifier: Putting it together

We now show how a verifier can verify the satisfiability of a circuit. Let C be a circuit of size n on km inputs. Suppose $x_1, \ldots, x_k$ form a satisfying assignment to C. To verify this suppose the verifier is given not only $x_1, \ldots, x_k$ but also the value of every wire on the circuit. Then the verifier only has to go check that for every gate the inputs and the output are consistent with each other. This forms the basis for our next two definitions.

Given k strings $x_1, \ldots, x_k$ of length m, the C-*augmented* representation of $x_1 x_2 \ldots x_k$, denoted $C\text{-aug}(x_1 \ldots x_k)$ is an n bit string z indexed by the wires of C, where the ith coordinate of z represents the value on the ith wire of the circuit C, on input $x_1 \ldots x_k$, with the property that the first km wires correspond to the input gates (i.e., $x_1 \ldots x_k$ is a prefix of $C\text{-aug}(x_1 \ldots x_k)$).

Given a circuit C of size n with km inputs, we associate with the circuit $n - km + 1$ polynomials P_j, $1 \leq j \leq n - km + 1$. For any fixed j, P_j is a polynomial of degree at most 2 on n variables $z^{(1)}, \ldots, z^{(n)}$, and is defined as follows. For $1 \leq j \leq n - km$, let the jth gate of C, be an addition gate with inputs being wires j_1 and j_2, and the let the output be the wire j_3. Then $P_j(z) = z^{(j_1)} + z^{(j_1)} - z^{(j_3)}$. (Similarly if the j gate had been a multiplication gate then $P_j(z)$ would be $z^{(j_1)} z^{(j_1)} - z^{(j_3)}$.) Lastly the polynomial $P_{n-km+1}(z)$ is the polynomial which is zero if and only if the output wire is accepting.

Thus C accepts on input $x_1 \ldots x_k$ iff

$$\exists z \in \mathrm{GF}(2)^n \text{ s.t. } \forall j \in [n - km + 1] P_j(z) = 0 \text{ and } \forall i \in [k] \pi_i(z) = x_i. \qquad (7)$$

However, the verifier will be given not $x_1, \ldots, x_k$ but merely functions that are purported to be $L_{x_1}, \ldots, L_{x_k}$. The verifier preceeds in two steps. First it checks that the provided functions are close to linear functions. Then, to check that the decodings of these functions satisfy the conditions in (7), it uses the procedures we described above.

Parity Verifier $V^{\oplus}$. Given a circuit C of size n on km inputs, the verifier $V^{\oplus}$ accesses oracles $X_1, \ldots, X_{k+1}$ all of which have answer size 1. The oracles X_1 to X_k take m bit strings as queries. The oracle X_{k+1} takes as input a pair (b, q) where $b \in \{0, 1\}$. Let A denote the oracle $X_{k+1}(0, \cdot)$ and let B denote the oracle $X_{k+1}(1, \cdot)$. Then $A : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ and $B : \mathrm{GF}(2)^{n^2} \to \mathrm{GF}(2)$. The verifier performs the following actions.

1. Linearity tests:
   (a) Linearity-test$(A; n)$.
   (b) Linearity-test$(B; n^2)$.
   (c) For $i = 1$ to k Linearity-test$(X_i; m)$.

2. *Consistency tests:* For $i = 1$ to k $\mathsf{Linear\text{-}concat\text{-}corr\text{-}test}(i, A, X_i; m, m, \ldots, n - mk)$.

3. *Quadratic tests:* $\mathsf{Quadratic\text{-}corr\text{-}test}(A, B; n)$.

4. *Circuit tests:*

   (a) Pick $r \in_R \mathrm{GF}(2)^{n-km+1}$.

   (b) Let $P : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ be the degree 2 polynomial given by $P(z) = \sum_{j=1}^{n-km+1} r^{(j)} P_j(z)$.

   (c) If $\mathsf{Linear\text{-}self\text{-}corr}(B, \mathsf{coeff}_P; n^2) \neq 0$ then reject.

5. If none of the tests above reject then accept.

We start by observing that it is easy to reorganize the description above so that $V^{\oplus}$ first tosses its random strings, and then generates a circuit C' and then queries the oracles $X_1, \ldots, X_{k+1}$ such that the function computed by C' on the responses of the oracles is the output of $V^{\oplus}$. The following proposition lists the parameters used by $V^{\oplus}$.

Proposition 31

1. *$V^{\oplus}$ tosses $2km + n(k+5) + 4n^2$ random coins.*

2. *$V^{\oplus}$ probes the oracles in $6k + 12$ places.*

3. *The computation of $V^{\oplus}$'s verdict as a function of the oracle responses can be expressed as a circuit of size at most 2^{6k+12}.*

We now show that the verifier $(V^{\oplus}, E^{\oplus}, (E^{\oplus})^{-1})$ satisfies the completeness condition.

Lemma 32 *For a circuit C of size n with km inputs, let the m bit strings $x_1, \ldots, x_k$ satisfy $C(x_1 \ldots x_k) = \mathsf{accept}$ and for $1 \leq i \leq k$, let $X_i = E^{\oplus}(x_i)$. Then there exists an oracle X_{k+1} such that $(V^{\oplus})^{X_1,\ldots,X_{k+1}}(C; R) = \mathsf{accept}$.*

Proof: Let z be the C-augmented representation of the string $x_1 \ldots x_k$. Let $A = E^{\oplus}(z)$ and let $B = \mathsf{quad}_z$ and let X_{k+1} be given by $X_{k+1}(0, \cdot) = A$ and $X_{k+1}(1, \cdot) = B$. It can be easily verified that the $X_1, \ldots, X_{k+1}$ as defined pass every one of the tests performed by $V^{\oplus}$. ▮

Lemma 33 *There exists a constant $e < 1$ (specifically, $e = 35/36$), such that if the verifier $V^{\oplus}$ accepts with probability e given oracles $X_1, \ldots, X_{k+1}$ and input C, then $C((E_m^{\oplus})^{-1}(X_1), \ldots, (E_m^{\oplus})^{-1}(X_k)) = \mathsf{accept}$.*

Proof: Let δ be chosen so as to minimize $e = \max\{1-\delta, 1/2+6\delta, 3/4+8\delta\}$. (This happens at $\delta = 1/36$.) We will show that this value of e suffices to prove the assertion.

Let $X_1, \ldots, X_{k+1} = (A, B)$ be oracles such that $V^{\oplus}$ accepts C with probability greater than e. Then by the fact that the linearity tests accept with probability at least $1-\delta$ and by Theorem 23 we find that there exist strings $x_1, \ldots, x_k$, z and z' such that the oracles X_i are 2δ close to the functions $E_m^{\oplus}(x_i)$ respectively; and the oracles A and B are 2δ close to L_z and $L_{z'}$ respectively.

Based on the fact that the quadratic consistency tests pass with probability at least $3/4+8\delta$, and the soundness condition of Proposition 29, we find that $L_{z'} = \mathsf{quad}_z$.

Let $z_1, \ldots, z_k$ be m bit strings which form the prefix of z. Next we use the fact that the acceptance probability of the verifier $V^{\oplus}$ in step 4(c) to show that $z = C$-aug$(z_1 \ldots z_k)$ and that C accepts on input $z_1 \ldots z_k$.

Claim 34 *If $V^{\oplus}$ accepts in step 4(c) with probability more than $1/2+4\delta$, then $z = C$-aug$(z_1, \ldots, z_k)$ and $C(z_1, \ldots, z_k) =$ accept.*

Proof: Suppose it is the case that z is not C-aug$(z_1 \ldots z_k)$ or that C does not accept $z_1 \ldots z_k$: Then it must be the case that there exists an index $j \in \{1, \ldots, n-mk+1\}$ such that the polynomial $P_j(z) \neq 0$. If this is the case, then for a randomly chosen vector $r \in \mathrm{GF}(2)^{n-mk+1}$, the polynomial $P = \sum_j r^{(j)} P_j$, will also be non-zero at z with probability $1/2$ (taken over the random string r.) Now the verifier $V^{\oplus}$ will reject in step 4(c) if $P(z)$ is non-zero and $B(\mathsf{coeff}_P) = \mathsf{quad}_z(\mathsf{coeff}_P)$. The latter identity fails to hold with probability at most 4δ (by Proposition 24). Thus with probability at most $1/2+4\delta$, the verifier will accept in step 4(c). ∎

We conclude by using the fact that the concatenation tests pass with probability at least $1/2+6\delta \leq 1-6\delta$ to see (by the soundness guarantee in Proposition 26) that L_z is the concatenation of $E^{\oplus}(x_1), \ldots, E_k^{\oplus}, f$ for some linear function $f : \mathrm{GF}(2)^{n-km} \rightarrow \mathrm{GF}(2)$. In particular this implies that the prefix of imply $z_i = x_i$ for all i.

Thus C accepts on input $(E_m^{\oplus})^{-1}(X_1) \ldots (E_m^{\oplus})^{-1}(X_k)$. ∎

Proof of Theorem 15: The inner verifier system $(V^{\oplus}, E^{\oplus}, (E^{\oplus})^{-1})$ yields the required system. Given $k < \infty$, let $c_1 = 3k+9$ and let $p = 6k+12$ and let e be as given by Lemma 33. Then, by Proposition 31 and Lemmas 32 and 33, it is clear that $(V^{\oplus}, E^{\oplus}, (E^{\oplus})^{-1})$ forms a $(k, c_1 \log n, p, 2^p, e)$ inner verifier system. ∎

7 Parallelization

The goal of this section is to prove Theorem 14 by constructing the inner verifier mentioned in it (i.e., one that only makes a constant number of queries, expects an oracle with polylogarithmic answer size, and uses logarithmic number of random bits). The theorem will be proved in Section 7.5. The starting point is a verifier constructed by Arora and Safra [5], which uses $O(\log n)$ random bits and queries the proof in $\mathrm{poly}(\log n)$ places. (Actually the number of queries in their strongest result is $\mathrm{poly}(\log\log n)$ but we don't need that stronger result. In fact, even the weaker verifier of Babai, Fortnow, Levin and Szegedy [12] would suffice for our purpose after some modification. This modification would cut down the number of random bits needed in [12] by using the idea of recycling randomness [31, 60].)

The following theorem was only implicit in the earlier versions of [5] but is explicit in the final version.

Theorem 35 ([5]) *For any constant k, there exist constants c_1, c_2, c_3 and $e < 1$ such that a $(k, c_1 \log n, \log^{c_2} n, \log^{c_3} n, 1, e)$ inner verifier exists.*

The shortcoming of this verifier is that it makes $\log^{c_2} n$ queries to the proof. We will show how to "aggregate" its queries into $O(1)$ queries, at the cost of a minor increase in the answer size (though the answer size remains $\mathrm{poly}(\log n)$). The aggregation uses the idea of a *multivariate polynomial encoding* [11] and a new, efficient, low-degree test for checking the correctness of its codewords. It also uses a procedure described in Section 7.4

7.1 Multivariate polynomial encodings

Let F be any finite field. The following fact will be used over and over again.

Proposition 36 *For every k distinct elements $x_1, \ldots, x_k \in F$ and k arbitary elements $y_1, \ldots, y_k \in F$, there exists a univariate polynomial p of degree at most $k-1$ such that $p(x_i) = y_i$ for all $i = 1, 2, \ldots, k$.*

Proposition 36 has an analogue for the multivariate case. Recall that $F_m^{(d)}$ is the family of m-variate polynomials over F of total degree at most d. Let H be a subset of F and $h = |H| - 1$. The multivariate analogue states that for each sequence of values for the set of points in H^m, there is a polynomial in $F_m^{(mh)}$ that takes those values on H^m. (This polynomial need not be unique.)

Proposition 37 *Let $H \subseteq F$ and $h = |H| - 1$. For each function $s\colon H^m \to H$, there is a polynomial $\hat{s} \in F_m^{(mh)}$ such that*

$$\hat{s}(u) = s(u) \qquad \forall u \in H^m.$$

Clearly, if s_1 and s_2 are two distinct functions from H^m to F, then $\hat{s_1} \neq \hat{s_2}$. Furthermore, if $mh/|F| < 0.25$, then Lemma 19 implies that the distance between $\hat{s_1}$ and $\hat{s_2}$ is at least 0.75.

We can also use Proposition 37 to define a map from bitstrings to polynomials. We fix an appropriate H, and use Proposition 37 to define a map from sequences of $(h+1)^m$ field elements to polynomials in $F_m^{(mh)}$. Of course, since $0, 1 \in F$, this also gives a way to encode bitstrings of length $(h+1)^m$ by polynomials in $F_m^{(mh)}$. If the bitstring has length $n < (h+1)^m$, we first extend it to a string of length $(h+1)^m$ in some canonical way.

Definition 38 *For n, m, H, F satisfying $|H|^m \geq n$ and $|F| \geq 2mh$, the (n, m, H, F)* polynomial encoding *$\mathcal{P}_{n,m,H,F}$ is a mapping from n-bit strings to the set $F_m^{(mh)}$ that was described in the previous paragraph.*

The polynomial decoding *function $\mathcal{P}^{-1}_{n,m,H,F}$ maps functions from $F^m \to F$ to n-bit strings and is defined as follows. Given any $r \colon F^m \to F$, first find the nearest polynomial $\tilde{r}$ in $F_m^{(mh)}$ (if there is more than one choice, break ties arbitrarily), then construct the sequence s of values of $\tilde{r}$ on H^m, then truncate s to its first n values, and finally, interpret each value as a bit in some canonical way (e.g., the most significant bit in its binary representation).*

Notice that the encoding/decoding pair defined above satisfy $\mathcal{P}^{-1}_{n,m,H,F} \circ \mathcal{P}_{n,m,H,F}(s) = s$, for every bitstring $s \in \{0,1\}^n$.

Note that if we are encoding a string of length n, we can choose $H^m = O(n)$ and $|F| = \text{poly}(h)$, so that the size of the encoding is $|F^m| = \text{poly}(n)$. These are the parameters we will use.

7.2 Lines representation of polynomials: Testing and correcting

In designing the verifier we will need two algebraic procedures. The first, called the *Low Degree Test*, efficiently determines if a given oracle $O : F^m \to F$ is δ-close to some degree d polynomial, where δ is some small constant. Low degree tests were invented as part of work on proof checking [11, 12, 40, 5, 54, 83, 84]. Efficiency is of paramount concern to us, so we would like the test to make as few queries to the oracle as possible. Most Low Degree Tests known make $\text{poly}(m, d, 1/\delta)$ queries. However, Rubinfeld and Sudan [84] give a test (which requires an auxiliary oracle in addition to O) whose number of queries depends on d but not on m. Arora and Safra [5] give a test whose number of queries depends only on m but not on d. We observe in this paper that the two analyses can be combined to give a test whose number of queries is independent of d and m. This test and its analysis is essentially due to Rubinfeld and Sudan. Our only new contribution is to use a lemma due to Arora and Safra that was unavailable when [84] was written.

The second procedure in this section does the following: Given an oracle $O : F^m \to F$ which is δ-close to a polynomial p, find the value of p at some specified point $x \in F^m$. This procedure is described below in Section 7.2.2.

7.2.1 Low Degree Test

To describe the low degree test, we need the notion of a *line* in F^m.

Definition 39 *Given $x, h \in F^m$, the* **line through** x **with slope** h*, denoted $l_{x,h}$, is the set of points $\{x + th | t \in F\}$. The line $l_{x,h}$ is thus a univariate subdomain of F^m parametrized by t. We let $l_{x,h}(t)$ denote the point $x + th$.*

Remark: Note that different $(x, h) \in F^{2m}$ can denote the same line. For instance, the lines $l_{x,h}$ and $l_{x,c\cdot h}$ are the same for every $c \in F \setminus \{0\}$.

Let $p: F^m \to F$ be a degree d polynomial and $l_{x,h}$ be any line. Let g be the function that is the *restriction* of p to line $l_{x,h}$. That is to say, $g: F \to F$ is the function defined as $g(t) = p(x + th)$. Then g is a univariate polynomial of degree d. Thus we see that the restriction of p to any line is a univariate polynomial of degree d. It is a simple exercise to see that the converse is also true if $|F| > 2d + 1$ (see for example [84]): If a function $p: F^n \to F$ is such that the restriction of p to every line in F^m is a univariate degree d polynomial, then f itself is a degree d polynomial. (See also [47] for a tight analysis of the condition under which this equivalence holds.) As we will see, the low degree test relies on a stronger form of the converse: if for "most" lines, there is a univariate degree d polynomial that agrees with f on "most" points of the line, then f itself is δ-close to a degree d polynomial.

The test receives as its input two oracles: the first a function $f: F^m \to F$, and the second an oracle B that gives, for each line in F^m, a univariate polynomial of degree d. In other words, B is a function from $\mathcal{L}$ to $F_1^{(d)}$, where $\mathcal{L}$ is the set of all lines in F^m.

But our lemma statements (and their proofs) will be rendered more elegant if we use the set F^{2m} in place of $\mathcal{L}$. Thus we think of B as a function from F^{2m} to $F_1^{(d)}$, such that for every pair $x, h \in F^m$, the value $B(x, h)$ is the univariate polynomial given in B for the line $l_{x,h}$. This notation involves some redundancy since, as already noted, there are many other pairs $(x', h') \in F^{2m}$ such that $l_{x,h}$ is the same line as $l_{x',h'}$. For all such pairs, we require that $B(x, h) = B(x', h')$. Such redundancy introduces some overcounting in our definitions and lemmas below, but does not affect their validity, since the overcounting introduced is the same for every $(x, h) \in F^{2m}$.

The above discussion motivates the following alternate representation of a degree d polynomial.

Definition 40 *Let $p \in F_m^{(d)}$ be an m-variate polynomial of degree d. The* **line representation** *of p is the function $\mathsf{lines}_p : F^{2m} \to F_1^{(d)}$ defined as follows: Given $x, h \in F^m$, the univariate polynomial $p' = \mathsf{lines}_p(x, h)$ is the polynomial obtained by restricting p to the univariate domain $l_{x,h}$.*

We now describe our test.

```
Poly-test(f, B; m, d, F):
/* Tests if oracle f : F^m → F is close to a degree d polynomial over F.
   Expects an auxiliary oracle B : F^{2m} → F_1^{(d)}.  */

   Pick x, h ∈_R F^m and t ∈_R F;
   If f(x + th) ≠ B(x, h)[t] then reject else accept.
```

The properties of the test are summarized below.

Theorem 41 ([84, 5]) *There exist constants $\delta_0 > 0$ and $\alpha < \infty$ such that for $\delta \leq \delta_0$, $d \in \mathcal{Z}^+$ if F is a field of cardinality at least $\alpha(d+1)^3$, then the following holds:*

1. *For any function $p \in F_m^{(d)}$, the probability that* Poly-test($p, \mathsf{lines}_p; m, d, F$) *accepts is* 1.
2. *If oracles f and B satisfy* $\Pr[$Poly-test$(f, B; m, d, F)$ *accepts* $] \geq 1 - \delta$, *then there exists a polynomial $p \in F_m^{(d)}$ such that $\Delta(f, p) \leq 2\delta$.*

We note that part (1) is trivial from the comments made above. The more nontrivial part (2) is a consequence, as already mentioned, of the work of Rubinfeld and Sudan [84] and Arora and Safra [5]. Since neither of these papers proves precisely this statement, we provide a proof in the Appendix (Section A). We note that the proof is a straightforward adaptation of the proof in [84]. The only new component is a result due to Arora and Safra [5] (Theorem 62 in the appendix), which was unavailable at the time [84] was written.

7.2.2 Correcting Polynomials

The procedure in this section is given an oracle $O : F^m \to F$ which is known to be close to a polynomial p. It needs to find the value of p at some specified point $x \in F^m$. We now describe a procedure which computes $p(x)$ using few probes into O and an auxiliary oracle B. The procedure owes its origins to the work of Beaver and Feigenbaum [15] and Lipton [71]. The specific analysis given below is borrowed from the work of Gemmell, Lipton, Rubinfeld, Sudan and Wigderson [54] and allows the number of queries to be independent of d, for error bounded away from 1.

```
Poly-self-corr(A, B, x; m, d, F):
/* Expects A : F^m → F, B : F^{2m} → F_1^{(d)} and x ∈ F^m.  */

   Pick h ∈_R F^m and t ∈_R F − {0};
   If B(x, h)[t] ≠ A(x + th) then reject else output B(x, h)[0].
```

Proposition 42

1. *For a polynomial $p \in F_m^{(d)}$ and its associated lines representation* lines_p*, the output of the procedure* $\mathsf{Poly\text{-}self\text{-}corr}(p, \mathsf{lines}_p, x; m, d, F)$ *is $p(x)$ with probability 1, for every $x \in F^m$.*

2. *If there exists a polynomial p such that $\Delta(A, p) = \epsilon < 0.01$, then $\forall x \in F^m$, $\forall B : F^{2m} \rightarrow F_1^{(d)}$,*

$$\Pr_{h,t}[\textit{Output of } \mathsf{Poly\text{-}self\text{-}corr}(A, B, x; m, d, F) \textit{ is not reject or } p(x)] \leq 2\sqrt{\epsilon} + \frac{d}{|F|} + \frac{1}{|F|^m}.$$

Proof: The proof of the first part is straightforward. For the second part, we follow the proof in Gemmell et al [54]. For $x, h \in F^m$ call the line $l_{x,h}$ *bad* if the quantity $\Pr_{t \in F - \{0\}}[p(x + th) \neq A(x + th)]$ is strictly greater than $\sqrt{\epsilon}$. By an application of Markov's inequality, we obtain that the probability (taken over h) that the line $l_{x,h}$ is *bad* is at most $\sqrt{\epsilon}$ (or else the fraction of points satisfying $p(x + th) \neq A(x + th)$ will be larger than ϵ).

Now consider the random choices of h and t. We have the following cases:

Case: The line $l_{x,h}$ is *bad*: This event happens with probability at most $\sqrt{\epsilon}$. In this case the procedure Poly-self-corr may output something other than reject or $p(x)$.

Case: $l_{x,h}$ is not bad: We now have the following subcases:

- Case: $B(x, h) = p|_{l_{x,h}}$: In this case the output is either $B(x, h[t]) = p(x + th)$ or reject.
- Case: $B(x, h) \neq p|_{l_{x,h}}$: Again we have the following subcases:
  - Case: $B(x, h)[t] \neq A(x + th)$: In this case the procedure rejects.
  - Case: $B(x, h)[t] = A(x + th) \neq p(x + th)$: This event happens with probability at most $\sqrt{\epsilon}$ (taken over the choice of t), since the line $l_{x,h}$ is not bad. In this case the procedure's output is not $p(x, h)$.
  - Case: $B(x, h)[t] = A(x + th) = p(x + th)$: For this event to happen the polynomials $B(x, h)$ and $p|_{x,h}$ must agree at t, an event which happens with probability at most $d/|F|$ for distinct degree d polynomials.

Thus, summing up over all the choices of the bad events, we find that the probability of not rejecting or producing as output $p(x)$ is at most $2\sqrt{\epsilon} + \frac{d}{|F|}$. ∎

7.3 Concatenation and testing

Recall that an inner verifier is presented a proof with many oracles. Each oracle contains an encoding of a binary string using some fixed encoding scheme. For purposes of this section the encoding used is the multivariate polynomial encoding. The procedure described next allows the verifier to do the following: given $k+1$ oracles $X, X_1, X_2, \ldots, X_k$, to check that X contains the information in $X_1, \ldots, X_k$.

Let $\zeta_1, \ldots, \zeta_{|F|}$ be some enumeration of the elements of F.

Definition 43 *Given polynomials $p_1, \ldots, p_k \in F_m^{(d)}$, their* concatenated polynomial $\mathsf{concat}_{p_1,\ldots,p_k}$ *is an element of $F_{m+1}^{(d+k-1)}$ which satisfies $\mathsf{concat}_{p_1,\ldots,p_k}(x, \zeta_i) = p_i(x)$, for every $i \in \{1, \ldots, k\}$ and $x \in F^m$.*

Remark: Notice that such a polynomial does indeed exist (which in addition maintains the individual degrees in the first m variables and has degree at most $k-1$ in the variable $x^{(m+1)}$). (This follows immediately from Proposition 36.) However, such a polynomial may not be unique. We assume that in such a case some such polynomial is picked arbitrarily.

Once again we describe a procedure to test if a given polynomial p is indeed the concatenation of k polynomials with the ith one being q.

```
Poly-concat-test(t, p, q; m, d, F):
/* Expects t ∈ F, p : F^{m+1} → F, and q : F^m → F.*/

    Pick x ∈_R F^m;
    If p(x,t) ≠ q(x) then reject else accept.
```

Proposition 44 *For polynomials $p \in F_{m+1}^{(d+k-1)}$ and $q \in F_m^{(d)}$, if p is a concatenated polynomial of $p_1, \ldots, p_k \in F_m^{(d+k-1)}$ with $p_i = q$, then* Poly-concat-test(ζ_i, p, q) *accepts with probability 1, else* Poly-concat-test(ζ_i, p, q) *accepts with probability at most $(d+k-1)/|F|$.*

Proof: Both parts are straightforward and follow from Proposition 19. ∎

In general the functions we will work with may not be actually be polynomials but only close to some polynomial. We now modify the above test appropriately to handle this situation.

```
Poly-corr-concat-test(t, A, B, C; m, d, F):
/* Expects t ∈ F, A : F^{m+1} → F, B : F^{2m} → F_1^{(d)}, and C : F^m → F. */

    Pick x ∈_R F^m;
    If Poly-self-corr(A, B, (x, t); (m + 1), d, F) ≠ C(x) then reject else accept.
```

Lemma 45 *Let $A : F^{m+1} \to F$ be an oracle which is ϵ-close to a polynomial $p \in F_{m+1}^{(d+k-1)}$ for $\epsilon < .01$. For an index $i \in \{1, \ldots, k\}$, let $p_i \in F_m^{(d+k-1)}$ be the polynomial given by $p_i(\cdot) = p(\cdot, \zeta_i)$. Given oracles $B : F^{2m} \to F_1^{(d+k-1)}$ and $C : F^m \to F$ the procedure* Poly-corr-concat-test(ζ_i, A, B, C) *behaves as follows:*

1. *If $A = p$, $B = \mathsf{lines}_p$ and $C = p_i$, then the procedure accepts with probability 1.*
2. *If there exists an oracle B such that the above procedure accepts with probability ρ, then $\Delta(C, p_i) < 1 - \rho + 2\sqrt{\epsilon} + \frac{d}{|F|}$.*

Proof: The first part is straightforward. For the second part we enumerate the different possible ways in which the procedure Poly-corr-concat-test may accept. For this to happen it must be the case that Poly-self-corr does not reject. Furthermore, either it must be the case that Poly-self-corr$(A, B, (x, \zeta_i); (m + 1), d, F)$ does not output $p(x, \zeta_i)$ or it must be the case that $p(x, \zeta_i) = C(x)$.

Now let $\delta = \Delta(C, p_i)$. Then for a randomly chosen x, the probability $\Pr_{x \in F^m}[p(x, \zeta_i) = C(x)]$ is at most $1 - \delta$. On the other hand, by Proposition 42, the quantity $\Pr_{x \in F^m}[p(x, \zeta_i) \neq$ Poly-self-corr$(A, B, (x, \zeta_i); (m + 1), d, F)]$ is at most $2\sqrt{\epsilon} + \frac{d}{|F|}$. Thus the probability that the procedure Poly-corr-concat-test accepts is at most $1 - \delta + 2\sqrt{\epsilon} + \frac{d}{|F|}$. The lemma follows by substituting $\rho = 1 - \delta + 2\sqrt{\epsilon} + \frac{d}{|F|}$. ∎

7.4 Curves and testing

This last section is the crucial part for parallelization. Here we are interested in the following task. We are given a polynomial $p \in F^m$ and we are interested in the value of p at l points $x_1, \ldots, x_l \in F^m$. We would like to find all the values by querying the oracle for p and some auxiliary oracle O at only a few points. The procedure Curve-check below will do this. We first introduce the notation of a curve.

Definition 46 *A* **curve** *through the vector space F^m is a function $\mathcal{C} : F \to F^m$, i.e., $\mathcal{C}$ takes a parameter t and returns a point $\mathcal{C}(t) \in F^m$. A curve is thus a collection of m functions $c_1, \ldots, c_m$, where each c_i maps elements from F to F. If each function $c_i \in F_1^{(d)}$, then $\mathcal{C}$ is a* **curve of degree** *d.*

The following lemma will be useful in our next definition.

Lemma 47 *Given l points $x_1, \ldots, x_l \in F^m$, and l distinct elements $t_1, \ldots, t_l \in F$, there exists a degree $l-1$ curve $\mathcal{C}$ such that $\mathcal{C}(t_i) = x_i$.*

Proof: Follows from Proposition 36. ▮

Definition 48 *Given points $x_1, \ldots, x_l \in F^m$, the curve $\mathsf{curve}_{x_1,\ldots,x_l}$ through $x_1, \ldots, x_l$ is a curve of degree $l-1$ with $\mathcal{C}(\zeta_i) = x_i$ for $i \in \{1, \ldots, l\}$. (Recall that $\zeta_1, \ldots, \zeta_{|F|}$ is a fixed enumeration of the elements of F.)*

Remark: By Lemma 47 we know that such a curve exists.

The basic idea for the "parallelization" is that instead of simply asking for the value of the polynomial p at the points $x_1, \ldots, x_l$, we will ask for its value on all points on the curve $\mathsf{curve}_{x_1,\ldots,x_l}$. We will need one more additional fact before we can go on.

Proposition 49 *Given a curve $C : F \to F^m$ of degree l and a polynomial $p \in F_m^{(d)}$, the function $f(t) = p(C(t))$ is a univariate polynomial in t of degree at most ld.*

We are now ready to describe the curves procedure:

$\mathsf{Curve\text{-}check}(p, O, x_1, \ldots, x_l; m, d, F)$:
`/* Expects` $p : F^m \to F$, $O : F^{ml} \to F_1^{(d(l-1))}$, $x_1, \ldots, x_l \in F^m$. `*/`

  `Pick` $t \in_R F$.
  `If` $O(x_1, \ldots, x_l)[t] \neq p(\mathsf{curve}_{x_1,\ldots,x_l}[t])$
    `then reject`
    `else output` $O(x_1, \ldots, x_l)[\zeta_1], \ldots, O(x_1, \ldots, x_l)[\zeta_l]$.

Proposition 50 *Given a degree d polynomial $p : F^m \to F$, if $O(x_1, \ldots, x_l) = p|\mathsf{curve}_{x_1,\ldots,x_l}$, then* Curve-check *outputs $p(x_1), \ldots, p(x_l)$ with probability 1. Conversely, if $O(x_1, \ldots, x_l) \neq p|\mathsf{curve}_{x_1,\ldots,x_l}$, then* Curve-check *accepts with probability at most $\frac{dl}{|F|}$.*

The proof of the above proposition is a straightforward consequence of the fact that two distinct degree dl univariate polynomials can agree in at most $\frac{dl}{|F|}$ places.

Finally we present a version of the above curve-check which works for functions which are only close to polynomials but not equal to them.

```
Curve-corr-check(A, B, O, x_1, ..., x_l; m, d, F):
/* Expects A : F^m → F, B : F^{2m} → F_1^{(d)}, O : F^{ml} → F_1^{(d(l−1))}, x_1, ..., x_l ∈ F^m.   */

    Pick t ∈_R F.
    If O(x_1, ..., x_l)[t] ≠ Poly-self-corr(A, B, (curve_{x_1,...,x_l}[t]); m, d, F)
       then reject
       else output O(x_1, ..., x_l)[ζ_1], ..., O(x_1, ..., x_l)[ζ_l].
```

Remark: Notice that Curve-corr-check only looks a one location in each of it three oracles.

By a straightforward combination of Propositions 42 and 50 we obtain the following:

Lemma 51 *Let $A : F^m \to F$ be an oracle which is ϵ-close to some polynomial $p \in F_m^{(d)}$ for $\epsilon < .01$. Given oracles $B : F^m \to F$ and oracle O, and points $x_1, \ldots, x_l$, the procedure* Curve-corr-check *behaves as follows:*

1. *If $A = p$, $B = \mathsf{lines}_p$ and $O(x_1, \ldots, x_l) = p|_{\mathsf{curve}_{x_1,\ldots,x_l}}$, then* Curve-corr-check *outputs $p(x_1), \ldots, p(x_l)$ with probability 1.*
2. *The probability that* Curve-corr-check *does not reject and outputs a tuple other than $(p(x_1), \ldots, p(x_l))$ is at most $2\sqrt{\epsilon} + \frac{d}{|F|} + \frac{dl}{|F|}$.*

7.5 Parallelization: Putting it together

In this section we prove Theorem 14.

Let V^{seq} be a verifier as guaranteed to exist by Theorem 35. We construct a new verifier V^{par}. Informally speaking, this new verifier expects the proof to contain a (multivariate) polynomial encoding of a bit string π, where π is a string that would have been accepted by V^{seq} with probability 1. Given such a proof, V^{par} first performs a low degree test (using only $O(1)$ queries to the proof; see Section 7.2.1) to check that the provided function is close to a polynomial. Next, V^{par} tries to simulate the actions of V^{seq} on the preimage of this polynomial (i.e., the string it is encoding). The simulation requires reconstructing poly$(\log n)$ values of the polynomial, which can be done using Procedure Curve-corr-check of Section 7.2. Again, only $O(1)$ queries to the proof are needed.

Now we formalize the above description. Let C be the circuit given as input to the verifier and n be its size and km the number of input nodes in C. Let r be the number of coins tossed by V^{seq} on input C. Let N be the size of the proof π examined by V^{seq}. Let $(E^{\mathsf{seq}}, (E^{\mathsf{seq}})^{-1})$ be the encoding/decoding scheme used by the verifier V^{seq} and let $x_1, \ldots, x_k$ be the k input oracles that V^{seq} works with. On random string $R \in \{0,1\}^r$, let the verifier's queries be given by the tuples

$(i_1(C,R),q_1(C,R)),\ldots,(i_l(C,R),q_l(C,R))$, where a pair (i,q) indicates that the ith oracle (x_i) is being queried with question q.

The verifier V^{par} fixes $w = \frac{\log N}{\log\log N}$, F to be a finite field of size $\Theta(\log^{2+c_2})N)$, and picks a subset H of cardinality $\log N$ (arbitrarily). Let $d = \frac{\log^2 N}{\log\log N} + k - 1$.

The verifier works with oracles $X_1,\ldots,X_{k+1}$: where for $1 \le i \le k$, the oracles X_i map F^w to F. The verifier defines, for its own use, three oracles using X_{k+1} as follows: Let $A(\cdot)$ denote $X_{k+1}(1,\cdot)$, $B(\cdot)$ denote $X_{k+1}(2,\cdot)$ and let $O(\cdot)$ denote the oracle $X_{k+1}(3,\cdot)$. Then A maps F^{w+1} to F, B maps F^{2w} to $F_1^{(d)}$ and O maps F^{wl} to $F_1^{d(l-1)}$. The verifier V^{par} now performs the following tests:

Verifier $V^{\mathsf{par}}(A,B,O,X_1,\ldots,X_k;w,d,l,r,F)$:
/* Expects oracles $A: F^{w+1} \to F$, $B: F^{2w} \to F_1^{(d)}$, $O: F^{(w+1)l} \to F_1^{(d(l-1))}$
and k oracles $X_1,\ldots,X_k : F^w \to F$. */

1. *Low-degree test*: Self-test$(A,B;(w+1),d,F)$;
2. *Concatenation tests*: For i = 1 to k, Concat-corr-check$(\zeta_i,A,B,X_i;w,d,F)$.
3. *Simulate V^{seq} test*:
   (a) Pick $R \in \{0,1\}^r$.
   (b) Let $(i_1(C,R),q_1(C,R)),\ldots,(i_l(C,R),q_l(C,R))$ be the questions generated by V^{seq} on random string R.
   (c) For $j = 1$ to l, Let $h_j \in H^w$ be such that $N_{w,H}(h_j) = q_j(C,R)$.
   (d) For $j = 1$ to l, Let $x_j = (h_j, \zeta^{(i_j(C,R))})$.
   (e) Let $(a_1,\ldots,a_l)$ be the output of Curve-corr-check$(A,B,O,x_1,\ldots,x_l;(w+1),d,F)$.
   (f) If any of the responses $a_1,\ldots,a_l$ is not a member of $\{0,1\}$ then reject.
   (g) If V^{seq} rejects on random string R and responses $a_1,\ldots,a_l$, then reject.
4. If none of the above procedures rejects then accept.

Proposition 52 *Given a circuit C of size n on km inputs, for some $k < \infty$, the verifier V^{par} exhibits the following properties:*

1. *There exists a constant α_k such that V^{par} tosses at most $\alpha_k \log n$ coins.*

2. *V^{par} probes the oracles $3k+5$ times.*

3. *There exists a constant β_k such that the answer size of the oracles are bounded by $\log^{\beta_k} n$.*

4. *There exists a constant γ_k such that the computation of $V^{\oplus}$'s verdict as a function of the oracles responses can be expressed as a circuit of size $\log^{\gamma_k} n$.*

Proof: Step 1 (polynomial self-testing) makes 2 queries one each into the oracles A and B. The total randomness used here is $(2(w+1)+1)\log|F|$. Step 2 (concatenation tests) makes 3 queries into the oracles A, B, X_i for each value of $i \in \{1,\ldots,k\}$. The number of random coins tossed in this step is $k(2(w+2))\log|F|$. In Step 3(a) V^{par} tosses r random coins. In addition in Step 3(e) it tosses $\log|F|(w+3)$ coins. Step 3(e) also makes three additional queries. Thus the total number of coins tossed is $(2wk+3w+2k+6)\log|F|+r \leq 13wk\log|F|+r$. By the choice of the parameters w and $|F|$ above this amounts to at most $13k(2+c_2)\log N + c_1\log n$ which in turn is at most $13k(2+c_2+1)c_1\log n$. Lastly since all of the verifier's tasks involve simple interpolation and field arithmetic, they can be done in time polynomial (more specifically in time cubic) in $\log|F|$ and d, k and l. ▮

We now analyze the completeness and soundness properties of V^{par}.

Lemma 53 *For a circuit C of km inputs and size n, let $x_1,\ldots,x_k$ satisfy $C(x_1,\ldots,x_k) =$ accept. For $i=1$ to k, let $X_i = \mathcal{P}_{n,w,H,F}(E^{\mathsf{seq}}(x_i))$. Then, there exists an oracle X_{k+1} such that, for all R,*

$$(V^{\mathsf{par}})^{X_1,\ldots,X_{k+1}}(C;R) = \mathsf{accept}.$$

Proof: Let π be the proof such that $(V^{\mathsf{seq}})^{E^{\mathsf{seq}}(x_1),\ldots,E^{\mathsf{seq}}(x_k),\pi}$ accepts C w.p. 1. Then let Π be the polynomial $\mathcal{P}_{N,w,H,F}(\pi)$. Let $A = \mathsf{concat}_{X_1,\ldots,X_k,\Pi}$. Let $B = \mathsf{lines}_A$. Let C be the oracle which on input $\mathcal{C}: F \to F^w$ responds with $A \circ \mathcal{C}$. Then it is clear that $(V^{\mathsf{par}})^{X_1,\ldots,X_{k+1}}$ accepts C with probability 1. ▮

Lemma 54 *There exists an error parameter $e < 1$, such that for oracles $X_1,\ldots,X_{k+1}$, if the verifier $(V^{\mathsf{par}})^{X_1,\ldots,X_{k+1}}$ accepts C with probability greater than e, then for $x_i = (E^{\mathsf{seq}})^{-1}(\mathcal{P}^{-1}_{n,w,H,F}(X_i))$, the circuit C accepts on input $x_1\ldots x_k$.*

Proof: Let δ_0 be as in Theorem 41. We will show that $e = \max\{.999, 1-\delta_0, 1-(1-e_{\mathsf{seq}})(.4)\}$ satisfies the condition above.

Suppose the test $\mathsf{Poly\text{-}self\text{-}test}(A,B;w+1,d,F)$ accepts A with probability $\max\{1-\delta_0, .999\}$. Then, by Theorem 41.2, there exists a degree d polynomial $p: F^{w+1} \to F$, such that $\Delta(A,p) \leq \min\{2\delta_0, .002\} \leq .002$.

Now suppose for each i, the concatenation tests accept with probability at least .999. Let $\epsilon = \Delta(A, p)$ and let $p_i(\cdot) = p(\cdot, i)$. Then, by Lemma 45, the distance $\Delta(X_i, p_i) \leq (1 - .999) + 2\sqrt{\epsilon} + \frac{d}{|F|}$ which (given the choice of d and $|F|$) is at most $1/4$. Thus p_i is the unique polynomial at distance $1/4$ from X_i.

Let $Y_i = \mathcal{P}^{-1}_{N,w,H,F}(X_i)$. Fix $R \in \{0,1\}^r$. We show that if the probability that the verifier V^{par} accepts in Step 3(e) is more than .6 then the verifier $(V^{\mathsf{seq}})^{Y_1,\ldots,Y_{k+1}}$ accepts on random string R. By Proposition 42 the probability that the procedure $\mathsf{Curve\text{-}corr\text{-}check}$ outputs something other than $p(z_1), \ldots, p(z_l)$ is at most $2\sqrt{\epsilon} + \frac{d}{|F|} + \frac{dl}{|F|}$, which, by our choice of $|F|, d$ and by our bound on ϵ, is at most .6. Thus if the probability that the verifier V^{par} accepts in step 3(e) is more than .6 it must be the case that the verifier V^{seq} accepts the answers $p(z_1), \ldots, p(z_l)$ on random string R (based on the action of V^{par} in Step 3(g)). But based on the success of the concatenation test we know that if the query (i_j, q_j) got converted to the point z_j in Steps 3(b)-3(d), then $p(z_j) = p_{i_j}(q_j) = Y_{i_j}(q_j)$. Thus it must be the case that V^{seq} on access to oracle $Y_1, \ldots, Y_k, \Pi$ would accept on random string R.

Finally, we conclude as follows: If the acceptance probability of V^{par} in Step 3(e) is greater than $1 - (1 - e_{\mathsf{seq}})(1 - .4)$, then the probability over $R \in \{0,1\}^r$ that V^{seq} accepts the oracles $Y_1, \ldots, Y_k, \pi$ must be at least $1 - e_{\mathsf{seq}}$. By the soundness property of V^{seq} it must be the case that $C((E^{\mathsf{seq}})^{-1}(Y_1), \ldots, (E^{\mathsf{seq}})^{-1}(Y_k)) =$ accept. But recall that Y_i was defined to be $\mathcal{P}^{-1}_{N,h,w,F}(X_i)$. Thus if we let x_i denote $(E^{\mathsf{seq}})^{-1}(\mathcal{P}^{-1}_{N,h,w,F}(X_i))$, then the above statement is restated as $C(x_1, \ldots, x_k) =$ accept. ▮

Finally, we prove Theorem 14.

Proof of Theorem 14: The system $(V^{\mathsf{par}}, \mathcal{P} \circ E^{\mathsf{seq}}, (E^{\mathsf{seq}})^{-1} \circ (\mathcal{P})^{-1})$ shall be the desired inner verifier system. Given $k < \infty$, let α_k, β_k and γ_k be as in Proposition 52. We let $c_1 = \alpha_k$, $p = 3k+5$, $c_2 = \gamma_k$, $c_3 = \beta_k$. Further, let e be as in Lemma 54. Then, by Proposition 52 and Lemmas 53 and 54, we see that $(V^{\mathsf{par}}, \mathcal{P} \circ E^{\mathsf{seq}}, (E^{\mathsf{seq}})^{-1} \circ (\mathcal{P})^{-1})$ forms a $(k, c_1 \log n, p, \log^{c_2} n, \log^{c_3} n, e)$ inner verifier system. ▮

8 Conclusions

We briefly mention some new results that have appeared since the first circulation of this paper.

New non-approximability results. Many new results have been proved regarding the hardness of approximation. Lund and Yannakakis [73] show that approximating the chromatic number of a graph within a factor n^ϵ is NP-hard, for some $\epsilon > 0$. They also show that if NP $\not\subseteq$ $Dtime(n^{\mathrm{poly}(\log n)})$, then Set Cover cannot be approximated within a factor $0.5 \log n$ in polynomial time. In a different work, Lund and Yannakakis [74] show hardness results for approximation versions of a large set of maximum subgraph problems. (These problems involve finding the largest subgraph that satisfies a a property Π, where Π is a nontrivial graph property closed under vertex deletion.) Khanna, Linial and Safra [66] study the hardness of coloring 3-colorable graph. They show that coloring a 3-colorable graph with 4 colors is NP-hard. Arora, Babai, Stern, and Sweedyk [2] prove hardness results for a collection of problems involving integral lattices, codes, or linear equations/inequations. These include Nearest Lattice Vector, Nearest Codeword, and the Shortest Lattice Vector under the ℓ_∞ norm. Karger, Motwani, and Ramkumar [65] prove the hardness of approximating the longest path in a graph to within a $2^{\log^{1-\epsilon} n}$ factor, for any $\epsilon > 0$.

There are many other results which we haven't mentioned here; see the compendium [35] or the survey [3].

Improved analysis of outer verifiers. Our construction of an efficient outer verifier for NP languages (Theorem 16) can be viewed as constructing a constant prover 1-round proof system that uses $O(\log n)$ random bits. (The "constant prover" means the same as "constant number of queries" in our context.) Recent results have improved the efficiency of this verifier. Bellare, Goldwasser, Lund and Russell [19] construct verifiers that use only 4 queries and logarithmic randomness to get the error down to an arbitrarily small constant (with polyloglog sized answer sizes). Feige and Kilian [41] construct verifiers with 2 queries, arbitrarily small error, and constant answer sizes. Tardos [90] shows how to get verifier that makes 3 queries and whose error goes down subexponentially in the answer size. Finally, all these constructions have been effectively superseded by Raz's proof [81] of the "parallel repetition conjecture." This conjecture was open for a long time, and allows constructions of a verifier that makes 2 queries, and whose error goes down exponentially with the answer size.

Better non-approximability results. Improved constructions of outer verifiers often lead to better non-approximability results. For instance, in the result for MAX-3SAT in this paper, we only demonstrate the existence of an $\epsilon > 0$ such that approximating MAX-3SAT within a factor $1 + \epsilon$ is NP-hard. But the improved verifier constructions mentioned above have steadily provided better and better values for this ϵ (it is now approximately 0.02, as shown by Bellare, Goldreich and

Sudan [18]). There has been much more progress on the MAX CLIQUE and the Chromatic Number problems.

1. MAX CLIQUE is hard to approximate to within a factor $n^{1-\epsilon}$, for any positive ϵ (Hastad [58]).
2. Chromatic number is hard to approximate to within a factor $n^{1-\epsilon}$, for any positive ϵ (Feige-Kilian[42]).

We note that the results for MAX CLIQUE and chromatic number are quite tight: the best algorithms approximate these problems within a factor $O(n/\log^2 n)$ and $O(n/\log^3 n)$ respectively. Feige [39] has recently proved a nonapproximability result for Set Cover that also closely matches the performance of the best known approximation algorithms.

A related issue is the following: "What is the smallest value of q for which NP $= \cup_{c>0}\mathrm{PCP}(c\log n, q)$?" We have shown that $q < \infty$, but did not give an explicit bound (though it could potentially be computed from this paper). But q has since been computed explicitly and then reduced through tighter analysis. It has gone from 29 [19] to to 22 [41] to 16 [18]. A parameter somewhat related to q is the *free-bit parameter* introduced by Feige and Kilian [41]. Reducing this parameter is necessary and sufficient for proving good inapproximability results for MAX CLIQUE [18], and Hastad has shown how to make this parameter smaller than every fixed $\delta > 0$.

Other technical improvements. As a side-product of our proof of the main theorem, we showed how to encode an assignment to a given circuit so that somebody else can check that it is a satisfying assignment by looking at $O(1)$ bits in the encoding. Babai [9] raised the following question: How efficient can this encoding be? In our paper, encoding an assignment of size n requires poly(n) bits. This was reduced to $n^{2+\epsilon}$ by Sudan [89]. The main hurdle in further improvement seemed to be Arora and Safra's proof [5] of Theorem 62, which requires a field size quadratic in the degree. Polishchuk and Spielman [80] present a new proof of Theorem 62 that works when the field size is linear. By using this new proof, as well as some other new ideas, they bring the size of the encoding down to $n^{1+\epsilon}$.

Some of the other results of this paper have also found new proofs with a careful attention to the constants involved. Friedl and Sudan [47] give an alternate proof of Theorem 41 and show that it works with $\delta_0 = 1/8$. The proof of the correctness of the linearity test (Theorem 23) has also been improved in works by Bellare, Goldwasser, Lund and Russell [19] and Bellare, Coppersmith, Hastad, Kiwi and Sudan [17].

Transparent math proofs. We briefly mention an application that received much attention in the popular press. Consider proofs of theorems of any reasonable axiomatic theory, such as Zermelo-Fraenkel set theory. A turing machine can check such proofs in time that is polynomial in the length of the proof. This means that the language

$$\{(\phi, 1^n) : \phi \text{ has a proof of length } n \text{ in the given system}\}$$

is in NP (actually, it is also NP-complete for most known systems). Our main theorem implies that a certificate of membership in this language can be checked probabilistically by using only $O(\log n)$ random bits and while examining only $O(1)$ bits in it. In other words, every theorem of the axiomatic system has a "proof" that can be checked probabilistically by examining only $O(1)$ bits in it. (Babai, Fortnow, Levin, and Szegedy [12] had earlier shown that proofs can be checked by examinining only poly$(\log n)$ bits in them.)

Actually, by working through our paper, a stronger statement can be obtained: there is a polynomial-time transformation between normal mathematical proofs and our "probabilistically checkable certificates of theoremhood."

8.1 Future Directions

Thus far our main theorem has been pushed quite far (much further than we envisioned at the time of its discovery!) in proving non-approximability results. We feel that it ought to have many other uses in complexity theory (or related areas like cryptography). One result in this direction is due to Condon et al [33], who use our main theorem (actually, a stronger form of it that we did not state) to prove a PCP-style characterization of PSPACE. We hope that there will be many other applications.

Acknowledgments

This paper was motivated strongly by the work of Arora and Safra [5] and we thank Muli Safra for numerous discussions on this work. We are grateful to Yossi Azar, Mihir Bellare, Tomas Feder, Joan Feigenbaum, Oded Goldreich, Shafi Goldwasser, Magnus Halldorsson, David Karger, Moni Naor, Steven Phillips, Umesh Vazirani and Mihalis Yannakakis for helpful discussions since the early days of work on this paper. We thank Manuel Blum, Lance Fortnow, Jaikumar Radhakrishnan, Michael Goldman and the anonymous referees for pointing out errors in earlier drafts and giving suggestions which have (hopefully) helped us improved the quality of the writeup.

References

[1] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems.* PhD thesis, U.C. Berkeley, 1994. Available from `http://www.cs.princeton.edu/~arora` .

[2] S. Arora, L. Babai, J. Stern and Z. Sweedyk. The hardness of approximating problems defined by linear constraints. *Proceedings of the Thirty Fourth Annual Symposium on the Foundations of Computer Science,* IEEE, 1993.

[3] S. Arora and C. Lund. Hardness of approximations. In *Approximation Algorithms for NP-hard problems*, D. Hochbaum, ed. PWS Publishing, 1996.

[4] S. Arora, R. Motwani, S. Safra, M. Sudan, and M. Szegedy. PCP and approximation problems. *Unpublished note*, 1992.

[5] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[6] G. Ausiello, A. D'Atri and M. Protasi. Structure Preserving Reductions among Convex Optimization Problems. *Journal of Computer and Systems Sciences*, vol. 21, pages 136–153, 1980.

[7] G. Ausiello, A. Marchetti-Spaccamela and M. Protasi. Towards a Unified Approach for the Classification of NP-complete Optimization Problems. *Theoretical Computer Science*, vol. 12, pages 83–96, 1980.

[8] L. Babai. Trading group theory for randomness. *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing,* ACM, 1985.

[9] L. Babai. Transparent (holographic) proofs. *Proceedings of the Tenth Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 665, Springer Verlag, 1993.

[10] L. Babai and L. Fortnow. Arithmetization: a new method in structural complexity theory. *Computational Complexity*, 1:41–66, 1991.

[11] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[12] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[13] L. Babai and K. Friedl. On slightly superlinear transparent proofs. *Univ. Chicago Tech. Report*, CS-93-13, 1993.

[14] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, v. 36, 254–276, 1988.

[15] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. *Proceedings of the Seventh Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 415, Springer Verlag, 1990.

[16] M. Bellare. Interactive proofs and approximation: reductions from two provers in one round. *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993.

[17] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi and M. Sudan. Linearity testing in characteristic two. *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science,* IEEE, 1995.

[18] M. Bellare, O. Goldreich and M. Sudan. Free bits, PCPs and non-approximability — towards tight results. *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science,* IEEE, 1995. TR95-024 of ECCC, the *Electronic Colloquium on Computational Complexity,* `http://www.eccc.uni-trier.de/eccc/`.

[19] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993. (See also Errata sheet in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994).

[20] M. Bellare and P. Rogaway. The complexity of approximating a nonlinear program. *Complexity of Numerical Optimization*, Ed. P.M. Pardalos, World Scientific (1993).

[21] M. Bellare and M. Sudan. Improved non-approximability results. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[22] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. *Proceedings of the Twentieth Annual Symposium on the Theory of Computing,* ACM, 1988.

[23] E. Berlekamp and L. Welch. Error correction of algebraic block codes. *US Patent* Number 4,633,470 (filed: 1986).

[24] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation* **96**, 77–94 (1992).

[25] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[26] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[27] M. Blum. Program checking. *Proc. FST&TCS*, Springer L.N.C.S. **560**, pp. 1–9.

[28] M. Blum and S. Kannan. Designing Programs that Check Their Work. *Proceedings of the Twenty First Annual Symposium on the Theory of Computing,* ACM, 1989.

[29] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:3, 1993.

[30] J.Y. Cai, A. Condon, and R. Lipton. PSPACE is Provable by Two Provers in One Round. *Proceedings of the Sixth Annual Conference on Structure in Complexity Theory*, IEEE, 1991.

[31] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[32] A. Condon. The complexity of the max word problem, or the power of one-way interactive proof systems. *Proceedings of the Eighth Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 480, Springer Verlag, 1991.

[33] A. Condon, J. Feigenbaum, C. Lund and P. Shor. Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993.

[34] S. Cook. The complexity of theorem-proving procedures. *Proceedings of the Third Annual Symposium on the Theory of Computing,* ACM, 1971.

[35] P. Crescenzi and V. Kann, A compendium of NP optimization problems. Technical Report, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", SI/RR-95/02, 1995. The list is updated continuously. The latest version is available by anonymous ftp from `nada.kth.se` as `Theory/Viggo-Kann/compendium.ps.Z`.

[36] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts. *SIAM Journal on Computing*, 23:4, pp. 864–894, 1994.

[37] W.F. De la Vega and G.S. Lueker. Bin Packing can be solved within $1+\epsilon$ in Linear Time. *Combinatorica*, vol. 1, pages 349–355, 1981.

[38] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard Karp (ed.), *Complexity of Computation*, AMS, 1974.

[39] U. Feige. A threshold of ln n for Set Cover. In *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1996.

[40] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. of the 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.

[41] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[42] U. Feige and J. Kilian. Zero knowledge and chromatic number. In *Proceedings of the Eleventh Annual Conference on Complexity Theory*, IEEE, 1996.

[43] U. Feige and L. Lóvasz. Two-prover one-round proof systems: Their power and their problems. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[44] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Proceedings of the Third Annual Conference on Structure in Complexity Theory*, IEEE, 1988.

[45] R. Freivalds. Fast Probabilistic Algorithms. *Proceedings of Symposium on Mathematical Foundations of Computer Science*, Springer-Verlag Lecture Notes in Computer Science, v. 74, pages 57–69, 1979.

[46] K. Friedl, Zs. Hátsági and A. Shen. Low-degree testing. *Proceedings of the Fifth Symposium on Discrete Algorithms*, ACM, 1994.

[47] K. Friedl and M. Sudan. Some improvements to low-degree tests. *Proceedings of the Third Israel Symposium on Theory and Computing Systems*, 1995.

[48] M. Furer. Improved hardness results for approximating the chromatic number. In *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science,* IEEE, 1995.

[49] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976.

[50] M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: motivation, examples and implications. *Journal of the ACM*, 25:499–508, 1978.

[51] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[52] M. Garey, D. Johnson and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science* **1**, pp. 237–267, 1976.

[53] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Info. Proc. Letters* **43** (1992), 169–174.

[54] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[55] M. Goemans and D. Williamson. .878 approximation algorithms for Max-CUT and Max-2SAT. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[56] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM J. on Computing*, 18(1):186–208, 1989.

[57] R.L. Graham. Bounds for certain multiprocessing anomalies, *Bell Systems Technical Journal*, 45:1563–1581, 1966.

[58] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. To appear in *Proceedings of the Thirty Seventh Annual Symposium on the Foundations of Computer Science,* IEEE, 1996.

[59] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[60] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[61] D. Johnson. Approximation algorithms for combinatorial problems. *J. Computer and Systems Sci.* **9** (1974), 256–278.

[62] V. Kann. Maximum bounded 3-dimensional matching is maxsnp-complete. *Information Processing Letters*, 37:27–35, 1991.

[63] N. Karmakar and R.M. Karp. An Efficient Approximation Scheme For The One-Dimensional Bin Packing Problem. *Proceedings of the Twenty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1982.

[64] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Advances in Computing Research, pages 85–103. Plenum Press, 1972.

[65] D. Karger, R. Motwani, and G.D.S. Ramkumar. On Approximating the Longest Path in a Graph. *Proceedings Workshop on Algorithms and Data Structures,* Lecture Notes in Computer Science (Springer-Verlag), vol. 709, pp. 421–430, 1993.

[66] S. Khanna, N. Linial and S. Safra. On the hardness of approximating the chromatic number. *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993.

[67] S. Khanna, R. Motwani, M. Sudan and U. Vazirani. On syntactic versus computational views of approximability. *Proceedings of the Thirty Fifth Annual Symposium on the Foundations of Computer Science,* IEEE, 1994.

[68] P. Kolaitis and M. Vardi. The decision problem for the probabilities of higher-order properties. *Proceedings of the Nineteenth Annual Symposium on the Theory of Computing,* ACM, 1987.

[69] D. Lapidot and A. Shamir. Fully Parallelized Multi-prover protocols for NEXP-time. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science,* IEEE, 1991.

[70] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973. A corrected English translation appears in an appendix to Trakhtenbrot [91].

[71] R. Lipton. New directions in testing. In J. Feigenbaum and M. Merritt, editors, *Distributed Computing and Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. American Mathematical Society, 1991.

[72] C. Lund, L. Fortnow, H. Karloff, N. Nisan. Algebraic Methods for Interactive Proof Systems. *J. ACM*, **39**, 859–868, 1992.

[73] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993.

[74] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. *Proceedings of ICALP 93,* Lecture Notes in Computer Science Vol. 700, Springer Verlag, 1993.

[75] R. Motwani. Lecture Notes on Approximation Algorithms. Technical Report, Dept. of Computer Science, Stanford University (1992).

[76] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences* **43**, pp. 425–440, 1991.

[77] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 1992.

[78] A. Paz and S. Moran. Non-deterministic polynomial optimization problems and their approximation. *Theoretical Computer Science*, vol. 15, pages 251–277, 1981.

[79] S. Phillips and S. Safra. PCP and tighter bounds for approximating MAXSNP. *Manuscript*, Stanford University, 1992.

[80] A. Polishchuk and D. Spielman. Nearly Linear Sized Holographic Proofs. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[81] R. Raz. A parallel repetition theorem. *Proceedings of the Twenty Seventh Annual Symposium on the Theory of Computing,* ACM, 1995.

[82] R. Rubinfeld. *A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs.* PhD thesis, U.C. Berkeley, 1990.

[83] R. Rubinfeld and M. Sudan. Testing polynomial functions efficiently and over rational domains. *Proceedings of the Third Symposium on Discrete Algorithms*, ACM, 1994.

[84] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing* 25:2, pp. 252–271, April 1996.

[85] S. Sahni. Approximate algorithms for the 0/1 knapsack problem, *Journal of the ACM*, 22:115–124, 1975.

[86] S. Sahni and T. Gonzales. P-complete approximation problems. *J. of the ACM*, 23:555–565, 1976.

[87] J. T. Schwartz. Probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980.

[88] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992.

[89] M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* Ph.D. Thesis, U.C. Berkeley, 1992. Also appear as ACM Distinguished Theses, Lecture Notes in Computer Science, no. 1001, Springer, 1996.

[90] G. Tardos. Multi-prover encoding schemes and three prover proof systems. *Proceedings of the Ninth Annual Conference on Structure in Complexity Theory*, IEEE, 1994.

[91] B. A. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing* vol. 6, pages 384–400, 1984.

[92] M. Yannakakis. On the approximation of maximum satisfiability. *Proceedings of the Third Symposium on Discrete Algorithms*, ACM, 1994.

[93] D. Zuckerman. NP-Complete Problems have a version that is hard to Approximate. *Proceedings of the Eighth Annual Conference on Structure in Complexity Theory*, IEEE, 1993.

A Correctness of the Low Degree Test

This section proves the correctness of the Low Degree test by proving part (2) of Theorem 41.

Let m and d be arbitrary positive integers that should be considered fixed in the rest of this section. Let $F = GF(q)$ be a field. Recall that the test receives as its input two oracles: the first a function from F^m to F, and the second an oracle B that gives, for each line in F^m, a univariate polynomial of degree d. As already explained in Section 7.2, we will view B as a function from F^{2m} to $F_1^{(d)}$ (recall that

$F_1^{(d)}$ is the set of univariate degree d polynomials over F). We will call any such oracle a *d-oracle.*

Definition 55 *Let $f: F^m \to F$ be any function, $(x, h) \in F^{2m}$ be a line, and g be a univariate degree d polynomial. For $t \in F$, we say that g* **describes f at the point** *$l_{x,h}(t)$ if*

$$f(x + th) = g(t).$$

Recall that the test picks a random line and a random point on this line, and checks whether the univariate polynomial provided in B for this line describes f at this point.

Therefore we define the *failure rate of B with respect to f,* denoted $\delta_f^{(B)}$, as

$$\delta_f^{(B)} \equiv \Pr_{x,h \in F^m, t \in F} \left[f(x + th) \neq B(x, h)[t] \right]. \tag{8}$$

We wish to prove part (2) of Theorem 41, which asserts the existence of some $\delta_0 > 0$ (independent of d, q, m) for which the following is true: if $f: F^m \to F$ is any function and B is any d-oracle such that $\delta_f^{(B)} \leq \delta_0$, then f is $2\delta_f^{(B)}$-close to $F_m^{(d)}$. (Note that the Theorem also has a technical requirement, namely that $|F|$ is sufficiently large.)

To simplify the proof exposition, we first give a simple characterization of the d-oracle B that minimizes $\delta_f^{(B)}$.

Definition 56 *For a function $f : F^m \to F$ and points $x, h \in F^m$, a* **degree d line polynomial for f on the line** *$l_{x,h}$ (or just* line polynomial *on $l_{x,h}$, when d and f are understood from context) is a univariate degree d polynomial that describes f on more (or at least as many) points on $l_{x,h}$ than every other univariate degree d polynomial. We let $P_{x,h}^{(f,d)}$ denote this polynomial. (If there is more than one choice possible for $P_{x,h}^{(f,d)}$, we choose between them arbitrarily.)*

Remark: The line polynomial may be equivalently defined as a univariate degree d polynomial that is closest to the restriction of f to the line. Thus it follows from Lemma 19 that if the restriction is $(1/2 - d/|F|)$-close, then the line polynomial is uniquely defined.

For $f: F^m \to F$, we denote by $B^{(f,d)}$ a d-oracle in which $B^{(f,d)}(x, h) = P_{x,h}^{(f,d)}$ for every $(x, h) \in F^{2m}$. We denote by $\delta_{f,d}$ the average failure rate of $B^{(f,d)}$ with respect to f.

Proposition 57 *If $f: F^m \to F$ is any function and $d > 0$ any integer, then every d-oracle has failure rate at least $\delta_{f,d}$ with respect to f.*

Proof: Let C be any oracle. The definition of a line polynomial implies that for every line $(x, h) \in F^{2m}$, the number of points in $l_{x,h}$ at which the polynomial $C(x, h)$ describes f is no more than the number at which $B^{(f,d)}$ describes f. By averaging over all (x, h), the proposition follows. ∎

It should be clear from Proposition 57 that the following result suffices to prove part 2 of Theorem 41.

Theorem 58 (The Section's Main Theorem) *There are fixed constants $\delta_0, c > 0$ such that the following is true. For every integers $m, d > 0$ and every field F of size cd^3, if $f: F^m \to F$ is a function such that the average failure rate of $B^{(f,d)}$ with respect to f is $\gamma \leq \delta_0$, then f is 2γ-close to $F_m^{(d)}$.*

We note that our proof of Theorem 58 will provide an explicit description of the polynomial closest to f. For any function $f : F^m \to F$, define another function $\hat{f}_d: F^m \to F$ as

$$\hat{f}_d(x) \equiv \text{plurality}_{h \in F^m} \left\{ P_{x,h}^{(f,d)}(0) \right\} \qquad \forall x \in F^m. \tag{9}$$

(Note that the line $l_{x,h}$ passes through x. Further, x is the point $x + 0 \cdot h$, so $P_{x,h}^{(f,d)}(0)$ is the value produced by the line polynomial $P_{x,h}^{(f,d)}$ at x. Thus $\hat{f}_d(x)$ is the most popular value produced at x by the line polynomials of the lines passing through x.)

The proof of Theorem 58 will show, first, that f is close to $\hat{f}_d$ and second, that $\hat{f}_d$ is a degree d polynomial. The first statement is easy and is proved in the next lemma. The proof of the second statement takes up the rest of the section.

Lemma 59 *For any function $f : F^m \to F$ and any integer d, $\Delta(f, \hat{f}_d) \leq 2\delta_{f,d}$.*

Proof: Let B be the set given by

$$B = \left\{ x \in F^m \mid \Pr_{h \in F^m}[f(x) \neq P_{x,h}^{(f,d)}(0)] > 1/2 \right\}.$$

Now imagine picking $(x, h) \in F^{2m}$ randomly. We have

$$\begin{aligned} \Pr_{x,h}[f(x) \neq P_{x,h}^{(f,d)}(0)] &\geq \frac{1}{2} \cdot \Pr_x[x \in B] && (10) \\ &= \frac{|B|}{2\,|F|^m}. && (11) \end{aligned}$$

But by the definition of $\delta_{f,d}$, we also know that

$$\Pr_{x,h}[f(x) \neq P_{x,h}^{(f,d)}(0)] = \delta_{f,d}. \tag{12}$$

From (11) and (12) we conclude that $\frac{|B|}{2|F|^m} \leq \delta_{f,d}$. Furthermore, the definition of $\hat{f}_d$ implies that every $x \notin B$ satisfies $\hat{f}_d(x) = f(x)$. Hence we have

$$\Delta(f, \hat{f}_d) \leq \frac{|B|}{|F|^m} \leq 2\delta_{f,d}.$$

Thus the lemma has been proved. ∎

For the rest of the proof of Theorem 58, we will need a certain lemma regarding bivariate functions. The following definition is required to state it.

Definition 60 *For the bivariate domain $F \times F$, the* **row (resp., column)** *through x_0 (resp., y_0), is the set of points $\{(x_0, y)|y \in F\}$ (resp., $\{(x, y_0)|x \in F\}$).*

Notice that rows and columns are lines. The notion of a line polynomial specializes to rows and columns as follows.

Definition 61 *For a function $f : F^2 \to F$ and a row through x_0 the* **row polynomial**, *is a univariate polynomial of degree d that agrees with f on more (or at least as many) points on the row as any other univariate degree d polynomial. We let $r_{x_0}^{(f,d)}$ denote this polynomial. (*If there is more than one choice for $r_{x_0}^{(f,d)}$, we pick one arbitrarily.*) We likewise define the* **column polynomial** *$c_{y_0}^{(f,d)}$ for the column through $y_0 \in F$.*

The next result (which we will not prove) shows that if f is a bivariate function such that its row and column polynomials agree with it in "most" points, then f is close to a bivariate polynomial. This result can be viewed as proving the subcase of Theorem 58 when the number of variables, m, is 2. This subcase will be crucial in the proof for general m.

Theorem 62 ([5]) *There are constants $\epsilon_0, c > 0$ such that the following holds. Let d be any positive integer, ϵ a fraction less than ϵ_0, and F a field of cardinality at least cd^3. For $f : F^2 \to F$, let $R, C : F^2 \to F$ be the functions defined as $R(x, y) = r_x^{(f,d)}(y)$ and $C(x, y) = c_y^{(f,d)}(x)$. If f satisfies $\Delta(f, R) \leq \epsilon$ and $\Delta(f, C) \leq \epsilon$, then there exists a polynomial $g : F^2 \to F$ of degree at most d in each of its variables, such that $\Delta(f, g) \leq 4\epsilon$.* ∎

Corollary 63 *Let $f\colon F^m \to F$ satisfy the hypothesis of Theorem 62 and let c, ϵ_0 be the constants mentioned in its conclusion. Let $\epsilon' > 0$ be such that $5\epsilon + \epsilon' + \frac{(d+1)}{|F|} < 1/2$.*

Then every two $x_0, y_0 \in F$ that satisfy

$$\Pr_x[f(x, y_0) \neq r_x^{(f,d)}(y_0)] \leq \epsilon' \tag{13}$$

$$\Pr_y[f(x_0, y) \neq c_y^{(f,d)}(x_0)] \leq \epsilon' \tag{14}$$

also satisfy $r_{x_0}^{(f,d)}(y_0) = c_{y_0}^{(f,d)}(x_0)$.

Proof: Theorem 62 implies there exists a polynomial g of degree d each in x and y such that $\Delta(f, g) \leq 4\epsilon$. Since $\Delta(f, C) \leq \epsilon$ and $\Delta(f, R) \leq \epsilon$, we conclude from the triangle inequality that $\Delta(g, C), \Delta(g, R) \leq 5\epsilon$.

To prove the desired statement it suffices to show that $r_{x_0}^{(f,d)}(y_0)$ and $c_{y_0}^{(f,d)}(x_0)$ are both equal to $g(x_0, y_0)$. For convenience, we only show that $c_{y_0}^{(f,d)}(x_0) = g(x_0, y_0)$; the other case is identical.

For $x \in F$, let $R(x, \cdot)$ denote the univariate degree d polynomial that describes values of R on the row that passes through x. Define $g(x, \cdot)$ and $g(\cdot, y)$ similarly. Now we claim that

$$\Pr_{x \in F}[g(x, \cdot) \neq R(x, \cdot)] \quad \leq \quad 5\epsilon + \frac{d}{|F|} \tag{15}$$

The reason is that $R(x, \cdot)$ and $g(x, \cdot)$ are univariate degree d polynomials, so if they are different, then they disagree on at least $|F| - d$ points. Since $\Delta(g, R) = \Pr_{x,y}[g(x, y) \neq f(x, y)]$, we have

$$\Delta(g, R) \geq \quad (1 - \tfrac{d}{|F|}) \Pr_{x \in F}[g(x, \cdot) \neq R(x, \cdot)],$$

which implies $\Pr_x[g(x, \cdot) \neq R(x, \cdot)] \leq 5\epsilon/(1 - \frac{d}{|F|})$, and it is easily checked that $5\epsilon/(1 - \frac{d}{|F|}) \leq 5\epsilon + \frac{d}{|F|}$.

According to the hypothesis of the corollary, $y_0 \in F$ is such that $\Pr_x[f(x, y_0) \neq r_x^{(f,d)}(y_0)] \leq \epsilon'$. In conjunction with (15), this implies that

$$\Pr_x [f(x, y_0) \neq g(x, y_0)] \leq 5\epsilon + \epsilon' + \frac{d}{|F|}.$$

But the previous statement just says that the univariate polynomial $g(x, y_0)$ describes f on $(5\epsilon + \epsilon' + \frac{d}{|F|})\,|F|$ points on the column through y_0. Further, the hypothesis also says that $5\epsilon + \epsilon' + \frac{d}{|F|} < 1/2 - \frac{d}{|F|}$, so we conclude that $g(\cdot, y_0)$ is simply the column polynomial for the column through y_0, namely, $c_{y_0}^{(f,d)}$. Hence it follows that $c_{y_0}^{(f,d)}(x_0) = g(x_0, y_0)$, which is what we desired to prove. ▮

Now we return to the m-variate case of Theorem 58. The next lemma shows that if the failure rate $\delta_{f,d}$ is small, then the line polynomials of f are mutually quite consistent.

Lemma 64 *Let ϵ_0 and c be as in Theorem 62, d be any positive integer, F be a field of size at least* $\max\{3(d+1), cd^3\}$*, and ϵ be any constant satisfying $\epsilon < \min\{\frac{1}{36}, \epsilon_0\}$. Then every function $f : F^m \to F$ satisfies:*

$$\forall x \in F^m, t_0 \in F, \quad \Pr_{h_1,h_2}\left[P_{x,h_1}^{(f,d)}(t_0) \neq P_{x+t_0h_1,h_2}^{(f,d)}(0)\right] \leq \frac{4\delta_{f,d}}{\epsilon} + \frac{4}{|F|}.$$

Remark: Note that when $h_1 \in F^m$ is random, the line l_{x,h_1} is a random line through x. When $h_2 \in F^m$ is also random, the line $l_{x+t_0h_1,h_2}$ is a random line through $x + t_0h_1$. The two lines intersect at the point $x + t_0h_1$.

Proof: We use the shorthand δ for $\delta_{(f,d)}$. Pick $h_1, h_2 \in_R F^m$ and let $m : F^2 \to F$ be the function given by $m(y,z) = f(x + yh_1 + zh_2)$. Note that by definition, $P^{(f,d)}_{x,h_1} = c^{(m,d)}_{z_0}$ and $P^{(f,d)}_{x+t_0h_1,h_2} = r^{(m,d)}_{y_0}$

The Lemma will be proved by showing that with probability at least $1-4(\delta/\epsilon+1/|F|)$ (over the choice of h_1 and h_2), m satisfies the conditions required for the application of Corollary 63 for $y_0 = t_0$ and $z_0 = 0$ and $\epsilon' = \epsilon$. Then it will follow that $c^{(m,d)}_{z_0}(y_0) = r^{(m,d)}_{y_0}(z_0)$. This, of course, proves the Lemma since, as already noted, $P^{(f,d)}_{x,h_1} = c^{(m,d)}_{z_0}$ and $P^{(f,d)}_{x+t_0h_1,h_2} = r^{(m,d)}_{y_0}$.

The first hypothesis for Corollary 63 certainly holds: ϵ and ϵ' satisfy $5\epsilon + \epsilon' + (d + 1)/|F| < 1/2$.

Next, note that $x + yh_1$ and h_2 are random and independent of each other. Thus, by the definition of δ, we have

$$\forall y \neq 0, \forall z, \quad \Pr_{h_1,h_2}\left[P^{(f,d)}_{x+yh_1,h_2}(z) \neq f(x + yh_1 + zh_2)\right] \leq \delta. \tag{16}$$

Note that the event $P^{(f,d)}_{x+yh_1,h_2}(z) \neq f(x+yh_1+zh_2)$ may be rephrased as $r^{(m,d)}_y(z) \neq m(y,z)$. Now, applying Markov's inequality to (16), we obtain

$$\Pr_{h_1,h_2}\left[\Pr_{y\neq 0,z}[m(y,z) \neq r^{(m,d)}_y(z)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon}.$$

By accounting for the probability of the event $y = 0$, we can conclude:

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[m(y,z) \neq r^{(m,d)}_y(z)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{17}$$

Applying Markov's Inequality again to (16), but this time fixing $z = z_0$, we get

$$\Pr_{h_1,h_2}\left[\Pr_{y}[m(y,z_0) \neq r^{(m,d)}_y(z_0)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{18}$$

We can similarly argue that

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[m(y,z) \neq c^{(m,d)}_z(y)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{19}$$

$$\text{and} \Pr_{h_1,h_2}\left[\Pr_{z}[m(y_0,z) \neq c^{(m,d)}_z(y_0)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{20}$$

Thus with probability at least $1-4(\delta/\epsilon+1/|F|)$, none of the events (17)-(20) happen and all the hypotheses of Corollary 63 are satisfied.

The next corollary relates the values produced by the line polynomials to the values of the "corrected" function $\hat{f}_d$.

Corollary 65 *Let ϵ_0, c be as in Theorem 62 and ϵ be any constant less than $\min\{\frac{1}{36}, \epsilon_0\}$. Every function $f : F^m \to F$ satisfies*

$$\forall x \in F^m, t \in F, \quad \Pr_{h\in F^m}\left[\hat{f}_d(x + th) \neq P^{(f,d)}_{x,h}(t)\right] \leq \frac{8\delta}{\epsilon} + \frac{8}{|F|} \tag{21}$$

if F is a finite field of size at least $\max\{3(d + 1), cd^3\}$.

Proof: Let $B_{x,t}$ be the set defined as

$$B_{x,t} = \left\{ h \in F^m | P_{x,h}^{(f,d)}(t) \neq \text{plurality}_{h'}\{P_{x+th'}^{(f,d)}(0)\} \right\}.$$

Note that every $h \notin B$, automatically satisfies $\hat{f}_d(x + th) = P_{x,h}^{(f,d)}(t)$. So the probability appearing on the left hand side of (21) is bounded from above by $\frac{|B_{x,t}|}{|F|^m}$. We will show that $\frac{|B_{x,t}|}{|F|^m} \leq 2 \cdot (4\delta/\epsilon + 4/|F|)$.

Imagine picking $h, h_1 \in F^m$ randomly. According to Lemma 64, the probability that $P_{x+th,h_1}^{(f,d)}(0) \neq P_{x,h}(t)$ is at most $(4\delta/\epsilon + 4/|F|)$.

But for each $h \in B_{x,t}$, the probability that a randomly chosen $h_1 \in F^m$ satisfies $P_{x+th,h_1}^{(f,d)}(0) \neq P_{\hat{x},\hat{h}}(t)$ is, by definition of $B_{x,t}$, at least $1/2$. Thus if we pick $h, h_1 \in F^m$ randomly, the probability that $P_{x+th,h_1}^{(f,d)}(0) \neq P_{x,h}(t)$ is at least $\frac{|B_{x,t}|}{2|F|^m}$.

Hence we conclude that $\frac{|B_{x,t}|}{|F|^m} \leq 2(4\delta/\epsilon + 4/|F|)$. ▮

Even the specialization of the lemma above to the case $t = 0$ is particularly interesting, since it says that the "plurality" in the definition of $\hat{f}_d$ is actually an overwhelming majority, provided $\delta_{f,d}$ is sufficiently small. The next lemma essentially shows that $\hat{f}_d$ is a degree d polynomial.

Lemma 66 *Let ϵ_0 and c be as in Theorem 62. Let F be a finite field of size at least $\max\{3(d+1), cd^3\}$ and $\epsilon = \min 1/36, \epsilon_0$. If $f : F^m \to F$ is any function for which $\delta = \delta_{f,d}$ satisfies*

$$\frac{256\delta}{\epsilon^2} + \frac{256}{\epsilon|F|} + \frac{56\delta}{\epsilon} + \frac{40}{|F|} < 1,$$

then

$$\forall x, h \in F^m \quad \hat{f}_d(x) = P_{x,h}^{(\hat{f}_d,d)}(0)$$

Proof: Pick $h_1, h_2 \in_R F^m$ and define $m : F^2 \to F$ to be

$$m(y, 0) = \hat{f}_d(x + yh) \text{ and } m(y, z) = f(x + yh + zh_1 + yh_2)) \text{ for } z \neq 0.$$

We will show, by an invocation of Corollary 63, that the event $c_0^{(m,d)}(0) = r_0^{(m,d)}(0)$ happens with probability strictly greater than $8\delta/\epsilon + 8/|F|$ over the random choices of h_1 and h_2. But by Corollary 65, we have $m(0, 0) = \hat{f}_d(x) = P_{x,h_1}^{(f,d)}(0) = c_0^{(m,d)}(0)$ with probability at least $1 - 8\delta/\epsilon - 8/|F|$. (Of the three equalities in the chain here — the first and the third are by definition and the middle one uses Corollary 65.) Our choice of ϵ, δ implies that the following event happens with with positive probability (over the choice of h_1 and h_2): "$\hat{f}_d(x) = r_0^{(m,d)}(0) = P_{x,h}^{(g,d)}(0)$." But this event does not mention h_1 and h_2 at all, so its probability is either 1 or 0. Hence it must be 1, and the Lemma will have been proved.

Thus to prove the Lemma it suffices to show that the conditions required for Corollary 63 are true for the function m, with $\epsilon' = \epsilon$ and $y_0 = z_0 = 0$.

For $z = 0$ and all y, we have $m(y, z) = \hat{f}_d(x+yh+z(h_1+yh_2))$, by definition. For any $z \neq 0$ and y, the probability $\Pr_{h_1,h_2}[m(y,z) \neq \hat{f}_d(x+yh+z(h_1+yh_2))]$ is at most 2δ (by Lemma 59). Also, for all $y, z \in F$, the probability that $\hat{f}_d(x+yh+z(h_1+yh_2))$ does not equal $P^{f,d}_{x+yh,h_1+yh_2}(z)$ is at most $8\delta/\epsilon + 8/|F|$ (by Corollary 65). Thus we have shown

$$\Pr_{h_1,h_2}\left[m(y,z) \neq c_z^{(m,d)}(y)\right] \leq 8\delta/\epsilon + 8/|F| + 2\delta = \delta_1.$$

As in the proof of Lemma 64, we can now conclude that

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[m(y,z) \neq c_z^{(m,d)}(y)] \geq \epsilon\right] \leq \frac{\delta_1}{\epsilon} + \frac{1}{|F|}. \tag{22}$$

$$\text{and} \Pr_{h_1,h_2}\left[\Pr_{z}[m(0,z) \neq c_z^{(m,d)}(0)] \geq \epsilon\right] \leq \frac{\delta_1}{\epsilon} + \frac{1}{|F|}. \tag{23}$$

For the rows the conditions required are shown even more easily. We first observe that the line $l_{x+zh_1,h+zh_2}$ is a random line through F^m for any $z \neq 0$. Thus we can use the definition of δ to claim that

$$\forall y \in F, \quad \Pr_{h_1,h_2}\left[m(y,z) = f(x+zh_1+y(h+zh_2)) = P^{(f,d)}_{x+zh_1,h+zh_2}(y) = r_z^{(m,d)}(y)\right] = \delta.$$

As in the proof of Lemma 64 we can argue that

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[m(y,z) \neq r_y^{(m,d)}(z)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{24}$$

$$\text{and} \Pr_{h_1,h_2}\left[\Pr_{z}[m(0,z) \neq r_y^{(m,d)}(0)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{25}$$

Thus with probability at least $1 - 16(\delta_1/\epsilon + \delta/\epsilon + 2/|F|)$, none of the events (22)-(25) happen and we can apply Corollary 63.

To conclude we need to show that $1 - 16(\delta_1/\epsilon + \delta/\epsilon + 2/|F|) > 8\delta/\epsilon + 8/|F|$ and this follows from the conditions given is the statement of the lemma. ▮

Now we can prove the main theorem of this section.

Proof of Theorem 58: The first part of the theorem is straightforward. Part (2) can be shown as follows. Let $\epsilon = \min 1/36, \epsilon_0$, where ϵ_0 is as in Theorem 62. Now pick α to be $\max\{c, 600/\epsilon\}$, where c is as given by Theorem 62. Now let $\delta_0 = \frac{\epsilon^2}{624}$. Notice that δ_0 is positive.

For any function $f : F^m \to F$, over a field of size at least $\alpha(d+1)^3$, if $\delta_f < \delta_0$, then the conditions required for the application of Lemma 66 are true and we can conclude that $\hat{f}_d$ satisfies

$$\forall x, h \quad \hat{f}_d(x) = P^{(\hat{f}_d,d)}_{x,h}.$$

Under the condition $|F| > 2d+1$, this condition is equivalent to saying $\hat{f}_d$ is a degree d polynomial (see [84]). (See also [47] for a tight analysis of the condition under which this equivalence holds.) By Lemma 59 $\Delta(f, \hat{f}_d) \leq 2\delta_{f,d}$. Thus f is $2\delta_{f,d}$-close to a degree d polynomial. ▮