# 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. If a string is in the language, then there exists a proof such that the verifier accepts with probability 1 (i.e., for every choice of its random string). For strings 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 [6] 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 [82] 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, Lovász, Safra and Szegedy [42], and Arora and Safra [6] and shows that there exists a positive $\epsilon$ such that approximating the maximum clique size in an $N$-vertex graph to within a factor of $N^{\epsilon}$ 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, developed by Cook [36], Karp [69] and Levin [75], shows that many decision problems of interest, such as satisfiability, are *NP-complete*. This theory also shows that decision versions of many optimization problems, such as the traveling salesman problem,

*arora@cs.princeton.edu. Department of Computer Science, Princeton University, NJ 08544. 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, an Alfred P. Sloan Fellowship and a Packard 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 an Alfred P. Sloan Research Fellowship, an 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@lcs.mit.edu. Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139. Parts of this work were done when this author was at the University of California at Berkeley, supported by NSF PYI Grant CCR 8896202, and at the IBM Thomas J. Watson Research Center.

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

bin packing, graph coloring, and maximum clique are NP-complete, thus implying that those optimization problems are *NP-hard*. If P $\neq$ NP, no polynomial-time algorithm can solve them optimally.

Given this evidence of intractability, researchers have attempted to design polynomial time *approximation algorithms* for the NP-hard optimization problems [53, 81]. 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 [53, 81], starting with the papers of Johnson [66] and Sahni and Gonzales [92], 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 [91, 64], 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 [60], 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, and the metric 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. The notion of *strong NP-completeness* was introduced by Garey and Johnson [52] to show that a large collection of problems are not in FPTAS if P $\neq$ NP. Sahni and Gonzalez [92] 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, Lovász, Safra, and Szegedy [42] 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 [6] 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 [82], 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. They used second 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 [40] and Kolaitis and Vardi [73].) 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 if a *MAX SNP-hard* problem is in PTAS, then 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 and Safra, involves constructing efficient verifiers that probabilistically check membership proofs for NP languages. As a consequence, we improve upon Arora and 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 the length of the input. 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 $\pi$ of length polynomial in $|x|$ such that $M$ accepts the pair $(x, \pi)$; conversely, if $x \notin L$, then for all strings $\pi$, $M$ rejects $(x, \pi)$.

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 [59] and Babai [10, 16] 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 [77] and Shamir [94], 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 [24], 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 [46] gave an equivalent definition of MIP as languages that have a probabilistic polynomial-time oracle verifier that checks membership proofs (possibly of exponential length) using *oracle access* to the proof. This equivalent definition is described below, using the term *probabilistically checkable proofs* introduced by Arora and Safra.

Babai, Fortnow and Lund [13] recently showed that MIP is exactly NEXP, the class of languages for which membership proofs can be checked deterministically in exponential time. This result is surprising because NEXP 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 = NEXP result can be "scaled-down" to say something interesting about NP. Babai, Fortnow, Levin and Szegedy [14] introduced the notion of *transparent* membership proofs, namely, membership proofs that can be checked in polylogarithmic time, provided the input is encoded with some error-correcting code. They showed that NP languages have such proofs. Feige et al. [42] showed a similar result, but with a somewhat more efficient verifier. Arora and Safra [6] 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. [46] 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. Note that we count only the bits of the *proof* that are read - not the bits of the *input* which the verifier is allowed to read fully. These parameters were first highlighted by the work of Feige et al.[42]. In fact, the definition of a class very similar to PCP was implicit in the work of Feige et al. [42].

**Definition 1 ([6])** *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 $L \in PCP(r(n), q(n))$ 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 $\pi$ such that $V$ accepts $(x, \pi)$ with probability* 1 *(i.e., for every choice of its random bits);*
- *if $x \notin L$, then for any membership proof $\pi$, $V$ accepts $\pi$ with probability at most* $1/2$.

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

- NEXP $= \cup_{c>0}$PCP$(n^c, n^c)$ (Babai, Fortnow, and Lund [13]).
- NP $\subseteq \cup_{c>2}$PCP$(\log^c n, \log^c n,)$ (Babai, Fortnow, Levin, and Szegedy [14]).
- NP $\subseteq \cup_{c>0}$PCP$(c \log n \operatorname{loglog} n, c \log n \operatorname{loglog} n)$ (Feige, Goldwasser, Lovász, Safra, and Szegedy [42]).
- NP $= \cup_{c>0}$PCP$(c \log n, c\sqrt{\log n})$ (Arora and Safra [6]).

Notice that the first and the last of the above relations are exact characterizations of the complexity classes NEXP 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. [42] implies that in order to prove the hardness of approximating the clique number, it suffices to show that some NP-complete language is low in the PCP hierarchy. The following statement summarizes this result.

**Theorem 2 ([42])** *Suppose $q : \mathcal{Z}^+ \to \mathcal{Z}^+$ is a logarithmically bounded non-decreasing function and $c$ is a constant such that*

$$\text{3-SAT} \in PCP(c \log n, q(n)).$$

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

**Remark:** The above statement is a well-known implication of the result of Feige et al. [42]. It uses the idea of reducing the error probability of the verifier using "recycled" random bits (see [32, 65]). For further details see [6].

Thus, as a consequence of the new characterization of NP due to Arora and Safra [6], 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 [18], Bellare and Rogaway [22], Feige and Lovász[45], and Zuckerman [100]). Another such connection is reported by Arora, Motwani, Safra, Sudan and Szegedy [5], 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 ([5])** *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 and Safra [6].

**Theorem 4 (Main)** *There is a positive integer $q$ such that*

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

The containment $\cup_{c>0}\text{PCP}(c \log n, q) \subseteq \text{NP}$ is trivial, since a verifier that uses $O(\log n)$ random bits can be replaced by a deterministic verifier. The nontrivial containment $\text{NP} \subseteq \cup_{c>0}\text{PCP}(c \log n, q)$ is a consequence of our Theorem 17 below.

We note that the characterization in Theorem 4 is probably optimal up to constant factors, since, as observed in [6], if $\text{P} \neq \text{NP}$, then $\text{NP} \not\subseteq \text{PCP}(o(\log n), o(\log n))$.

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 $\text{P} \neq \text{NP}$ then no MAX SNP-hard problem is in PTAS. The class of MAX SNP-hard problems includes MAX 2SAT, MAX CUT, vertex cover [82], metric TSP [83], Steiner tree [26], shortest superstrings [27], MAX 3DM [67], and multiway cut [38]. Our theorem above implies that for every one of these problems $\Pi$, there exists a threshold $\epsilon_\Pi$ such that approximating $\Pi$ 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 $N^{\epsilon}$ is NP-hard.*

The previous best result (in [6]) 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 [14]; Feige, Goldwasser, Lovász, Safra and Szegedy [42]; and Arora-Safra [6]). 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 [30], and Rubinfeld and Sudan [90]. The influence of the former will be apparent in Section 5, while the latter work (together with a lemma of Arora and Safra [6]) is used in our analysis of a *Low Degree Test* described in Section 7.2. Finally, we were influenced by work on *constant prover 1-round interactive proof systems* [74, 45]. 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 [6]. 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 $q_1(n)$ bits in the membership proofs while checking it and the second verifier queries $q_2(n)$ 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 gates 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 trivially reformulated as a circuit, so 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 each of whose entries 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 $r, p, c, a: \mathbb{Z}^+ \to \mathbb{Z}^+$, 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 $\pi$ 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 $\pi$:** *$V$ uses $x$ and $R$ to compute $p(n)$ locations in $\pi$. Let $q_1, \ldots, q_{p(n)}$ denote these locations.*
4. **Queries the oracle:** *$V$ queries the oracle $\pi$ 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 $\pi$.*
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** *For $r, p, c, a: \mathcal{Z}^+ \to \mathcal{Z}^+$ and $e < 1$, a language $L \in$ **RPCP**$(r(n), p(n), c(n), a(n), e)$, 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 $\pi$ such that*

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

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

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

*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 [46]. Fairly efficient constructions of such verifiers are implicit in [74, 45]; for instance, it is shown that NP $\subseteq \cup_{c<\infty}$RPCP$((\log)^c n, O(1), (\log)^c n, (\log)^c n, 1/n)$. We also observe that the definition of RPCP generalizes that of PCP. In particular, RPCP$(r(n), p(n), c(n), a(n), 1/2) \subseteq$ 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 $L \in$ RPCP$(r(n), p(n), c(n), a(n), 1/2)$ for some NP-complete language $L$, 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 17). 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 poly$(\log n)$ to poly$(\log \log n)$ to $O(1)$.

First we define an inner verifier, a notion implicit in Arora and Safra [6]. 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. [14] 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 $E_n^{-1} : \mathcal{F}_{l(n),a(n)} \to \{0,1\}^n$. An encoding/decoding scheme pair $(E, E^{-1})$ 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 $\pi$ for $X_{k+1}$. The verifier runs in* $\mathrm{poly}(n)$ *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 $\pi$:** *$V$ 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 $\pi$ 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 $1 \le j \le k$.*

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

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

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

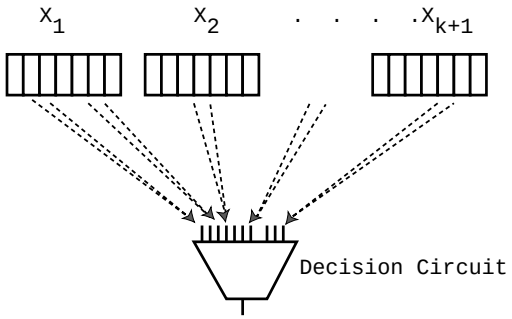


Figure 1: An inner verifier uses its random string to compute a (small) decision circuit and a sequence of queries to the oracles. Then it receives the oracles' answers to the queries. The verifier accepts iff the concatenation of the answers satisfies the decision circuit.

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 $r, p, c, a: \mathcal{Z}^+ \to \mathcal{Z}^+$ 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 (rephrasing of results in [6])** *Let $p \in \mathcal{Z}^+$ be such that a $(p, r_1(n), p_1(n), c_1(n), a_1(n), e_1)$-inner verifier system exists for some functions $r_1, p_1, c_1, a_1: \mathcal{Z}^+ \to \mathcal{Z}^+$ and $0 < e_1 < 1$. Then, for all functions $r, c, a: \mathcal{Z}^+ \to \mathcal{Z}^+$ and every 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 $\tau$ 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.*

Before proving Theorem 4 we first point out the following simple amplification rule.

**Proposition 16** *For integer $k$, real $e$ and functions $r(\cdot), p(\cdot), c(\cdot), a(\cdot)$, if a $(k, r(n), p(n), c(n), a(n), e,)$ inner verifier system exists, then for every postive integer $l$, a $(k, lr(n), lp(n), lc(n), la(n), e^l)$ inner verifier system also exists.*

Proposition 16 is proved easily by sequentially repeating the actions of the given inner verifier $l$ times and accepting only if one accepts in each iteration. As a consequence of Proposition 16 one can reduce the error of the inner verifiers given by Theorems 15 and 14 to any $e > 0$. In proving Theorem 4 we will be using this ability for $e = 1/16$. Now we prove Theorem 4, our main theorem. It is a simple consequence of the following theorem.

**Theorem 17** *There exists a constant $C$, such that for every language $L \in NP$, there exists a constant $c_L$ so that*

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

**Proof:** Since CKT-SAT is NP-complete, we are done if we can show that CKT-SAT $\in$ 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 14 and use recursive proof checking to construct more efficient verifiers. First we use Theorem 14 with $k = 1$, and reduce the error of the inner verifier to $1/16$ using Proposition 16. This 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

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

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 14. 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 (again after applying Proposition 16).

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

$$\text{CKT-SAT} \in \text{RPCP}(c_1 \log n + c' d_1 \log\log n,\ c'',\ (d_1 \log\log n)^{d'},\ (d_1 \log\log n)^{d'},\ \frac{2}{16}). \tag{2}$$

This verifier for CKT-SAT makes $c''$ queries to the oracle in the membership proof. Using the constant $c''$ as $k$ 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 (again after amplifying using Proposition 16).

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

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

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

**Remark 18** *By working through the proofs of the various theorems in this paper and earlier papers, it will be clear that the result in Theorem 17 is constructive, in the following sense. Given a satisfying assignment to a circuit, we can in polynomial time construct a "proof oracle" that will be accepted by the verifier of Theorem 17 with probability 1. Conversely, given a proof oracle that the verifier accepts with probability $\geq 1/2$, we can in polynomial time construct a satisfying assignment. The reader can check this by noticing throughout that our polynomial-based encodings/decodings are efficiently computable.*

Now we prove Theorem 13.

**Proof:** (*of Theorem 13*) First we outline the main idea. Let $L$ be a language in $\text{RPCP}(r(n), p, c(n), a(n), e)$ 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 need 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

$$Y = \text{Number of locations that } V_1 \text{ expects in a proof oracle for input } x. \qquad (4)$$

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 $\pi$. Then each address in $\pi$ 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 $\pi$ 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

$$C = \text{the circuit computed by Step 2 of } V_1. \qquad (5)$$

$$Q_1, \ldots, Q_p = \text{the queries generated by Step 3 of } V_1. \qquad (6)$$

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 $\pi$ 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 $\pi'$ which $V_1$ accepts with probability 1. We describe how to convert $\pi'$ into an oracle $\pi$ which $V$ accepts with probability 1. First, replace the string in each location of $\pi'$ with the encoding of that string using $E_{c(n)}$. (This encoding is an oracle of answer size $a_1(c(n))$.) Thus if $\pi'$ had $Y$ locations, we obtain a sequence of $Y$ oracles. We make these oracles the first $Y$ sub-oracles of $\pi$. Next, we construct the last $2^{r(n)}$ suboracles of $\pi$. 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 $\pi'$ 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 $\tau$ such that

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

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

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

In other words, $V$ accepts $\pi$ with probability 1.

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

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

We use $\pi$ to construct an oracle $\pi'$ such that

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

thus contradicting the soundness condition for $V_1$.

To construct $\pi'$, we take the first $Y$ suboracles in $\pi$, and apply $E^{-1}_{a(n)}$ on each. (Of course, if $\pi$ 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 $\pi'$ with $Y$ locations and answer size $a(n)$. Now we show that $V_1$ accepts $\pi'$ 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 $\pi$ 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 $\pi'$ 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

Here 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 19 (MAX 3-SAT)** *An instance of this problem is a* 3*-CNF formula* $\phi$*. 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 $\phi_x$ with at most $2^q 2^{c_L \log n}$ clauses. This formula satisfies the following conditions.

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

The reduction goes as follows. For every potential query $Q_i$ to the oracle $\pi$ 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 $\psi_{x,R}$ in CNF (conjunctive normal form) over the variables $v_{i_1}, \ldots, v_{i_q}$. Observe that $\psi_{x,R}$ has at most $2^q$ clauses, each of length at most $q$. Then express it in the standard way (see e.g. [53]) as a 3-CNF formula, by using up to $q$ 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 $\phi_x$ by $m$. Note that $m$ is at most $q2^q 2^{c_L \log n}$.

We now argue the completeness and soundness properties. We first show that if $x \in L$, then we can find a satisfying assignment to $\phi_x$. Let $\pi$ be an oracle such that $V$ accepts with probability 1. For every $i$, set $v_i = \pi(Q_i)$. Notice that this assignment satisfies $\psi_{x,R}$ for every $R$. Now we fill in the values of the auxiliary variables so that every formula $\phi_{x,R}$ is satisfied. Note that since the auxiliary variables are not shared by different $\phi_{x,R}$'s, there is no conflict in this assignment. Conversely, suppose $x \notin L$. Suppose for contradiction's sake that there exists an assignment to the variables $v_i$ and the auxiliary variables such that at most $\epsilon m$ clauses in $\phi_x$ are not satisfied, for $\epsilon = \frac{1}{q2^{(q+1)}}$. Construct an oracle $\pi$ such that $\pi(Q_i) = v_i$. Notice that for every $R$ such that $\phi_{x,R}$ is satisfied by the assignment, $V$ accepts $\Pi$ on random string $R$. But the number of $R$'s such that $\phi_{x,R}$ is not satisfied is at most $\epsilon m = 2^{c_L log n - 1}$. Thus the probability that $V$ accepts $\pi$ is at least $1/2$, implying $x \in L$. This is a contradiction. We conclude that no assignment can satisfy more than $(1-\epsilon)m$ clauses of $\phi_x$.

We conclude by observing that if there exists an algorithm that can distinguish between the cases that the number of satisfiable formulae in $\phi_x$ is $m$ and the case where the number of satisfiable formulae in $\phi_x$ is less than $(1-\epsilon)m$, 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 20** *Let $D$ and $R$ be finite sets. Let $f, g$ be functions from $D$ to $R$ and let $\mathcal{F}$ be a family of 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. The* relative distance between $f$ and $\mathcal{F}$, *denoted $\Delta(f,\mathcal{F})$, is the minimum over $g \in \mathcal{F}$ of $\Delta(f,g)$. If $\Delta(f,g) \leq \epsilon$, we say that $f$ is* $\epsilon$-close to $g$. *Similarly, if $\Delta(f,\mathcal{F}) \leq \epsilon$, we say that $f$ is* $\epsilon$-close to $\mathcal{F}$.

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.

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

The function families considered in this paper are families of multivariate polynomials over finite fields of "low degree". 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 family of $m$-variate polynomials of degree $d$. Specializing our definition of closeness above to such families, we see that a function $f: F^m \to F$ is *$\delta$-close to $F_m^{(d)}$* (or just $\delta$-close when $d$ is understood from the context) if there is a polynomial $g \in F_m^{(d)}$ such that $\Delta(f, g) \leq \delta$. The following lemma is often attributed to Schwartz [93].

**Lemma 21** *If $f, g \in F_m^{(d)}$ and $f \neq g$, then $\Delta(f, g) \geq 1 - d/|F|$.*

**Remark 22** *From the above lemma, it follows that the nearest polynomial $g$ is unique if $\delta < 1/4$ and $d/|F| \leq 1/2$. This fact will be used many times.*

In the following sections we will be using the space $F_m^{(d)}$ to represent and encode information. Two very different representations of this space will be used. The first is the "terse" representation. In this representation, an element of $F_m^{(d)}$, i.e., an $m$-variate polynomial of degree $d$, will be represented using $\binom{m+d}{d}$ coefficients from $F$. In particular, in this representation an element of $F_1^{(d)}$ can be specified using at most $(d+1)\lceil \log |F| \rceil$ bits. The other representation is the "redundant" representation. In this we first map $F_m^{(d)}$ to $F^m \to F$, i.e., to the space of all functions on $m$ variables from $F$ to $F$, using the natural map — every polynomial in $m$ variables is a function from $F^m \to F$. We then represent elements of $F^m \to F$ as $|F|^m$ elements from $F$, i.e., by a listing of values of the function on all possible inputs. For all choices of $m$ and $d$ that will be used in this paper, the "redundant" representation is significantly longer that the "terse" representation. However this redundancy renders the representation "robust". Lemma 21 above tells that two polynomials represented this way differ on most indices (provided $d/|F|$ is small). The "terse" representation will be the default representation for elements of $F_m^{(d)}$. We will specify explicitly whenever we use the "redundant" representation.

# 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 $\mathrm{GF}(2)$ and all operations (additions and multiplications) are assumed to be over $\mathrm{GF}(2)$. We will use an encoding/decoding scheme that encodes bit-strings with linear functions over $\mathrm{GF}(2)$. 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 $\epsilon$-close to the encoding of a satisfying assignment).

Think of $n$-bit strings as $n$-dimensional vectors over $\mathrm{GF}(2)$. For an $n$-bit string $x$, we let $x^{(i)}$ represent its $i$-th 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 $\mathrm{GF}(2)$ is the encoding of some $n$ bit string.

**Definition 23 (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 24** *For $x \neq y$, $\Delta(E_n^{\oplus}(x), E_n^{\oplus}(y)) = 1/2$.*

We now define the corresponding decoding scheme.

**Definition 25 (parity decoding)** *Given a function $f : GF(2)^n \to GF(2)$, 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 26** *From Remark 22 it follows that 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 $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$, 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 [30] 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 27 ([30])**

1. *If $f$ is a linear function, 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 27 may not match the bound as stated in [30]. However this bound can be easily reconstructed from some of the subsequent papers, for instance, in the work of Bellare et al. [19]. In any case, the exact bound is unimportant for what follows.

A 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 [30].

```
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 28 ([30])**

1. *If $f$ is a linear function, then the procedure* Linear-self-corr$(f, x; n)$ *outputs $f(x)$ with probability* 1.
2. *Given a function $f$ that is $\delta$-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 $1 - 2\delta$.*

**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 29** *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:*

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

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 $f_1, \ldots, f_k$, where $f_i : \mathrm{GF}(2)^{n_i} \to \mathrm{GF}(2)$ is defined as

$$f_i(x_i) = f(0, \ldots, 0, x_i, 0, \ldots, 0) \qquad \forall x_i \in \mathrm{GF}(2)^{n_i}, \tag{7}$$

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 $f_i$ as defined in (7) is equal to some given linear function $f' : \mathrm{GF}(2)^{n_i} \to \mathrm{GF}(2)$. A randomized test suggests itself: pick $x_i \in \mathrm{GF}(2)^{n_i}$ at random and test if $f(0, \ldots, 0, x_i, 0, \ldots, 0) = f'(x_i)$. We now show that a simple variant of this test works even if $f$ and $f'$ are not linear functions but only $\epsilon$-close to some linear functions.

First we introduce some notation: For $i \in \{1, \ldots, k\}$ and $x_i \in \mathrm{GF}(2)^{n_i}$, the inverse projection of $x_i$ is the vector $y = \pi_i^{-1}(x_i; n_1, n_2, \ldots, n_k) \in \mathrm{GF}(2)^{n_1+\cdots+n_k}$ 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, ..., n_k):
/* Expects oracles f : GF(2)^{n_1+...+n_k} → GF(2)
        and f' : GF(2)^{n_i} → GF(2).   */

   Pick x_i ∈_R GF(2)^{n_i};
   If Linear-self-corr(f, π_i^{-1}(x_i; n_1, ..., n_k); Σ_{i=1}^k n_i) = f'(x_i)
        then accept else reject.
```

**Proposition 30**

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 $i$-th 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 $\epsilon$-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 $i$-th function being $g'$.*

**Proof:** The proof of the completeness part is straightforward. For the second part, we use the fact that since $g$ is linear it is the concatenation of linear functions $g_1, \ldots, g_k$, where $g_j(\cdot) = g(\pi_j^{-1}(\cdot; n_1, \ldots, n_k))$ for $j = 1, \ldots, k$. Assume for contradiction that $g_i(\cdot) \neq g'(\cdot)$. Then since both $g_i$ and $g'$ are linear, 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 $2\epsilon$ (by Proposition 28) and the final event with probability at most $\epsilon$ (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 31** *For $x \in GF(2)^n$, the function* $\mathsf{quad}_x$ *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 $x$, i.e., any $n$-variate polynomial in the variables $x^{(1)}, \ldots, x^{(n)}$ of degree at most two, can be computed from the value of $\mathsf{quad}_x$ 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 with $p^{(00)} = 0$, we use the notation $\mathsf{coeff}_p$ 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 32** *For any $x \in GF(2)^n$, the function $\mathsf{quad}_x$ 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 $x, y \in \mathrm{GF}(2)^n$, the *outer product* of $x$ and $y$, denoted $x@y$, is the $n^2$ bit vector with $(x@y)^{(ij)} = x^{(i)} y^{(j)}$.

```
Quadratic-consistency(f, f'; n);
/* Expects f : GF(2)^n → GF(2) and f' : GF(2)^{n^2} → GF(2).  */

   Pick x, y ∈_R GF(2)^n.
   If f(x) · f(y) = f'(x@y) then accept else reject.
```

The test above is based on Freivalds test [47] for matrix multiplication and its correctness is established as follows.

**Proposition 33**

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 [47] 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

$$\Pr_{z_1 \in \mathrm{GF}(2)^n} \left[ z_1^T B z_2 \neq z_1^T C z_2 | B z_2 \neq C z_2 \right] \geq \frac{1}{2}.$$

By taking the product of the two events above we get

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

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 $\mathsf{quad}_x$, 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 z_1, z_2 ∈_R GF(2)^n;
   If f(z_1) · f(z_2) ≠ Linear-self-corr(f', z_1@z_2; n^2)
         then reject
         else accept.
```

The following proposition can be proved as an elementary combination of Propositions 33 and 28.

**Proposition 34**

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-corr-test$(f, f'; n)$ *accepts with probability* 1.
2. *Given functions $f : GF(2)^n \to GF(2)$, which is $\epsilon$-close to some linear function $g$, and $f' : GF(2)^{n^2} \to GF(2)$, which is $\epsilon$-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 gate 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 and that the value on the output gate is accepting. This forms the basis for our next two definitions.

For strings $x_1, \ldots, x_k$ of length $m$, the *$C$-augmented* representation of $x_1 x_2 \ldots x_k$, denoted $C$-aug$(x_1 \ldots x_k)$ is an $n$ bit string $z$ indexed by the gates of $C$, where the $i$th coordinate of $z$ represents the value on the $i$-th gate of the circuit $C$, on input $x_1 \ldots x_k$, with the property that the first $km$ gates correspond to the input gates (i.e., $x_1 \ldots x_k$ is a prefix of $C$-aug$(x_1 \ldots x_k)$). Given an $n$-bit string $z$, let $\pi_i(z)$ be the *projection*

| Oracle | Contents | What contents mean to $V^{\oplus}$ |
|---|---|---|
| $X_1$<br>$X_2$<br>...<br>$X_k$ | Each $X_i$ is a function from $\mathrm{GF}(2)^m$ to $\mathrm{GF}(2)$ | Each $X_i$ is a linear function and encodes a bit string using the encoding $E^{\oplus}_m$. The concatenation of these bit strings is a satisfying assignment $s$. |
| $X_{k+1}$:<br>(has 2 suboracles) | | |
| A | $A : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ | $A = E^{\oplus}(z)$, where $z$ is the $C$-augmented representation of the satisfying assignment $s$. |
| B | $B : \mathrm{GF}(2)^{n^2} \to \mathrm{GF}(2)$ | $B = \mathsf{quad}_z$ |

Figure 2: Different oracles in the proof and what $V^{\oplus}$ expects in them

of $z$ on to the coordinates $(i-1)m+1$ to $im$. Notice that $\pi_i$ is defined so that $\pi_i(C\text{-aug}(x_1,\ldots,x_k)) = x_i$.

Given a circuit $C$ of size $n$ with $km$ inputs, we associate with the circuit $n-km+1$ polynomials $P_j$, $1 \le j \le n-km+1$. 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 $1 \le j \le n-km$, if the $j$-th gate of $C$ is an addition gate with inputs being gates $j_1$ and $j_2$ and output being the gate $j_3$, then $P_j(z) = z^{(j_1)} + z^{(j_1)} - z^{(j_3)}$. Similarly if the $j$-th gate is 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 gate is accepting.

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

$$\exists z \in \mathrm{GF}(2)^n \text{ s.t. } \forall j \in [n-km+1],\ P_j(z) = 0 \text{ and } \forall i \in [k],\ \pi_i(z) = x_i. \qquad (8)$$

However, the verifier will be given not $x_1,\ldots,x_k$ but merely functions that are purported to be $L_{x_1},\ldots,L_{x_k}$ and some extra functions that will be useful in the above checking. The intuition behind the use of the polynomials $P_j$ is that since all these polynomials are quadratic polynmials in $z$, the process of "evaluating" them may hopefully reduce to making a few queries to oracles that purport to be $L_{x_1},\ldots,L_{x_k}$ and $\mathsf{quad}_z$ for $z = C\text{-aug}(x_1,\ldots,x_k)$. In what follows we define a verifier $V^{\oplus}$ that validates this hope. 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 (8), it uses the procedures described in Section 5.

**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. For notational clarity we think of the oracle $X_{k+1}$ as consisting of two suboracles. Thus it 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 $X_{k+1}(1,\cdot)$. The contents of these oracles are described in Figure 2. The verifier performs the following actions.

1. Linearity tests:
   (a) $\mathsf{Linearity\text{-}test}(A; n)$.
   (b) $\mathsf{Linearity\text{-}test}(B; n^2)$.

(c) For $i = 1$ to $k$ Linearity-test$(X_i; m)$.

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

3. *Quadratic test:* Quadratic-corr-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 $P(z) = \sum_{j=1}^{n-km+1} r^{(j)} P_j(z)$.

   (c) If Linear-self-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 35**

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 36** *For a circuit $C$ of size $n$ with $km$ inputs, let the $m$ bit strings $x_1, \ldots, x_k$ satisfy $C(x_1 \ldots x_k) =$ 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 for every $R \in \{0,1\}^r$ $(V^{\oplus})^{X_1,\ldots,X_{k+1}}(C; R) =$ accept (where $r = 2km + n(k+5) + 4n^2$).*

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

**Proof:** Let $\delta$ be chosen so as to minimize $e = \max\{1-\delta, 1/2+6\delta, 3/4+8\delta\}$, subject to the condition $\delta < 2/9$. (This happens at $\delta = 1/36$.) We will show that this value of $e$ suffices to prove the assertion.

Let $X_1, \ldots, X_k$ and $X_{k+1} = (A, B)$ be oracles such that $V^{\oplus}$ accepts $C$ with probability greater than $e$. Then by the fact that the linearity tests (Step 1) accept with probability greater than $e \geq 1-\delta$ and by Theorem 27 we find that there exist strings $x_1, \ldots, x_k$, $z$ and $z'$ such that the oracles $X_i$ are $2\delta$ close to the functions $E^{\oplus}_m(x_i)$ respectively; and the oracles $A$ and $B$ are $2\delta$ close to $L_z$ and $L_{z'}$ respectively.

Based on the fact that the quadratic consistency test (Step 3) accepts with probability greater than $3/4 + 8\delta$, and the soundness condition of Proposition 33, we find that $L_{z'} = \mathsf{quad}_z$ and thus $B$ is $2\delta$-close to $\mathsf{quad}_z$.

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) is high to show that $z = C\text{-aug}(z_1 \ldots z_k)$ and that $C$ accepts on input $z_1 \ldots z_k$.

**Claim 38** *If $V^{\oplus}$ accepts in Step 4(c) with probability more than $1/2 + 4\delta$, then $z = C\text{-aug}(z_1, \ldots, z_k)$ and $C(z_1, \ldots, z_k) =$ accept.*

**Proof:** Assume for contradiction that either $z \neq C\text{-aug}(z_1 \ldots z_k)$ or $C$ does not accept on input $z_1 \ldots z_k$. Then it must be the case that there exists an index $j \in \{1, \ldots, n - mk + 1\}$ such that the polynomial $P_j(z) \neq 0$. In such a 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 consider the event that the verifier $V^{\oplus}$ accepts in Step 4(c). For this event to occur, at least one of the following events must also occur:

- $P(z) = 0$: The probability of this event is at most $1/2$, as argued above.
- $\mathsf{quad}_z(\mathsf{coeff}_P) \neq \mathsf{Linear\text{-}self\text{-}corr}(B, \mathsf{coeff}_P; n^2)$: Since $B$ is $2\delta$ close to $\mathsf{quad}_z$ this event happens with probability at most $\delta$, by Proposition 28.

Thus the probability that the verifier will accept in Step 4(c) is at most $1/2 + 4\delta$. ∎

Finally we use the fact that concatenation tests (Step 2) accept with high probability to claim that $z_i$ must equal $x_i$ for every $i$. Recall that for every $i \in [k]$ the concatenation test with oracle access to $A$ and $X_i$ accepts with probability at least $1/2 + 6\delta$. Furthermore, $A$ and $X_i$ are $2\delta$ close to linear functions $L_z$ and $L_{x_i}$ respectively. By applying the soundness guaranee in Proposition 30, we find that $L_z$ is the concatenation of functions $f_1, \ldots, f_{k+1}$ where $f_i(\cdot) = L_{x_i} = E^{\oplus}(x_i)$. Combining these conditions for $i = 1$ to $k$ we find that $L_z$ is the concatenation of of $E^{\oplus}(x_1), \ldots, E^{\oplus}(x_k), f$ for some linear function $f : \mathrm{GF}(2)^{n-km} \to \mathrm{GF}(2)$. This implies that the prefix of imply $z_i = x_i$ for all $i$.

Thus we conclude that if $V^{\oplus}$ accepts $C$ with probability greater than $e$ given oracle access to $X_1, \ldots, x_{k+1}$, then $C$ accepts on input $(E^{\oplus}_m)^{-1}(X_1) \ldots (E^{\oplus}_m)^{-1}(X_k)$. ∎

**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 37. Then, by Proposition 35 and Lemmas 36 and 37, it is clear that $(V^{\oplus}, E^{\oplus}, (E^{\oplus})^{-1})$ forms a $(k, c_1 \log n, p, 2^p, 1, e)$ 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 [6], 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 $(\log\log n)^{O(1)}$ but we don't need that stronger result. In fact, even the weaker verifier of Babai, Fortnow, Levin and Szegedy [14] would suffice for our purpose after some modification. This modification would cut down the number of random bits needed in [14] by using the idea of recycling randomness [32, 65].)

The following theorem was only implicit in the earlier versions of [6] but is explicit in the final version.

**Theorem 39 ([6])** *For every 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* [13] 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 40** *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 40 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 41** *Let $H \subseteq F$ and $h = |H| - 1$. For each function $s: H^m \to F$, 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 21 implies that the distance between $\hat{s_1}$ and $\hat{s_2}$ is at least 0.75.

Proposition 41 defines a mapping from $F^{(h+1)^m}$ to $F_m^{(mh)}$. Since $0, 1 \in F$, we can so use this mapping to define a map from bit strings in $\{0,1\}^{(h+1)^m}$ to polynomials in $F_m^{(mh)}$.

This map will be called a *polynomial encoding* in the rest of the paper. If the bit string has length $n < (h+1)^m$, we first extend it to a bit string of length $(h+1)^m$ in some canonical way.

**Definition 42** *For $n, m, H, F$ satisfying $|H|^m \geq n$ and $|F| \geq 2mh$, the $(n, m, H, F)$* polynomial encoding *$\mathcal{P}_{n,m,H,F}$ maps $n$-bit strings to functions from $F^m$ to $F$ that was described in the previous paragraph.*

*The* polynomial decoding *function $\mathcal{P}^{-1}_{n,m,H,F}$ maps $\{F^m \to F\}$ to $n$-bit strings and is defined as follows. Given any $r: F^m \to F$, 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 for every bit string $s \in \{0,1\}^n$, the encoding/decoding pair defined above satisfy $\mathcal{P}^{-1}_{n,m,H,F} \circ \mathcal{P}_{n,m,H,F}(s) = s$,

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, which is essentially $|F^m|$ is still $\text{poly}(n)$. Such a choice requires $m \leq O(\frac{\log n}{\log\log n})$ and $h = \Omega(\log^{(O(1)} n)$. Our parameters will be chosen subject to these conditions.

## 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 $\delta$-close to some degree $d$ polynomial, where $\delta$ is some small constant. Low degree tests were invented as part of work on proof checking [13, 14, 42, 6, 56, 90]. 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 [90] 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 [6] 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 from [90]. Our only new contribution is to use a result from [6] (Theorem 69 in the appendix) instead of a weaker result from [90].

The second procedure in this section does the following: Given an oracle $O : F^m \to F$ which is $\delta$-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 43** *Given $x, h \in F^m$, let $l_{x,h} : F \to F^m$ denote the function $l_{x,h}(t) = x + th$. With some abuse of notation, we also let $l_{x,h}$ denote the set $\{l_{x,h}(t) | t \in F\}$ and call it the* **line through $x$ with slope $h$.**

**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 $c \in F - \{0\}$.

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 $l_{x,h}$, i.e., $g(t) = p(l_{x,h}(t))$. 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 $|F|$ is sufficiently large: 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, for example, [90] or [49] — the latter contains a tight analysis of 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 $\delta$-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 $B : F^{2m} \to F_1^{(d)}$ where, for each pair $(x,h) \in F^{2m}$, $B(x,h)$ is a univariate polynomial of degree $d$. In what follows we use the notation $B(x,h)[t]$, for $t \in F$, to denote the evaluation of this polynomial at $t$. (It may be useful to the reader to think of $B$ as a function from $\mathcal{L}$ to $F_1^{(d)}$, where $\mathcal{L}$ is the set of all lines in $F^m$.) The oracle $B$ is motivated by the following alternate representation of a degree $d$ polynomial.

**Definition 44** *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 $p'(t) = p(l_{x,h}(t))$.*

We now describe our test.

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

   Pick x, h ∈_R F^m and t ∈_R F;
   Let q(·) = B(x, h).   /* q(·) ∈ F_1^{(d)}.   */
   If f(x + th) ≠ q(t) then reject else accept.
```

The properties of the test are summarized below.

**Theorem 45 (follows by combining [90, 6])** *There exist constants $\delta_0 > 0$ and $\alpha < \infty$ such that for $\delta \le \delta_0$, $d \in \mathcal{Z}^+$ 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* $] \ge 1 - \delta$, *then there exists a polynomial $p \in F_m^{(d)}$ such that $\Delta(f, p) \le 2\delta$.*

We note that Part 1 is trivial from the comments made above. The nontrivial part, Part 2, is proved in Appendix (Section A).

### 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 [17] and Lipton [76]. The specific analysis given below is borrowed from the work of Gemmell, Lipton, Rubinfeld, Sudan and Wigderson [56] 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};
   Let q(·) = B(x, h)  /* q(·) ∈ F_1^{(d)}  */
   If q[t] ≠ A(x + th) then reject else output B(x, h)[0].
```

**Proposition 46**

1. *For a polynomial $p \in F_m^{(d)}$ and its associated lines representation* $\mathsf{lines}_p$*, the output of the procedure* Poly-self-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 $p \in F_m^{(d)}$ such that $\Delta(A, p) = \epsilon$, then $\forall x \in F^m$, $\forall B : F^{2m} \to F_1^{(d)}$,*

$$\Pr_{h,t}\left[\text{Poly-self-corr}(A, B, x; m, d, F) \neq p(x) \text{ or reject}\right] \leq 2\sqrt{\epsilon} + \frac{d}{|F| - 1}.$$

**Proof:** The proof of the first part is straightforward. We now prove the second part. For any $x \in F^m$ and $t \in F - \{0\}$, notice that the random variable $x + th$ is distributed uniformly in $F^m$, if $h$ is picked uniformly from $F^m$. Thus since $\Delta(A.p) = \epsilon$, we have

$$\Pr_{h \in F^m, t \in F - \{0\}}\left[p(x + th) \neq A(x + th)\right] = \epsilon.$$

For $x, h \in F^m$ call the line $l_{x,h}$ *bad* if

$$\Pr_{t \in F - \{0\}}\left[p(x + th) \neq A(x + th)\right] > \sqrt{\epsilon}.$$

By Markov's inequality, we have

$$\Pr_{h \in F^m}\left[l_{x,h} \text{ is bad }\right] < \frac{\epsilon}{\sqrt{\epsilon}} = \sqrt{\epsilon}.$$

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 $\sqrt{\epsilon}$, taken over random choices of $h$. 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 fix $x$ and $h$ and consider the random choice of $t$. We have the following subcases:

Case: $B(x,h) = p|_{l_{x,h}}$: In this case the output is either $B(x,h)[0] = p(x)$ 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 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 necessarily $p(x,h)$.

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 $d/(|F|-1)$ 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 $2\sqrt{\epsilon} + \frac{d}{|F|-1}$. ∎

## 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 concatenation of the information in $X_1, \ldots, X_k$.

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

**Definition 47** *For $k \leq |F|$, 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 $F_{m+1}^{(d+k-1)}$ that, for every $i \in \{1,\ldots,k\}$ and $x \in F^m$, satisfies* $\mathsf{concat}_{p_1,\ldots,p_k}(x, \zeta_i) = p_i(x)$.

**Remark:** (i) Notice that $\mathsf{concat}_{p_1,\ldots,p_k}$ does exist, one such polynomial can be obtained by an interpolation on $p_1, \ldots, p_k$ using Proposition 40. This interpolation maintains the individual degrees in the first $m$ variables and leads to degree at most $k-1$ in the variable $x^{(m+1)}$). However, such a polynomial may not be unique. We assume that in such a case some such polynomial is picked arbitrarily. (ii) To understand the usefulness of this definition, see Lemma 50 below.

Next we describe a procedure to test if a given polynomial $p$ is indeed the concatenation of $k$ polynomials with the $i$-th one being $q$. In what follows, we switch notation and assume that the concatenated polynomial has degree $d$ and the polynomials being concatenated have degree $d-k+1$.

```
Poly-concat-test(t, p, q; m, 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 48** *For positive integers $d, k$, polynomials $p_1, \ldots, p_k, q \in F_m^{(d-k+1)}$, let $p = \mathsf{concat}_{p_1,\ldots,p_k} \in F_m^{(d)}$. If $p_i = q$, then* Poly-concat-test$(\zeta_i, p, q; m, F)$ *accepts with probability* 1*, else* Poly-concat-test$(\zeta_i, p, q; m, F)$ *accepts with probability at most $d/|F|$.*

**Proof:** Both parts are straightforward and follow from Proposition 21. ▮

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^{2(m+1)} → 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 49** *Let $A : F^{m+1} \to F$ be an oracle which is $\epsilon$-close to a polynomial $p \in F_{m+1}^{(d)}$ for some $\epsilon$. For any index $i \in \{1, \ldots, k\}$, let $p_i \in F_m^{(d)}$ be the polynomial given by $p_i(\cdot) = p(\cdot, \zeta_i)$. Given oracles $B : F^{2(m+1)} \to F_1^{(d)}$ and $C : F^m \to F$ the procedure* Poly-corr-concat-test$(\zeta_i, A, B, C; m, d, F)$ *behaves as follows:*

1. *If $A = p$, $B =$ 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 $\rho$, then $\Delta(C, p_i) < 1 - \rho + 2\sqrt{\epsilon} + \frac{d}{|F|-1}$.*

**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 is necessary that Poly-self-corr does not reject. Furthermore, at least one of the following events must occur: (1) Poly-self-corr$(A, B, (x, \zeta_i); m + 1, d, F)$ does not output $p(x, \zeta_i)$; or (2) $p(x, \zeta_i) = C(x)$.

Now let $\delta = \Delta(C, p_i)$. Then

$$\Pr_{x \in F^m}[p(x, \zeta_i) = C(x)] \leq 1 - \delta.$$

From Proposition 46, we have

$$\Pr_{x \in F^m}\left[\text{Poly-self-corr}(A, B, (x, \zeta_i); m + 1, d, F) \neq p(x, \zeta_i) \text{ or reject } \right] \leq 2\sqrt{\epsilon} + \frac{d}{|F| - 1}.$$

Thus the probability that the procedure Poly-corr-concat-test accepts is at most $1 - \delta + 2\sqrt{\epsilon} + \frac{d}{|F|-1}$. The lemma follows by substituting $\rho = 1 - \delta + 2\sqrt{\epsilon} + \frac{d}{|F|-1}$. ▮

To finish up, we mention an important property of concatenations of polynomials. Roughly speaking if $s_1, \ldots, s_k$ are any bit strings and $s_1 \circ s_2 \circ \cdots \circ s_k$ denotes their concatenation as strings, then the (polynomial) concatenation of their polynomial encodings is simply the polynomial encoding of $s_1 \circ s_2 \circ \cdots \circ s_k$. This property will be used in the proof of Lemma 60 later.

**Lemma 50 (A structural property of polynomial concatenation)** *Let $H \subseteq F$ be such that $|H| \geq k$. Let $n, m$ be such that $n = |H|^m$. Suppose $s_1, \ldots, s_k \in \{0,1\}^n$ are any bitstrings and $p_1, \ldots, p_k : F^m \to F$ are their polynomial encodings, that is, $\mathcal{P}_{n,m,H,F}(s_i) = p_i$. Then the polynomial encodings satisfy the following property*

$$\mathcal{P}^{-1}_{n,m+1,H,F}(\mathsf{concat}_{p_1,\ldots,p_k}) = s_1 \circ s_2 \circ \cdots s_k \circ T, \tag{9}$$

*where $T$ is some bitstring of length $n(|H| - k)$.*

**Proof:** Follows trivially from the definition of polynomial encoding.

## 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 $x_1, \ldots, x_l \in F^m$. The procedure Curve-check described in this section allows us to find all the values by making only $O(1)$ queries to the oracle for $p$ and some auxiliary oracle $O$. We emphasize that the number of queries does not depend upon $l$, the number of points for which we desire the value of $p$. Intuitively speaking, the procedure Curve-check allows us to "aggregate" queries.

We first introduce the notation of a curve.

**Definition 51** *A* **curve** *through the vector space $F^m$ is a function $C : F \to F^m$, i.e., $C$ takes a parameter $t$ and returns a point $C(t) \in F^m$. 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 $C$ is a* **curve of degree** *$d$.*

**Remark:** As in the case of lines, it will be convenient to think of a curve $C$ as really the set $C = \{C(t)\}$. Thus curves really form a generalization of lines, with lines being curves of degree 1.

The following lemma will be useful in our next definition.

**Lemma 52** *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 $C$ such that $C(t_i) = x_i$. Furthermore, $C$ can be constructed using $\mathrm{poly}(l, m)$ field operations and in particular given $t \in F$, the point $C(t)$ can be computed using $\mathrm{poly}(l, m)$ field operations.*

**Proof:** Follows from Proposition 40 and the fact that polynomial interpolation and evaluation can be performed in polynomial time. ∎

For the next definition, recall that $\zeta_1, \ldots, \zeta_{|F|}$ is a fixed enumeration of the elements of $F$.

**Definition 53** *For $l \leq |F|$, 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 $C(\zeta_i) = x_i$ for $i \in \{1, \ldots, l\}$.*

**Remark 54** *By Lemma 52 we know that such a curve exists and can be constructed and evaluated efficiently.*

We will need one more fact before we can go on. For a function $f : F^m \to F$ and a curve $C : F \to F^m$, we let $p|_C : F \to F$ denote the function $p|_C(t) = p(C(t))$.

**Proposition 55** *Given a curve $C : F \to F^m$ of degree $l$ and a polynomial $p \in F_m^{(d)}$, the function $p|_C(t)$ is a univariate polynomial in $t$ of degree at most $(l-1)d$.*

The basic idea that allows us to aggregate queries 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 the univariate polynomial $p|_C$, where $C = \mathsf{curve}_{x_1,\ldots,x_l}$.

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

   Let q(·) = O(x_1, ..., x_l).
   Pick t ∈_R F.
   If q(t) ≠ p(curve_{x_1,...,x_l})[t]
      then reject
      else output q(ζ_1), ..., q(ζ_l).
```

**Proposition 56** *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 $\frac{d(l-1)}{|F|}$.*

The proof of the above proposition is a straightforward consequence of the fact that two distinct degree $d(l-1)$ univariate polynomials can agree in at most $\frac{d(l-1)}{|F|}$ 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. Note that this algorithm needs access to an oracle $B$ that contains, for every line in $F^m$, a univariate degree $d$ polynomial.

```
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))}, and x_1, ..., x_l ∈ F^m.  */

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

**Remark:** Notice that Curve-corr-check only looks at one location in each of its three oracles.

The following Lemma proves the correctness of this procedure.

**Lemma 57** *Let $A : F^m \to F$ be an oracle which is $\epsilon$-close to some polynomial $p \in F_m^{(d)}$ for $\epsilon < .01$. Given oracles $B : F^m \to F$ and oracle $O$, and points $x_1, \ldots, x_l \in F^m$, 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 $2\sqrt{\epsilon} + \frac{d}{|F|-1} + \frac{dl-1}{|F|}$.*

Lemma 57 follows by a straightforward combination of Propositions 46 and 56, since Proposition 46 says that the result of $\mathsf{Poly\text{-}self\text{-}corr}(A, B, \mathsf{curve}_{x_1,\ldots,x_l}(t); m, d, F)$ is $p(x)$ with high probability.

## 7.5 Parallelization: Putting it together

In this section we prove Theorem 14.

Let $k \in \mathcal{Z}^+$ be any constant. Let $V^{\mathsf{seq}}$ be a $(k, c_1 \log n, \log^{c_2} n, \log^{c_3} n, 1, e)$ inner verifier as guaranteed to exist by Theorem 39. We construct a new inner verifier $V^{\mathsf{par}}$ with the desired parameters.

Let $C$ be a circuit of size $n$ and $km$ input nodes. Given such a circuit as input, the inner verifier $V^{\mathsf{par}}$ expects the proof to contain $k+1$ oracles $X_1, X_2, \ldots, X_{k+1}$. These oracles contain some information which $V^{\mathsf{par}}$ can examine probabilistically (by using $O(k)$ queries). This information is meant to convince $V^{\mathsf{par}}$ that there *exist* some other $k+1$ oracles $Y_1, \ldots, Y_{k+1}$ which make $V^{\mathsf{seq}}$ accept input $C$ with high probability (and consequently, that $C$ is a satisfiable circuit).

The main idea is that $X_1, \ldots, X_k, X_{k+1}$ each contains a polynomial that (supposedly) encodes a bit string. These $k+1$ bit strings (supposedly) represent a sequence of $k+1$ oracles that would have been accepted by $V^{\mathsf{seq}}$ with probability 1. Given such oracles $V^{\mathsf{par}}$ first performs low degree tests (using only $O(k)$ queries to the proof; see Section 7.2.1) to check that the provided functions are close to polynomials. Next, $V^{\mathsf{par}}$ tries to simulate the actions of $V^{\mathsf{seq}}$ on the preimages of the polynomials (i.e., the strings encoded by the polynomials). The simulation requires reconstructing $\mathrm{poly}(\log n)$ values of the polynomials, which can be done using Procedure Curve-corr-check of Section 7.2. Note that this step requires only $O(k)$ queries to the proof, *even though it is reconstructing as many as* $\mathrm{poly}(\log n)$ *values of the polynomials.*

Now we formalize the above description. First we recall how $V^{\mathsf{seq}}$ acts when given $C$ as input. $V^{\mathsf{seq}}$ expects the proof to contain $k+1$ oracles $Y_1, Y_2, \ldots, Y_{k+1}$. Let

$$N = \text{Maximum number of bits in any of } Y_1, \ldots, Y_{k+1} \quad (10)$$

$$r = \text{number of random bits used by } V^{\mathsf{seq}} \text{ on input } C \quad (11)$$

$$(E^{\mathsf{seq}}, (E^{\mathsf{seq}})^{-1}) = \text{encoding/decoding scheme of } V^{\mathsf{seq}} \quad (12)$$

On random string $R \in \{0,1\}^r$, let the queries of $V^{\mathsf{seq}}$ 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 $i$-th oracle $Y_i$ is being queried with question $q$. By Theorem 39, the number of queries $l \leq \log^{c_3} n$ and the maximum size of an oracle $N \leq n^{c_1} l$ (this is because the verifier uses only $c_1 \log n$ random bits, each of which results in $\leq l$ queries to the oracles; each query results in an answer of 1 bit).

| Oracle | Contents | What contents mean to $V^{\mathsf{par}}$ |
|---|---|---|
| $X_1$<br>$X_2$<br>...<br>$X_k$ | Each $X_i$ is a function from $F^w$ to $F$ | Each $X_i$ is a polynomial encoding of a bit string; the $i$th bitstring is the $i$th oracle for $V^{\mathsf{seq}}$. |
| $X_{k+1}$:<br>(has 4 suboracles)<br>Z | $Z: F^w \to F$ | Polynomial that encodes the $k+1$th oracle needed by $V^{\mathsf{seq}}$ |
| A | $A: F^{w+1} \to F$ | Encodes the concatenation of $X_1, \ldots, X_k$ and $Z$ as defined in Definition 47. |
| B | $B: F^{2(w+1)} \to F_1^{(d)}$. Contains, for each line in $F^{w+1}$, a univariate degree $d$ polynomial. | Allows low degree test to be performed on $A$ |
| O | $O: F^{(w+1)l} \to F_1^{(d(l-1))}$ Contains, for each degree $l$ curve in $F^{(w+1)l}$, a univariate degree $d(l-1)$ polynomial | Allows the procedure Curve-corr-check to reconstruct up to $l$ values of the polynomial closest to $A$. |

Figure 3: Different oracles in the proof and what $V^{\mathsf{par}}$ expects in them

The verifier $V^{\mathsf{par}}$ fixes $w = \frac{\log N}{\log\log N}$, $F$ to be a finite field of size $\Theta(\log^{\max\{2+c_3,6\}} N)$, and picks a subset $H$ of cardinality $\log N$ (arbitrarily). Let $d = \frac{\log^2 N}{\log\log N} + k - 1$. Notice that under this choice of $H$ and $w$, we have $|H|^w \geq N$. The encoding scheme used by the parallel verifier system is $E^{\mathsf{par}} = \mathcal{P}_{N,w,H,F} \circ E^{\mathsf{seq}}$, where $\mathcal{P}_{N,w,H,F}$ is the polynomial encoding defined in Section 7.1. (In other words, computing $E^{\mathsf{par}}(s)$ involves first computing $E^{\mathsf{seq}}(s)$, and then using $\mathcal{P}_{N,w,H,F}$ to encode it using a polynomial.) The decoding scheme is $(E^{\mathsf{par}})^{-1} = (E^{\mathsf{seq}})^{-1} \circ (\mathcal{P}_{N,w,H,F})^{-1}$. Recall that the encoding $\mathcal{P}_{N,w,H,F}$ used a canonical mapping from $[N]$ to $H^w$. We need to refer to this mapping while describing the actions of the verifier $V^{\mathsf{par}}$, so we give it the name #. Thus the image of $q \in [N]$ is $\#(q) \in H^w$.

Of the $k+1$ oracles in the proof, the verifier $V^{\mathsf{par}}$ views the last oracle $X_{k+1}$ as consisting of four suboracles, denoted by $X_{k+1}(1,\cdot), X_{k+1}(2,\cdot), X_{k+1}(3,\cdot)$ and $X_{k+1}(4,\cdot)$ respectively. Let us use the shorthand $Z$, $A$, $B$, $O$ respectively for these suboracles. Figure 7.5 describes the contents of all the oracles.

Notice that the oracle $O$ as described in Figure 7.5 above has $|F|^{(w+1)l}$ entries, which for our choices of $l, w$ is superpolynomial in $n$. However, we shall see below —in Step 3(d)— that the verifier will need only $2^r = \text{poly}(n)$ entries from this oracle (and so the rest of the entries need not be present). Specifically, step 3(d) requires an entry in $O$ only for the curve that passes through the points $z_1, \ldots, z_l$ mentioned in that step. The tuple of points $z_1, \ldots, z_l$ is generated using a random string $R \in \{0,1\}^r$, so there are only $2^r$ such tuples that the verifier can generate in all its runs.

Now we describe the verifier $V^{\mathsf{par}}$. (After each step we describe its intuitive meaning in parentheses.) Recall that $\zeta_1, \ldots \zeta_{|F|}$ are the elements of field $F$.

1. Run Poly-test$(A, B; w+1, d, F)$. ( If this test accepts, the verifier can be reasonably confident that $A$ is close to a degree $d$ polynomial.)
2. *Concatenation tests*:

For $i = 1$ to $k$, run Poly-concat-corr-check$(\zeta_i, A, B, X_i; w, d, F)$.
Run Poly-concat-corr-check$(\zeta_{k+1}, A, B, Z; w, d, F)$.
(If all these tests accept, then the verifier can be reasonably confident that the $X_i$'s and $Z$ are close to degree $d$ polynomials and that $A$ is their concatenation in the sense of Definition 47.)

3. (Next, the verifier tries to simulate $V^{\text{seq}}$ using $\mathcal{P}^{-1}_{N,w,H,F}(X_1), \ldots, \mathcal{P}^{-1}_{N,w,H,F}(X_k)$, and $\mathcal{P}^{-1}_{N,w,H,F}(Z)$ as the $k+1$ oracles. It uses the Curve-corr-check procedure to produce any desired entries in those oracles.)
   (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 $V^{\text{seq}}$ on random string $R$. For conciseness, denote these by just $(i_1, q_1), \ldots, (i_l, q_l)$.
   (c) For $j = 1$ to $l$, Let $h_j = \#(q_j)$ and $z_j = (h_j, \zeta_{i_j})$. Note that each $z_j \in F^{w+1}$.
   (d) Run Curve-corr-check$(A, B, O, z_1, \ldots, z_l; w+1, d, F)$ and let $(a_1, \ldots, a_l)$ be its output.
   (e) If any of the responses $a_1, \ldots, a_l$ is not a member of $\{0,1\}$ then reject.
   (f) If $V^{\text{seq}}$ 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 58** *Given a circuit $C$ of size $n$ on $km$ inputs, for some $k < \infty$, the verifier $V^{\text{par}}$ has the following properties:*

1. *There exists a constant $\alpha_k$ such that $V^{\text{par}}$ tosses at most $\alpha_k \log n$ coins.*
2. *$V^{\text{par}}$ probes the oracles $3k + 5$ times.*
3. *There exists a constant $\beta_k$ such that the answer size of the oracles are bounded by $\log^{\beta_k} n$.*
4. *There exists a constant $\gamma_k$ such that the computation of $V^{\text{par}}$'s verdict as a function of the oracles responses can be expressed as a circuit of size $\log^{\gamma_k} n$.*

**Proof:** Step 1 (low degree test) 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) $V^{\text{par}}$ tosses $r$ random coins. In addition in Step 3(d) it tosses $\log|F|(w+3)$ coins. Step 3(d) 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 we bound the size of the circuit that expresses its decision. Note that all of the verifier's tasks, except in Step 3(f), involve simple interpolation and field arithmetic, and they can be done in time polynomial (more specifically in time cubic) in $\log|F|$ and $d$, $k$ and $l$. By Theorem 39 the action in Step 3(f) involves evaluating a circuit of size $\log^{c_3} n$. Thus the net size of $V^{\text{par}}$'s circuit is bounded by some polynomial in $\log n$. ∎

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

**Lemma 59** *Let $C$ be a circuit with $km$ inputs and size $n$. Let $x_1, \ldots, x_k$ be such that $C(x_1, \ldots, x_k)$ accepts. For $i = 1$ to $k$, let $X_i = \mathcal{P}_{N,w,H,F}(E^{\text{seq}}(x_i))$. Then, there exists an oracle $X_{k+1}$ such that, for all $R$,*

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

**Proof:** By the definition of an inner verifier, there is a string $\pi$ such that $(V^{\mathsf{seq}})^{E^{\mathsf{seq}}(x_1),\ldots,E^{\mathsf{seq}}(x_k),\pi}$ accepts $C$ with probability 1. Let $Z$ be the polynomial $\mathcal{P}_{N,w,H,F}(\pi)$ that encodes $\pi$. Notice that the $X_i$'s as well as $Z$ are polonomials of degree $m|H| = \frac{\log^2 N}{\log\log N}$. Let $A = \mathsf{concat}_{X_1,\ldots,X_k,Z}$. Then $A$ is a polynomial of degree at most $d = m|H| + k - 1$. Let $B = \mathsf{lines}_A$. Let $O$ be the oracle which, given any tuple of points $z_1,\ldots,z_l \in F^{w+1}$ as a query, returns the univariate degree $d(l-1)$ polynomial $A|_{\mathsf{curve}_{z_1,\ldots,z_l}}$, where $\mathsf{curve}_{z_1,\ldots,z_l}$ is the curve defined in Definition 53. Let $X_{k+1}$ be the oracle combining $Z$, $A$, $B$ and $O$, i.e., such that $X_{k+1}(1,\cdot) = Z(\cdot)$, $X_{k+1}(2,\cdot) = A(\cdot)$, $X_{k+1}(3,\cdot) = B(\cdot)$, $X_{k+1}(4,\cdot) = 0(\cdot)$, Then it is clear that $(V^{\mathsf{par}})^{X_1,\ldots,X_{k+1}}$ accepts $C$ with probability 1. ∎

**Lemma 60** *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 $\delta_0$ be as in Theorem 45. Let $e_{\mathsf{seq}} < 1$ denote the error of the verifier $V^{\mathsf{seq}}$. We will show that $e = \max\{.999, 1-\delta_0, 1-(1-e_{\mathsf{seq}})(.4)\}$ satisfies the stated condition.

Suppose the test $\mathsf{Poly\text{-}test}(A,B;w+1,d,F)$ accepts $A$ with probability $\max\{.999, 1-\delta_0\}$. Then, by the soundness condition of Theorem 45, there exists a degree $d$ polynomial $p : F^{w+1} \to F$, such that $\Delta(A,p) \le \min\{2\delta_0, .002\} \le .002$. (Notice that to apply Theorem 45 we need to ensure that $|F| \ge \alpha(d+1)^3$. This does hold for our choice of $F$ and $d$ — the latter is $o(\log^2 N)$ and the former is $\Omega(\log^6 N)$.)

Now suppose each of the concatenation tests accepts with probability at least .999. Let $\epsilon = \Delta(A,p)$ and let $p_i(\cdot) = p(\cdot,\zeta_i)$. Then, by Lemma 49, the distance $\Delta(X_i,p_i) \le (1-.999) + 2\sqrt{\epsilon} + \frac{d}{|F|}$ which (given the choice of $d$ and $|F|$) is at most $1/4$. Thus $p_i$ is the unique polynomial with distance at most $1/4$ from $X_i$. Similarly, $p_{k+1}$ is the unique degree $d$ polynomial with distance at most $1/4$ to $Z$.

For $i = 1,\ldots,k+1$, let $Y_i = \mathcal{P}^{-1}_{N,w,H,F}(X_i) = \mathcal{P}^{-1}_{N,w,H,F}(p_i)$ be the decodings of the $p_i$'s. We now show that $V^{\mathsf{seq}}$ accepts the proof oracles $Y_1,\ldots,Y_{k+1}$ with reasonable probability, thus implying — since $V^{\mathsf{seq}}$ is an inner verifier — that $(E^{\mathsf{seq}})^{-1}(Y_1)(E^{\mathsf{seq}})^{-1}(Y_2)\cdots(E^{\mathsf{seq}})^{-1}(Y_k)$ is a satisfying assignment to the input circuit $C$. This will prove the theorem.

In the program of $V^{\mathsf{par}}$, consider fixing $R \in \{0,1\}^r$. Suppose further that $R$ is such that verifier $V^{\mathsf{par}}$ accepts after Step 3 with probability more than 0.6 (where the probability is over the choice of the randomness used in Step 3(d)). We show that then the verifier $(V^{\mathsf{seq}})^{Y_1,\ldots,Y_{k+1}}$ accepts on random string $R$. Recall that whether or not $V^{\mathsf{par}}$ accepts after Step 3 depends upon whether or not $(a_1,\ldots,a_l)$, the output of Curve-corr-check, is a satisfactory reply to the queries of verifier $V^{\mathsf{seq}}$ using the random string $R$. By Proposition 46, the probability that the procedure Curve-corr-check outputs something other than $p(z_1),\ldots,p(z_l)$ is at most $2\sqrt{\epsilon} + \frac{d}{|F|-1} + \frac{dl}{|F|}$. By our choice of $|F|$, $d$, and $\epsilon$, we have $2\sqrt{\epsilon} + \frac{d}{|F|-1} + \frac{dl}{|F|} < 0.6$. Thus if the probability that the verifier $V^{\mathsf{par}}$ accepts in step 3 is more than 0.6 it must be the case that $p(z_1),\ldots,p(z_l)$ is a satisfactory answer to the verifier $V^{\mathsf{seq}}$. But $p$ is simply a polynomial encoding of the concatenation of $Y_1,\ldots,Y_k,Y_{k+1}$. Lemma 50 implies that if we were to run $V^{\mathsf{seq}}$ on the oracles $Y_1,\ldots,Y_{k+1}$ using random string $R$, then the answer it would get is exactly $p(z_1),\ldots,p(z_l)$! We conclude that $V^{\mathsf{seq}}$ accepts the oracles $Y_1,\ldots,Y_{k+1}$ on random string $R$.

Now we finish the proof. Suppose $V^{\mathsf{par}}$ accepts in Step 3(f) with probability more than $1-(1-e_{\mathsf{seq}})(1-0.4)$. Then the fraction of $R \in \{0,1\}^r$ for which it accepts with probability more than 0.6 must be at least $1-e_{\mathsf{seq}}$. We conclude that $V^{\mathsf{seq}}$ accepts the oracles $Y_1,\ldots,Y_k,Y_{k+1}$ with probability at least $1-e_{\mathsf{seq}}$, whence we invoke soundness property

of $V^{\mathsf{seq}}$ to conclude that $(E^{\mathsf{seq}})^{-1}(Y_1), \ldots, (E^{\mathsf{seq}})^{-1}(Y_k)$ is a satisfying assignment to the circuit $C$. Since $Y_i = \mathcal{P}^{-1}_{N,h,w,F}(X_i)$, the theorem has been proved. ∎

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 $\alpha_k$, $\beta_k$ and $\gamma_k$ be as in Proposition 58. We let $c_1 = \alpha_k$, $p = 3k + 5$, $c_2 = \gamma_k$, $c_3 = \beta_k$. Further, let $e$ be as in Lemma 60. Then, by Proposition 58 and Lemmas 59 and 60, 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 [78] show that approximating the chromatic number of a graph within a factor $n^{\epsilon}$ 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 [79] 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 $\Pi$, where $\Pi$ is a nontrivial graph property closed under vertex deletion.) Khanna, Linial and Safra [71] 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 [3] 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 $\ell_\infty$ norm. Karger, Motwani, and Ramkumar [70] 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 [37] or the survey [4].

**Improved analysis of outer verifiers.** Our construction of an efficient outer verifier for NP languages (Theorem 17) 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 [21] 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 [43] construct verifiers with 2 queries, arbitrarily small error, and constant answer sizes. Tardos [96] 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 [87] 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. Very recently, Raz and Safra [88] have constructed verifiers making constant number of queries with logarithmic randomness and answer size, where the error is as low as $2^{-\log^{1-\epsilon} n}$ for every $\epsilon > 0$. An alternate construction is given in Arora and Sudan [7].

**Better non-approximability results.** Part of the motivation for improving the construction of outer verifiers is to improve the ensuing non-approximability results. 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 $\epsilon$. This line of research was initiated by Bellare, Goldwasser, Lund, and Russell [21] who provided improved non-approximability results with explicit constants for a number of problems, such as MAX 3SAT, MAX CLIQUE, Chromatic Number and Set Cover. Since then a number of works [20, 23, 41, 43, 44, 50, 61, 62, 63] have focussed on improving

the non-approximability results. These results have culminated with some remarkably strong inapproximability results listed below. The attributions only cite the latest result in a long sequence of papers - the interested reader can look up the cited result for details of the intermediate results.

1. Håstad [63] has shown that MAX 3SAT is NP-hard to approximate within a factor $7/8 - \epsilon$, for every $\epsilon > 0$.
2. Håstad [62] has shown that MAX CLIQUE is hard to approximate to within a factor $n^{1-\epsilon}$, for every positive $\epsilon$, unless NP = RP.
3. Feige and Kilian [44], combined with Håstad [63], show that Chromatic number is hard to approximate to within a factor $n^{1-\epsilon}$, for every positive $\epsilon$, unless NP = RP.
4. Feige [41] shows that Set Cover is hard to approximate to within $(1 - o(1)) \ln n$ unless NP $\subset$ DTIME($n^{\log \log n}$.

We note all the results above are tight.

A related issue is the following: "What is the smallest value of $q$ for which NP = $\cup_{c>0}$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 [21] to to 22 [43] to 16 [20] to at most 9 (which is the best that can be inferred directly from [63] - though this result is not optimized.) Parameters that are somewhat related to $q$ are the *free-bit* parameter introduced by Feige and Kilian [43] and the *amortized free-bit* parameter introduced by Bellare and Sudan [23]. Reducing the latter parameter is necessary and sufficient for proving good inapproximability results for MAX CLIQUE [20], and Håstad [61, 62] 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 [11] 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 [95]. The main hurdle in further improvement seemed to be Arora and Safra's proof [6] of Theorem 69, which requires a field size quadratic in the degree. Polishchuk and Spielman [86] present a new proof of Theorem 69 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. In particular, the low-degree test Theorem 45 has been improved significantly since then. Arora and Sudan [7] show that part 2 of Theorem 45 can be improved to show that if a function passes the low degree test with probability $\epsilon$ then it is $1 - \epsilon$-close to some degree $d$ polynomial. A similar result for a different test was shown earlier by Raz and Safra [88]. The proof of the correctness of the linearity test (Theorem 27) has also been improved in works by Bellare, Goldwasser, Lund and Russell [21] and Bellare, Coppersmith, Hastad, Kiwi and Sudan [19].

**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 [14] had earlier shown that proofs can be checked by examinining only poly$(\log n)$ bits in them.)

Actually, by looking at Remark 18, a stronger statement can be obtained: there is a polynomial-time transformation between normal mathematical proofs and our "probabilistically checkable certificates of theoremhood."

**Surprising algorithms:** There has also been significant progress on designing better approximation algorithms for some of the problems mentioned earlier. Two striking results in this direction are those of Goemans and Williamson [57] and Arora [2]. Goemans and Williamson [57] show how to use semidefinite programming to give better approximation algorithms for MAX-2SAT and MAX-CUT. Arora [2] has discovered a polynomial time approximation scheme (PTAS) for Euclidean TSP and Euclidean Steiner tree problem. (Mitchell [80] independently discovered similar results a few months later.) These were two notable problems not addressed by our hardness result in this paper since they were not known to be MAX SNP-hard. Arora's result finally resolves the status of these two important problems.

### 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. [34, 35], 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 [6] 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. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 37th IEEE Symp. on Foundations of Computer Science*, pp 2-12, 1996.

[3] S. ARORA, L. BABAI, J. STERN, AND Z. SWEEDYK. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317-331, April 1997.

[4] S. ARORA AND C. LUND. Hardness of approximations. In *Approximation Algorithms for NP-hard problems*, D. Hochbaum, ed. PWS Publishing, 1996.

[5] S. ARORA, R. MOTWANI, S. SAFRA, M. SUDAN, AND M. SZEGEDY. PCP and approximation problems. *Unpublished note*, 1992.

[6] S. ARORA AND S. SAFRA. Probabilistic checking of proofs: a new characterization of NP. To appear *Journal of the ACM*. Preliminary version in *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

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

[8] G. AUSIELLO, A. D'ATRI, AND M. PROTASI. Structure Preserving Reductions among Convex Optimization Problems. *Journal of Computer and Systems Sciences*, 21:136-153, 1980.

[9] G. AUSIELLO, A. MARCHETTI-SPACCAMELA AND M. PROTASI. Towards a Unified Approach for the Classification of NP-complete Optimization Problems. *Theoretical Computer Science*, 12:83-96, 1980.

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

[11] 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.

[12] L. BABAI AND L. FORTNOW. Arithmetization: a new method in structural complexity theory. *Computational Complexity*, 1:41-66, 1991.

[13] L. BABAI, L. FORTNOW, AND C. LUND. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3-40, 1991.

[14] 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.

[15] L. BABAI AND K. FRIEDL. On slightly superlinear transparent proofs. *Univ. Chicago Tech. Report*, CS-93-13, 1993.

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

[17] 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.

[18] 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.

[19] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi and M. Sudan. Linearity testing in characteristic two. *IEEE Transactions on Information Theory* 42(6):1781-1795, November 1996.

[20] M. Bellare, O. Goldreich and M. Sudan. Free bits, PCPs and non-approximability — towards tight results. To appear *SIAM Journal on Computing.* Preliminary version in *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science,* IEEE, 1995. Full version available as TR95-024 of ECCC, the *Electronic Colloquium on Computational Complexity*, `http://www.eccc.uni-trier.de/eccc/`.

[21] 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).

[22] M. Bellare and P. Rogaway. The complexity of approximating a nonlinear program. *Journal of Mathematical Programming B*, 69(3):429-441, September 1995. Also in *Complexity of Numerical Optimization*, Ed. P. M. Pardalos, World Scientific, 1993.

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

[24] 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.

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

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

[27] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of the ACM*, 41(4):630-647, July 1994.

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

[29] 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.

[30] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549-595, December 1993.

[31] J. Cai, A. Condon, and R. Lipton. PSPACE is provable by two provers in one round. Journal of Computer and System Sciences, 48(1):183-193, February 1994.

[32] 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.

[33] 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.

[34] 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.

[35] A. Condon, J. Feigenbaum, C. Lund and P. Shor. Random debaters and the hardness of approximating stochastic functions. *SIAM Journal on Computing*, 26(2):369-400, April 1997.

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

[37] 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`.

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

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

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

[41] 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.

[42] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268-292, March 1996.

[43] 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.

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

[45] U. Feige and L. Lovász. 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.

[46] L. FORTNOW, J. ROMPEL, AND M. SIPSER. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545-557, November 1994.

[47] 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.

[48] K. FRIEDL, ZS. HÁTSÁGI AND A. SHEN. Low-degree testing. *Proceedings of the Fifth Symposium on Discrete Algorithms*, ACM, 1994.

[49] K. FRIEDL AND M. SUDAN. Some improvements to low-degree tests. *Proceedings of the Third Israel Symposium on Theory and Computing Systems*, 1995.

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

[51] M. GAREY AND D. JOHNSON. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43-49, 1976.

[52] M. GAREY AND D. JOHNSON. "Strong" NP-completeness results: motivation, examples and implications. *Journal of the ACM*, 25:499-508, 1978.

[53] M. GAREY AND D. JOHNSON. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[54] M. GAREY, D. JOHNSON AND L. STOCKMEYER. Some simplified NP-complete graph problems. *Theoretical Computer Science* 1:237-267, 1976.

[55] P. GEMMELL AND M. SUDAN. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169-174, September 1992.

[56] 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.

[57] M. GOEMANS AND D. WILLIAMSON. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115-1145, November 1995.

[58] O. GOLDREICH. A Taxonomy of Proof Systems. In *Complexity Theory Retrospective II*, L.A. Hemaspaandra and A. Selman (eds.), Springer-Verlag, New York, 1997.

[59] S. GOLDWASSER, S. MICALI, AND C. RACKOFF. The knowledge complexity of interactive proof-systems. *SIAM J. on Computing*, 18(1):186-208, February 1989.

[60] R. GRAHAM. Bounds for certain multiprocessing anomalies, *Bell Systems Technical Journal*, 45:1563-1581, 1966.

[61] J. HÅSTAD. Testing of the long code and hardness for clique. *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1996.

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

[63] J. HÅSTAD. Some optimal inapproximability results. *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1997,

[64] O. IBARRA AND C. KIM. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463-468, 1975.

[65] R. IMPAGLIAZZO AND D. ZUCKERMAN. How to Recycle Random Bits. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[66] D. JOHNSON. Approximation algorithms for combinatorial problems. *J. Computer and Systems Sci.* 9:256-278, 1974.

[67] V. KANN. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27-35, 1991.

[68] N. KARMAKAR AND R. 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.

[69] R. 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.

[70] D. KARGER, R. MOTWANI, AND G. RAMKUMAR. On approximating the longest path in a graph. *Algorithmica*, 18(1):82-98, May 1997.

[71] 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.

[72] S. KHANNA, R. MOTWANI, M. SUDAN, AND U. VAZIRANI. On syntactic versus computational views of approximability. To appear *SIAM Journal on Computing*. Preliminary version in *Proceedings of the Thirty Fifth Annual Symposium on the Foundations of Computer Science,* IEEE, 1994.

[73] 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.

[74] D. LAPIDOT AND A. SHAMIR. Fully parallelized multi-prover protocols for NEXP-time. *Journal of Computer and System Sciences*, 54(2):215-220, April 1997.

[75] 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 [97].

[76] 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.

[77] C. LUND, L. FORTNOW, H. KARLOFF, AND N. NISAN. Algebraic Methods for Interactive Proof Systems. *J. ACM*, **39**, 859–868, 1992.

[78] C. LUND AND M. YANNAKAKIS. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960-981, September 1994.

[79] 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.

[80] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: Part II-A simple PTAS for geometric $k$-MST, TSP, and related problems. Preliminary manuscript, April 30, 1996. *To appear in SIAM J. Computing.*

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

[82] C. PAPADIMITRIOU AND M. YANNAKAKIS. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences* 43:425-440, 1991.

[83] C. PAPADIMITRIOU AND M. YANNAKAKIS. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 1992.

[84] A. PAZ AND S. MORAN. Non-deterministic polynomial optimization problems and their approximation. *Theoretical Computer Science*, 15:251-277, 1981.

[85] S. PHILLIPS AND S. SAFRA. PCP and tighter bounds for approximating MAXSNP. *Manuscript*, Stanford University, 1992.

[86] A. POLISHCHUK AND D. SPIELMAN. Nearly Linear Sized Holographic Proofs. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

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

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

[89] R. RUBINFELD. *A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs*. Ph.D. thesis, U.C. Berkeley, 1990.

[90] R. RUBINFELD AND M. SUDAN. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing* 25(2):252-271, April 1996.

[91] S. SAHNI. Approximate algorithms for the 0/1 knapsack problem, *Journal of the ACM*, 22:115-124, 1975.

[92] S. SAHNI AND T. GONZALES. P-complete approximation problems. *Journal of the ACM*, 23:555-565, 1976.

[93] J. SCHWARTZ. Probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701-717, 1980.

[94] A. SHAMIR. IP = PSPACE. *Journal of the ACM*, 39(4):869-877, October 1992.

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

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

[97] B. TRAKHTENBROT. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing* 6:384-400, 1984.

[98] L. WELCH AND E. BERLEKAMP. Error correction of algebraic block codes. *US Patent* Number 4,633,470 (filed: 1986).

[99] M. YANNAKAKIS. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17(3):475-502, November 1994.

[100] D. ZUCKERMAN. On unapproximable versions of NP-complete problems. *SIAM Journal on Computing*, 25(6):1293-1304, December 1996

# A Correctness of the low degree test

This section proves the correctness of the Low Degree test by proving part (2) of Theorem 45. Let $m$ and $d$ be arbitrary positive integers that should be considered fixed in the rest of this section. Let $F$ be a finite 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 61** *Let $f: F^m \to F$ be any function, $(x,h) \in F^{2m}$, 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

$$\delta_f^{(B)} \equiv \Pr_{x,h \in F^m, t \in F}[f(x+th) \neq B(x,h)[t]]. \tag{13}$$

We wish to prove part (2) of Theorem 45, 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 62** *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 63** *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 21 that if the restriction is $(1/2 - d/|F|)$-close to some degree $d$ polynomial, then the line polynomial is uniquely defined. In the case when the line polynomial is not uniquely*

*defined, we assume that some polynomial is picked arbitrarily but* consistently. *More specifically, notice that the lines $l_{x,h}$ and $l_{x+t_1h,t_2h}$ are identical for every $t_1 \in F$ and $t_2 \in F - \{0\}$. We will assume that $P^{(f,d)}_{x,h} = P^{(f,d)}_{x+t_1h,t_2h}$. This will simplify our notation in the following proof.*

For $f: F^m \to F$, we denote by $B^{(f,d)}$ a $d$-oracle in which $B^{(f,d)}(x,h) = P^{(f,d)}_{x,h}$ for every $(x,h) \in F^{2m}$. We denote by $\delta_{f,d}$ the quantity $\delta_f^{(B(f,d))}$, i.e., the failure rate of $B^{(f,d)}$ with respect to $f$. Notice that by Remark 63, we have that

$$\begin{aligned}
\delta_{f,d} &= \Pr_{x,h\in F^m, t\in F}\left[P^{(f,d)}_{x,h}[t] = f(x+th)\right] \\
&= \Pr_{x,h\in F^m, t\in F}\left[P^{(f,d)}_{x+th,h}[0] = f(x+th)\right] \\
&= \Pr_{x'=x+th, h\in F^m}\left[P^{(f,d)}_{x',h}[0] = f(x')\right] \qquad (14)
\end{aligned}$$

**Proposition 64** *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 pair $(x,h) \in F^{2m}$, 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 64 that the following result suffices to prove part 2 of Theorem 45.

**Theorem 65 (The Section's Main Theorem)** *There are fixed constants $\delta_0 > 0$ and $\alpha < \infty$ such that the following is true. For all integers $m, d > 0$ and every field $F$ of size $\alpha d^3$, if $f: F^m \to F$ is a function such that $\delta_{f,d} \le \delta_0$, then $f$ is $2\delta_{f,d}$-close to $F^{(d)}_m$.*

We note that our proof of Theorem 65 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 follows:

$$\hat{f}_d(x) \equiv \text{plurality}^1{}_{h\in F^m}\left\{P^{(f,d)}_{x,h}(0)\right\} \quad \forall x \in F^m. \qquad (15)$$

(Note that the line $l_{x,h}$ passes through $x$. Further, $x$ is the point $x + 0\cdot h$, so $P^{(f,d)}_{x,h}(0)$ is the value produced by the line polynomial $P^{(f,d)}_{x,h}$ 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 65 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 66** *For any function $f: F^m \to F$ and any integer $d$, $\Delta(f, \hat{f}_d) \le 2\delta_{f,d}$.*

[1] The plurality of a multiset of elements is the most commonly occuring element, with ties being broken arbitrarily.

**Proof:** Let $B$ be the set given by

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

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

$$\begin{aligned} \Pr_{x,h}[f(x) \neq P_{x,h}^{(f,d)}(0)] &\geq \frac{1}{2} \cdot \Pr_x[x \in B] & (16) \\ &= \frac{|B|}{2\,|F|^m}. & (17) \end{aligned}$$

But by (14), we also know that

$$\Pr_{x,h}[f(x) \neq P_{x,h}^{(f,d)}(0)] = \delta_{f,d}. \quad (18)$$

From (17) and (18) 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 65, we will need a certain lemma regarding bivariate functions. The following definition is required to state it.

**Definition 67** *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 68** *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 65 when the number of variables, $m$, is 2. This subcase will be crucial in the proof for general $m$.

**Theorem 69 ([6])** *There are constants $\epsilon_0, c > 0$ such that the following holds. Let $d$ be any positive integer, $\epsilon \leq \epsilon_0$, 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 70** *Let $\epsilon_0, c$ be as given by Theorem 69. Let $\epsilon < \min\{\epsilon_0, \frac{1}{5} - \frac{d}{5|F|}\}$, $d$ be any non-negative integer and $F$ a field of cardinality at least $cd^3$. Let $f: F^2 \to F$ satisfy the hypothesis of Theorem 69. Let $\epsilon'$ be such that $0 < \epsilon' < \frac{1}{2} - \frac{2d}{|F|} - 5\epsilon$.*

*Then if $x_0, y_0 \in F$ satisfy*

$$\Pr_x[f(x, y_0) \neq r_x^{(f,d)}(y_0)] \leq \epsilon' \tag{19}$$

$$\Pr_y[f(x_0, y) \neq c_y^{(f,d)}(x_0)] \leq \epsilon' \tag{20}$$

*then $r_{x_0}^{(f,d)}(y_0) = c_{y_0}^{(f,d)}(x_0)$.*

**Proof:** Theorem 69 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 $g(x, \cdot)$ denote the univariate degree $d$ polynomial that describes $g$ on the row that passes through $x$. We first argue that

$$\Pr_{x \in F}[g(x, \cdot) \neq r_x^{(f,d)}(\cdot)] \leq 5\epsilon + \frac{d}{|F|} \tag{21}$$

The reason is that $r_x^{(f,d)}(\cdot)$ 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

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

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

Immediately from (21) it follows $\Pr_x[g(x, y_0) \neq r_x^{(f,d)}(y_0)] \leq 5\epsilon + \frac{d}{|F|}$. The hypothesis of the corollary implies that $y_0 \in F$ is such that $\Pr_x[f(x, y_0) \neq r_x^{(f,d)}(y_0)] \leq \epsilon'$. Thus we find that

$$\begin{aligned} \Pr_x[f(x, y_0) \neq g(x, y_0)] &\leq \Pr_x\left[f(x, y_0) \neq r_x^{(f,d)}(y_0)\right] + \Pr_x\left[r_x^{(f,d)}(y_0) \neq g(x, y_0)\right] \\ &\leq 5\epsilon + \epsilon' + \frac{d}{|F|}. \end{aligned}$$

But the previous statement just says that the univariate polynomial $g(\cdot, y_0)$ describes $f$ on all but $(5\epsilon + \epsilon' + \frac{d}{|F|})\,|F|$ 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 65. 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 71** *Let $\epsilon_0$ and $c$ be as given in Theorem 69. Let $d$ be any positive integer, $F$ be a field of size at least $\max\{6d, cd^3\}$, and $\epsilon$ be any constant satisfying $0 < \epsilon < \min\{\frac{1}{36}, \epsilon_0\}$. Then every function $f : F^m \to F$ satisfies:*

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

**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 $\delta$ for $\delta_{f,d}$. Pick $h_1, h_2 \in_R F^m$ and let $M = M_{h_1,h_2} : F^2 \to F$ be the function given by $M(y, z) = f(x + yh_1 + zh_2)$.

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 70 for $y_0 = t_0$ and $z_0 = 0$ and $\epsilon' = \epsilon$. Then it will follow that $c^{(M,d)}_{z_0}(y_0) = r^{(M,d)}_{y_0}(z_0)$. But note that by definition of $M$, $c^{(M,d)}_{z_0} = P^{(f,d)}_{x,h_1}$ and $r^{(M,d)}_{y_0} = P^{(f,d)}_{x+t_0h_1,h_2}$. This suffices to prove the lemma since now we have

$$P^{(f,d)}_{x,h_1}(t_0) = c^{(M,d)}_{z_0}(y_0)r^{(M,d)}_{y_0}(z_0) = P^{(f,d)}_{x+t_0h_1,h_2}(0).$$

We first verify that the first hypothesis for Corollary 70 holds: i.e., that $\epsilon$ and $\epsilon'$ satisfy $5\epsilon + \epsilon' + (2d)/|F| < 1/2$. Since by the hypothesis of the lemma we have $|F| \geq 6d$, we find that $2d/|F| \leq 1/3$ and thus we need to show that $5\epsilon + \epsilon' = 6\epsilon < 1/2 - 1/3 = 1/6$. The finally inequality holds since $\epsilon < 1/36$ by the hypothesis of the lemma.

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

$$\forall y \neq 0, \forall z, \quad \Pr_{h_1,h_2}\left[P^{(f,d)}_{x+yh_1,h_2}(z) \neq f(x + yh_1 + zh_2)\right] \leq \delta. \tag{22}$$

Note that the event $P^{(f,d)}_{x+yh_1,h_2}(z) \neq f(x + yh_1 + zh_2)$ may be rephrased as $r^{(M,d)}_y(z) \neq M(y, z)$. We now argue that

$$\Pr_{h_1,h_2}\left[\Pr_{y\neq 0,z}[M(y, z) \neq r^{(M,d)}_y(z)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon}. \tag{23}$$

To see this, consider the indicator variable $X_{h_1.h_2,y,z} = 1$ that is 1 if $P^{(f,d)}_{x+yh_1,h_2}(z) \neq f(x+yh_1+zh_2)$ and 0 otherwise. In what follows, let $E_r[A(r)]$ denote the expectation of $A$ as a function of the random variable $r$. In this notation, (22) can be expressed as

$$\forall y \neq 0, \forall z, \quad E_{h_1,h_2}\left[X_{h_1,h_2,y,z}\right] \leq \delta.$$

For $h_1, h_2 \in F^m$, define $Y_{h_1,h_2}$ to be $E_{y\neq 0,z}[X_{h_1,h_2,y.z}]$ where $y$ and $z$ are chosen uniformly and independently at random from $F - \{0\}$ and $F$ respectively. >From (22) it follows that

$$E_{h_1,h_2}[Y_{h_1,h_2}] = E_{h_1,h_2,y\neq 0,z}[X_{h_1,h_2,y,z}] \leq \max_{y\neq 0,z}\{E_{h_1.h_2}[X_{h_1,h_2,y,z}]\} \leq \frac{\delta}{\epsilon}.$$

(23) now follows by applying Markov's inequality to the random variable $Y_{h_1,h_2}$.[2]

[2]Recall that Markov's inequality states that for a non-negative random variable $Y$, $\Pr[Y \geq k] \leq E[Y]/k$.

Going back to (23), by accounting for the probability of the event $y = 0$, we conclude:

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

Applying Markov's inequality again to (22) (in a manner similar to the derivation of (23)) but this time fixing $z = z_0$, we get

$$\Pr_{h_1,h_2}\left[\Pr_{y}[M(y,z_0) \neq r_y^{(M,d)}(z_0)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{25}$$

We can similarly argue that

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[M(y,z) \neq c_z^{(M,d)}(y)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{26}$$

$$\text{and } \Pr_{h_1,h_2}\left[\Pr_{z}[M(y_0,z) \neq c_z^{(M,d)}(y_0)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{27}$$

Thus with probability at least $1 - 4(\delta/\epsilon + 1/|F|)$, none of the events (24)-(27) happen and all the hypotheses of Corollary 70 are satisfied. ▮

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

**Corollary 72** *Let $\epsilon_0$, $c$ be as given by Theorem 69. Let $0 < \epsilon < \min\{\frac{1}{36}, \epsilon_0\}$. Then every function $f : F^m \to F$ satisfies*

$$\forall x \in F^m, t \in F, \quad \Pr_{h\in F^m}\left[\hat{f}_d(x+th) \neq P_{x,h}^{(f,d)}(t)\right] \leq \frac{8\delta_{f,d}}{\epsilon} + \frac{8}{|F|} \tag{28}$$

*if $F$ is a finite field of size at least $\max\{6d, cd^3\}$.*

**Proof:** Let $\delta = \delta_{f,d}$. 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 (28) 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. Using Lemma 71 we have,

$$\Pr_{h,h_1}[P_{x+th,h_1}^{(f,d)}(0) \neq P_{x,h}(t)] \leq (4\delta/\epsilon + 4/|F|).$$

But for each $h \in B_{x,t}$, (by the definition of $B_{x,t}$) we have

$$\Pr_{h_1\in F^m}\left[P_{x+th,h_1}^{(f,d)}(0) \neq P_{x,h}(t)\right] \leq \frac{1}{2}.$$

$$\text{Thus } \Pr_{h,h_1\in_R F^m}\left[P_{x+th,h_1}^{(f,d)}(0) \neq P_{x,h}(t)\right] \geq \frac{|B_{x,t}|}{2|F|^m}.$$

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 73** *Let $\epsilon_0$ and $c$ be as in Theorem 69. Let $F$ be a finite field of size at least $\max\{6d, cd^3\}$ 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*

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

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

$$M(y, 0) = \hat{f}_d(x + yh) \text{ and } M(y, z) = f(x + yh + zh_1 + yzh_2) \text{ for } z \neq 0.$$

Notice that by the definition of $M$, for every $y$, $r^{(M,d)}_y(z) = P^{(f,d)}_{x+yh,h_1+yh_2}(z)$ and for every $z \neq 0$, $c^{(M,d)}_z(y) = P^{(f,d)}_{x+zh_1,h+zh_2}(y)$. Finally $c^{(M,d)}_0(y) = P^{(\hat{f}_d,d)}_{x,h}(y)$. Thus the 0-th column of $M$ is independent of $h_1, h_2$ and the goal of the lemma is to show that $M(0,0) = c^{(M,d)}_0(0)$. We will show, by an invocation of Corollary 70, that the event $c^{(M,d)}_0(0) = r^{(M,d)}_0(0)$ happens with probability strictly greater than $8\delta/\epsilon + 8/|F|$ over the random choices of $h_1$ and $h_2$. But by Corollary 72, we have $M(0,0) = \hat{f}_d(x) = P^{(f,d)}_{x,h_1}(0) = r^{(M,d)}_0(0)$ 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 72.) Our choice of $\epsilon, \delta$ implies that the following event happens with with positive probability (over the choice of $h_1$ and $h_2$): "$\hat{f}_d(x) = r^{(M,d)}_0(0) = P^{(\hat{f}_d,d)}_{x,h}(0)$." 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 70 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 $2\delta$ (by Lemma 66). 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 72). Thus we have shown

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

As in the proof of Lemma 71, we can now conclude that

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[M(y, z) \neq r^{(M,d)}_y(z)] \geq \epsilon\right] \leq \frac{\delta_1}{\epsilon} + \frac{1}{|F|}. \tag{29}$$

$$\text{and } \Pr_{h_1,h_2}\left[\Pr_{y}[M(y, 0) \neq r^{(M,d)}_y(0)] \geq \epsilon\right] \leq \frac{\delta_1}{\epsilon} + \frac{1}{|F|}. \tag{30}$$

For the columns 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 $\delta$ to claim that, for every $y \in F$,

$$\Pr_{h_1,h_2}\left[M(y,z) = f(x+zh_1+y(h+zh_2)) = P^{(f,d)}_{x+zh_1,h+zh_2}(y) = c^{(M,d)}_y(z)\right] = \delta.$$

As in the proof of Lemma 71 we can argue that

$$\Pr_{h_1,h_2}\left[\Pr_{y,z}[M(y,z) \neq c^{(M,d)}_z(y)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{31}$$

$$\text{and } \Pr_{h_1,h_2}\left[\Pr_{z}[M(0,z) \neq c^{(M,d)}_z(0)] \geq \epsilon\right] \leq \frac{\delta}{\epsilon} + \frac{1}{|F|}. \tag{32}$$

Thus with probability at least $1 - 16(\delta_1/\epsilon + \delta/\epsilon + 2/|F|)$, none of the events (29)-(32) happen and we can apply Corollary 70.

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 in the statement of the lemma. ▌

Now we can prove the main theorem of this section.

**Proof of Theorem 65:** The choice of $\alpha$ and $\delta_0$ are made as follows: Let $\epsilon = \min\{1/36, \epsilon_0\}$, where $\epsilon_0$ is as in Theorem 69. Now pick $\alpha$ to be $\max\{c, 600/\epsilon\}$, where $c$ is as given by Theorem 69. Now let $\delta_0 = \frac{\epsilon^2}{624}$. Notice that $\delta_0$ is positive, and $\alpha < \infty$.

Let $f : F^m \to F$ be a function over a field $F$ of size at least $\alpha d^3$ such that $\delta_{f,d} < \delta_0$. It can be verified that $\delta$, $\epsilon$ and $F$ satisfy the conditions required for the application of Lemma 73. Thus 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 [90]). (See also [49] for a tight analysis of the condition under which this equivalence holds.) By Lemma 66 $\Delta(f,\hat{f}_d) \leq 2\delta_{f,d}$. Thus $f$ is $2\delta_{f,d}$-close to a degree $d$ polynomial. ▌