Locally Testable Codes and PCPs of Almost-Linear Length*

Oded Goldreich† Madhu Sudan‡

Abstract

Locally testable codes are error-correcting codes that admit very efficient codeword tests. Specifically, using a constant number of (random) queries, non-codewords are rejected with probability proportional to their distance from the code.

Locally testable codes are believed to be the combinatorial core of PCPs. However, the relation is less immediate than commonly believed. Nevertheless, we show that certain PCP systems can be modified to yield locally testable codes. On the other hand, we adapt techniques we develop for the construction of the latter to yield new PCPs. Our main results are locally testable codes and PCPs of almost-linear length. Specifically, we present:

- *Locally testable (linear) codes in which k information bits are encoded by a codeword of length approximately $k \cdot \exp(\sqrt{\log k})$. This improves over previous results that either yield codewords of exponential length or obtained almost quadratic length codewords for sufficiently large non-binary alphabet.*
- *PCP systems of almost-linear length for SAT. The length of the proof is approximately $n \cdot \exp(\sqrt{\log n})$ and verification in performed by a constant number (i.e., 19) of queries, as opposed to previous results that used proof length $n^{1+O(1/q)}$ for verification by q queries.*

The novel techniques in use include a random projection of certain codewords and PCP-oracles, an adaptation of PCP constructions to obtain "linear PCP-oracles" for proving conjunctions of linear conditions, and a direct construction of locally testable (linear) codes of sub-exponential length.

*Extended Abstract. For further details, see [14].

†Department of Computer Science, Weizmann Institute of Science, Rehovot, ISRAEL. oded@wisdom.weizmann.ac.il. Supported by the MINERVA Foundation, Germany.

‡Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. madhu@mit.edu. Supported in part by NSF Awards CCR 9875511, CCR 9912342, and MIT-NTT Award MIT 2001-04.

1 Introduction

We study the existence of (good) error-correcting codes that admit very efficient codeword tests. Specifically, we require the testing procedure to use only a constant number of (random) queries, and reject non-codewords with probability proportional to their distance from the code. Such codes may be thought of as a combinatorial counterparts of the complexity theoretic notion of probabilistically checkable proofs (PCPs). They were formally introduced by Friedl and Sudan [11]. Here we initiate a systematic study of this notion.

Some examples: Codeword testing is meaningful only for good codes. In particular, it is easy to test trivial codes (e.g., for codes containing all possible strings of certain length or, on the other extreme, for codes containing a single codewords (or very few codewords)). One non-trivial code allowing efficient testing is the Hadamard code: the codewords are linear functions, and so codeword testing amounts to linearity testing [7].

The drawback of the Hadamard code is that k bits of information are encoded using a codeword of length 2^k. (The k information bits represent the k coefficients of a linear function $\{0,1\}^k \to \{0,1\}$, and bits in the codeword correspond to all possible evaluation points.)

The question addressed in this work is whether one can hope for a better relation between the number of information bits, k, and the length of the codeword, denoted n. Specifically, *can n be polynomial or even linear in k?* For (sufficiently large) *non-binary* alphabet, Friedl and Sudan [11] showed that n can be almost quadratic in k. We show that n may be *almost-linear* in k (i.e., $n = k^{1+o(1)}$), *even for the binary alphabet*.

1.1 Relation to PCP

It is a common belief, among PCP enthusiasts, that the PCP Theorem [1, 2] already provides codes as we desire. Consider the mapping of standard witnesses for, say SAT, to PCP-oracles. When applied to an instance of SAT that is a tautology, the map typically induces a good error-correcting code mapping k information bits to codewords of length poly(k) (or almost linear in k, when using [18]). The common belief is that the PCP verifier also yields a codeword test. However, this is not quite true: It is only guaranteed that each passing oracle can be "decoded" to a corresponding NP-witness, but this does not mean that a passing oracle is (close to) a valid codeword (because the "decoding" procedure is actually stronger than is standard in coding theory), or that only codewords pass the test with probability one. For example, part of the PCP oracle is supposed to encode an m-variate polynomial of *individual degree* d, yet the PCP verifier will also accept the encoding of any m-variate polynomial of *total degree* $m \cdot d$ (and the "decoding" procedure will work in this case too).

Still, we show that many known PCP constructions can be modified to yield good codes with efficient codeword tests. We stress that these modifications are non-trivial and furthermore are unnatural in the context of PCP. Yet, they yield coding results of the type we seek (e.g., see Theorem 2.1).

On the other hand, a technique that emerges naturally in the context of our study of efficient codeword tests yields improved results on the length of efficient PCP proofs. Specifically, we obtain constant-query PCP systems that utilize oracles that are shorter than known before (see Theorem 2.3).

1.2 Relation to Locally Decodable Codes

The problem of designing efficient codeword tests *seems* easier than the question of designing efficient decoding procedures that allow to recover any desired information bit by reading only a constant number of bits in the codeword. Our results confirm this intuition:

- We show the existence of almost-linear (i.e., $n = k^{1+o(1)}$) length (binary) codes supporting codeword tests with a constant number of queries. In contrast, it was shown that locally decodable codes cannot have almost-linear length [17].[1]
- For large alphabet, we show almost-linear length coordinate-linear codes in which testing requires only *two* queries. In contrast, it was shown that coordinate-linear codes with *two* query recovery require exponential length [13].

2 Formal Setting

Throughout this work, all oracle machines (i.e., codeword testers and PCP verifiers) are non-adaptive; that is, they determine their queries based solely on their input and random choices. This is in contrast to adaptive orcale machines that may determine their queries based on answers obtained to prior queries. Since our focus is on positive results, this makes our results only stronger.

[1]If q queries are used for recovery then $n = \Omega(k^{1+(1/(q-1))})$.

2.1 Codes

We consider codes mapping a sequence of k input symbols into a sequence of $n \geq k$ symbols over the same alphabet, denoted Σ, which may but need not be the binary alphabet. Such a generic code is denoted by $\mathcal{C} : \Sigma^k \to \Sigma^n$. Throughout this paper, *the integers k and n are to be thought of as parameters*, and Σ may depend on them. Thus, we actually discuss infinite families of codes (which are associated with infinite sets of possible k's), and whenever we say that some quantity of the code is a constant we mean that this quantity is constant for the entire family (of codes). Typically, we seek to have Σ as small as possible, desire that $|\Sigma|$ be a constant (i.e., does not depend on k), and are most content when $\Sigma = \{0, 1\}$ (i.e., a binary code).

Distance between n-symbol sequences over Σ is defined in the natural manner; that is, for $u, v \in \Sigma^n$, the distance $\Delta(u, v)$ is defined as the number of locations on which u and v differ (i.e., $\Delta(u, v) \stackrel{\text{def}}{=} |\{i : u_i \neq v_i\}|$, where $u = u_1 \cdots u_n \in \Sigma^n$ and $v = v_1 \cdots v_n \in \Sigma^n$). The distance of a code $\mathcal{C} : \Sigma^k \to \Sigma^n$ is the minimum distance between its codewords; that is, $\min_{a \neq b}\{\Delta(\mathcal{C}(a), \mathcal{C}(b))\}$. *Throughout this work, we focus on codes of "large distance"*; specifically, codes $\mathcal{C} : \Sigma^k \to \Sigma^n$ of distance $\Omega(n)$.

The distance of $w \in \Sigma^n$ from a code $\mathcal{C} : \Sigma^k \to \Sigma^n$ is the minimum distance between w and the codewords; that is, $\min_a\{\Delta(w, \mathcal{C}(a))\}$. An interesting case is of non-codewords that are "relatively far from the code", which may mean that their distance from the code is greater than (say) half the distance of the code.

By a codeword test (for the code $\mathcal{C} : \Sigma^k \to \Sigma^n$) we mean a randomized (non-adaptive) oracle machine (called tester) that given oracle access to $w \in \Sigma^n$ (viewed as a function $w : \{1, ..., n\} \to \Sigma$) satisfies the following two conditions:[2]

- *Accepting codewords*: For any $a \in \Sigma^k$, given oracle access to $w = \mathcal{C}(a)$, the tester accepts with probability 1.
- *Rejection of non-codeword*: For every $w \in \Sigma^n$ that is at distance ϵn from $\mathcal{C}$, given oracle access to w, the tester rejects with probability $\Omega(\epsilon) - o(1)$. (The $o(1)$ term can be avoided if we consider only non-codewords that are at distance more than $\epsilon_0 n$ from the code, for some constant $\epsilon_0 > 0$.)[3]

[2]Both the following conditions may be meaningfully relaxed. For example, the tester may be allowed to err with small probability in case it is given oracle access to a codeword, and the rejection condition may be restricted to non-codewords that are relatively far from the code. Since our results are positive, it make sense for us to use the stronger definition provided below.

[3]Following this alternative (i.e., of considering only non-codewords that are very far from the code), we may use an alternative formulation (which is more standard in the "property testing" literature; cf. [20, 12]). Specifically, we may require that every non-codeword that is at least $\epsilon_0 n$-far from the code be rejected with probability at least $1/2$.

We say that the code $\mathcal{C} : \Sigma^k \to \Sigma^n$ is locally testable if it has a codeword test that makes a *constant number of queries*. Our main result regarding codes is

Theorem 2.1 *For every $c > 0.5$ and infinitely many k's, there exist locally testable codes with binary alphabet such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$. Furthermore, these codes are linear and have distance $\Omega(n)$.*

Theorem 2.1 (as well as Part 2 of Theorem 2.2) vastly improves over the Hadamard code (in which $n = 2^k$), which is the only locally testable *binary* code previously known. Theorem 2.1 is proven by combining Part 1 of the following Theorem 2.2 with non-standard modifications of standard PCP constructions. Due to space limitations, the proof of this theorem is omitted from this extended abstract, and can be found in the full version of this paper [14]. Instead we give here a simpler construction that gives the following, weaker but self-contained, result.

Theorem 2.2
1. *For every $c > 0.5$ and infinitely many k's, there exist locally testable codes with non-binary alphabet Σ such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$ and $\log|\Sigma| = \exp((\log k)^c) = k^{o(1)}$. Furthermore, the tester makes two queries.*
2. *For every $c > 1$ and infinitely many k's, there exist locally testable codes binary alphabet such that $n < k^c$.*

In both cases, the codes are linear in a suitable sense and have distance $\Omega(n)$.

Part 1 improves over the work of Friedl and Sudan [11], which only yields $n = k^{2+o(1)}$. The set of k's for which such codes exist is reasonable dense; in both cases, if k is in the set then the next integer in the set is smaller than $k^{1+o(1)}$.

We comment that (good) *binary* codes cannot be tested using two queries (cf. [6]). In contrast it can be shown that locally testable codes over small alphabets can be modified such that the tester only uses randomness that is logarithmic in the codeword and only makes three queries. We stress that the stated modification increases the length of the codewords by a constant factor. (See our technical report [14] for a proof.)

2.2 PCP

A probabilistic checkable proof (PCP) system for a set L is a probabilistic polynomial-time (non-adaptive) oracle machine (called verifier), denoted V, satisfying

- *Completeness*: For every $x \in L$ there exists an oracle π_x so that V, on input x and access to oracle π_x, always accepts x.
- *Soundness*: For every $x \notin L$ and every oracle π, machine V, on input x and access to oracle π, rejects x with probability at least $\frac{1}{2}$.

As usual, we focus on PCP systems with *logarithmic randomness complexity* and *constant query complexity*. This means that, without loss of generality, the length of the oracle is polynomial in the length of the input. However, we aim at PCP systems that utilize oracles that are of almost-linear length. Our main result regarding such PCP systems is

Theorem 2.3 *For every $c > 0.5$, there exists an almost-linear time randomized reduction of SAT to a promise problem that has a 19-query PCP system that utilizes oracles of length $\exp((\log n)^c) \cdot n = n^{1+o(1)}$, where n is the length of the input. Furthermore, the reduction maps k-bit inputs to n-bit inputs such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$.*

This should be compared to the PCP system for SAT of Polishchuk and Spielman [18] that when utilizing oracles of length $n^{1+\epsilon}$ makes $O(1/\epsilon)$ queries. In contrast, our PCP system utilizing oracles of length $n^{1+o(1)}$ while making 19 queries.

3 Direct Constructions of Codes

In this section, we prove Theorem 2.2. Although we do not use any variant of the PCP Theorem, our constructions are somewhat related to known PCP constructions in the sense that we use codes (and analysis) that appear (at least implicitly) in the latter. Specifically, we will use results regarding low-degree tests that were proven for deriving the PCP Theorem [1, 2]. We stress that we neither use the (complex) parallelization procedure (of [1, 2]) nor the full power of the proof composition paradigm (of [2], which is more complex than the classical notion of concatenated codes [10] used below).

3.1 The Basic Code (FS-Code)

Our starting point is a code proposed by Friedl and Sudan [11] based on a low-degree test due to Rubinfeld and Sudan [20].

Let F be a finite field, and m, d be integer parameters such that (typically) $m \le d < |F|$. Denote by $P_{m,d}$ the set of m-variate polynomials of total degree d over F. We represent each $p \in P_{m,d}$ by the list of its $\binom{m+d}{d}$ coefficients; that is, $|P_{m,d}| = |F|^{\binom{m+d}{d}}$. (For $m \le d$, we use $|P_{m,d}| < |F|^{(2d/m)^m}$.)

Denote by L_m the set of lines over F^m, where each line is define by two points $a, b \in F^m$; that is, for $a = (a_1, ..., a_m)$ and $b = (b_1, ..., b_m)$, the line $\ell_{a,b}$ consists of the set of $|F|$ points $\{\ell_{a,b}(t) \stackrel{\rm def}{=} ((a_1 + tb_1), ..., (a_m + tb_m)) : t \in F\}$.

We consider the code $\mathcal{C} : P_{m,d} \to \Sigma^{|L_m|}$, where $\Sigma = F^{d+1}$; that is, $\mathcal{C}$ assigns each $p \in P_{m,d}$ a ($|L_m|$-long) sequence of Σ-values, where each Σ-value corresponds to a different element of L_m. The element associated with $\ell \in L_m$ in the ($|L_m|$-long) sequence $\mathcal{C}(p)$, denoted $\mathcal{C}(p)_\ell$, is the univariate polynomial that represents the values of the polynomial $p : F^m \to F$ on the line ℓ; that is, for $\ell_{a,b} \in L_m$, the univariate polynomial $\mathcal{C}(p)_{\ell_{a,b}}$ can be formally written as $q_{a,b}(z) \stackrel{\rm def}{=} p(\ell_{a,b}(z)) = p((a_1 + b_1 z), ..., (a_m + b_m z))$. Since the polynomial p has total degree d, so does the univariate polynomial $q_{a,b}$.

To evaluate the basic parameters of the code $\mathcal{C}$, let use consider it as mapping $\Sigma^k \to \Sigma^n$, where indeed $n = |L_m| = |F|^{2m}$ and $k = \log|P_{m,d}|/\log|\Sigma|$. Note that

$$k \;=\; \frac{\log|P_{m,d}|}{\log|\Sigma|} \;=\; \frac{\binom{m+d}{d}\log|F|}{(d+1)\log|F|} \;=\; \frac{\binom{m+d}{m}}{d+1} \qquad (1)$$

which, for $m \ll d$, is approximated by $(d/m)^m/d \approx (d/m)^m$. Using $|F| = \mathrm{poly}(d)$, we have $n = |F|^{2m} = \mathrm{poly}(d^m)$, and so k is polynomially related to n (provided, say, $m < \sqrt{d}$). Note that the code has large distance (since the different $\mathcal{C}(p)$'s tend to disagree on most lines).

The Codeword Test: The test consists of selecting two random lines that share a random point, and checking that the univariate polynomials associated with these lines yield the same value for the shared point. That is, to check whether $w \in \Sigma^{|L_m|}$ is a codeword, we select a random point $r \in F^m$, and two random lines ℓ', ℓ'' going through r (i.e., $\ell'(t') = r$ and $\ell''(t'') = r$ for some $t', t'' \in F$), obtain the answer polynomials q' and q'' (i.e., $q' = w_{\ell'}$ and $q'' = w_{\ell''}$) and check whether they agree on the shared point (i.e., whether $q'(t') = q''(t'')$). This test is essentially the one analyzed in [1], where it is shown that (for $|F| = \mathrm{poly}(d)$) if the oracle is ϵ-far from the code then this is detected with probability $\Omega(\epsilon)$.

3.2 Random Truncation of the FS-Code

Our aim is to tighten the relation between k and n. Recall that the gap between them is due to two sources; firstly, the analysis in [1] required a field F that is polynomially bigger than the degree d. This problem can be eliminated using the better analysis of [18], which only requires $|F| = \Omega(d)$ (see [11]). The second problem is that n is quadratic in $|F|^m$, whereas $k = o(d^m) = o(|F|^m)$. Thus, to obtain n almost-linear in k, we must use a different code.

We will use a random projection (or "truncation") of the FS-code on approximately $|F|^m$ of the coordinates. Let $R_m \subset L_m$ be a random subset of $O(|F|^m \log|F|)$ lines. We consider the code $\mathcal{C}^{R_m} : P_{m,d} \to \Sigma^{|R_m|}$, where the element associated with $\ell_{a,b} \in R_m \subset L_m$ in the sequence $\mathcal{C}^{R_m}(p)$ is the univariate polynomial that represents the values of the polynomial $p : F^m \to F$ on the line $\ell_{a,b}$. When R_m is (unimportant or) understood from the context, we shorthand $\mathcal{C}^{R_m}$ by $\mathcal{C}$.

To evaluate the basic parameters of the code $\mathcal{C}$, let use consider it as mapping $\Sigma^k \to \Sigma^n$, where $n = |R_m| = O(|F|^m \log|F|)$ (and as before $k = \log|P_{m,d}|/\log|\Sigma|$). Thus, for $m \ll d$, we have $k \approx d^{m-1}/m^m$ and, for $|F| = O(d)$, we have $n = O(|F|^m \log|F|) = O(d)^m$. We highlight two possible settings of the parameters:

1. Using $d = m^m$, we get $k \approx m^{m^2-2m}$ and $n = m^{m^2+o(m)}$, which yields $n \approx \exp(\sqrt{\log k}) \cdot k$ and $\log|\Sigma| = \log|F|^{d+1} \approx d\log d \approx \exp(\sqrt{\log k})$.
2. Letting $d = m^e$ for constant $e > 1$, we get $k \approx m^{(e-1)m}$ and $n \approx m^{em}$, which yields $n \approx k^{e/(e-1)}$ and $\log|\Sigma| \approx d\log d \approx (\log k)^e$.

The Codeword Test: The original codeword test can be extended to the current setting. Specifically, the new test consists of selecting two random lines in R_m that share a random point, and checking that the univariate polynomials associated with these lines yield the same value for the shared point. (We stress that we first select uniformly a point $r \in F^m$, and next select two lines in R_m that pass through r.) We prove that this codeword test for the randomly-truncated code $\mathcal{C}^{R_m}$ works as well as the codeword test for the basic FS-code.

Lemma 3.1 *Let $|F| = \Omega(d)$ and $|F| < \exp(m^m)$. Then, for $1 - o(1)$ fraction of the possible choices of R_m of size n, the following holds for every $w \in \Sigma^n$: if the distance of w from the code $\mathcal{C}^{R_m}$ is ϵn then the probability that the above codeword test rejects is $\Omega(\epsilon) - o(1)$.*

Proof sketch: First we reduce the analysis of the above codeword test (which compares the value given to two intersecting lines) to an analysis of a point-vs-line test that compares the value of a *suitable* function $f : F^m \to F$ on a random point with the value induced by (the polynomial associated with) a random line passing through this point. Fixing any R_m and any $w \in \Sigma^n$, we construct a random function $f : F^m \to F$ by selecting uniformly, for each $r \in F^m$, a line ℓ in R_m that passes through r and setting $f(r)$ accordingly (i.e., $f(r) = w_\ell(t)$ where $r = \ell(t)$). We note that the probability that the original intersecting-lines test accepts w equals the probability that the point-vs-line test accepts w *along with the resulting random* f, because the (random) value $f(r)$ (obtained from f) may be viewed as obtained from a (second) random line that passes through r. Thus, it suffices to analyze the point-vs-line test as applied to w and the corresponding random f. This will be done in two stages: In the first stage we relate the distance of w from the code $\mathcal{C} = \mathcal{C}^{R_m}$ to the distance of f from the set $P_{m,d}$, and in the second stage we relate the rejection probability of w and f to the distance of f from $P_{m,d}$.

First stage: We will show that for every $p \in P_{m,d}$, the (fractional) distance of f from p approximates the (fractional) distance of w from $\mathcal{C}(p)$. For simplicity, we first assume that R_m covers all points uniformly (i.e., each point in F^m resides in exactly $|F| \cdot |R_m|/|F^m|$ lines of R_m). Let $p \in P_{m,d}$ and denote by εn the distance of w from $\mathcal{C}(p)$; that is, $w_\ell \neq \mathcal{C}(p)$ $(= p(\ell))$ on an ε fraction of the ℓ's in R_m. For each $\ell \in R_m$ for which $w_\ell \neq \mathcal{C}(p)$ it is the case that w_ℓ disagrees with p on almost all (i.e., all but d) points that reside on the line ℓ (because both $w_\ell(\cdot)$ and $p(\ell(\cdot))$ are low-degree polynomials that determine the corresponding values). Since $f(r)$ is defined according to a random line $\ell \in R_m$ that passes through r, it holds that the expected (fractional) disagreement of a random f with p is at least $(1-(d/|F|)) \cdot \varepsilon$. Furthermore, since f is define independently on each point of F^m, with probability at least $1-\exp(\varepsilon|F|^m)$, a random f disagrees with p on at least a $\varepsilon/2$ fraction of the points. Using the union bound (for all $p \in P_{m,d}$) and $|P_{m,d}| < |F|^{(2d/m)^m} \ll 2^{\varepsilon|F|^m}$ (for $\varepsilon > 2^{-m}$), with very high probability, the distance of a random f from every $p \in P_{m,d}$ (i.e., f's distance from $P_{m,d}$) approximates (up-to an additive term of $(\epsilon/2) - o(1)$) the distance of w from the corresponding $\mathcal{C}(p)$. We conclude that the expected distance of a random f from the set $P_{m,d}$ approximates the distance of w from the code $\mathcal{C}$.

Recall that, in the above analysis, we have assumed that R_m covers all points uniformly (i.e., each point resides on the same number of lines in R_m). In general, this is not the case. Yet, with very high probability, a random set R_m cover almost all points in an almost uniform manner. This "almost uniformity" suffices for extending the above analysis. Thus, for almost all R_m's, the distance of w from the code $\mathcal{C}^{R_m}$ is well-approximated by the distance of a corresponding random function f from the set $P_{m,d}$.

Second stage: We turn to analyze the performance of the point-vs-line test applied to any $w \in \Sigma^n$ and a corresponding random $f : F^m \to F$ (constructed as above). Following [20, 1, 2, 18], we observe that for each possible function $f : F^m \to F$ there exists an optimal strategy of answering all possible line-queries (such that the acceptance probability of the line-vs-point test is maximized). Specifically, for a fixed function f, and each line ℓ, the optimal way to answer the line-query ℓ is given by the degree d univariate polynomial that agrees with the value of f on the maximum number of points of ℓ. Thus, the optimal acceptance probability of the line-vs-point test on f depends only on f (and not on w, which may not be optimal for f). Furthermore, this probability is the average of quantities (i.e., the agreement of f with the best univariate polynomial) that f associates with each of the possible lines. Let us denote by $D_\ell(f)$ the fractional disagreement of f restricted to ℓ with the best univariate polynomial. Then, by the relevant results in [1, 2, 18], the average of $D_\ell(f)$ *taken over all lines* (i.e., over L_m) is linearly related to the distance of f from $P_{m,d}$. Clearly, the rejection probability of our test (i.e., the line-vs-point test for lines uniformly selected in R_m, when applied to w and f as above) is lower-bounded by the average of the $D_\ell(f)$'s over the lines in R_m (rather than over the set of all lines, L_m). Now, *for each fixed* f, with probability $1-\exp(-|R_m|)$, the average of the $D_\ell(f)$'s (taken over all lines) is approximated (up-to some constant) by the average taken over a random set R_m. Taking the union bound over all $|F|^{|F|^m}$ functions f's we conclude that, for almost all R_m, the point-vs-line test rejects each f with probability that is linearly related to the distance of f from $P_{m,d}$ (because $\exp(-|R_m|) \cdot |F|^{|F|^m} = o(1)$).

By the first stage, for almost all R_m's, the distance of each w is related to the expected distance of a corresponding random f from $P_{m,d}$, whereas by the second stage (for almost all R_m's) each w and the corresponding random f is rejected by the point-vs-line with probability that is linearly related to the distance of f from $P_{m,d}$. Combining these two facts, the lemma follows. ■

F-linearity: The (modified as well as the original) code $\mathcal{C}$ is F-linear; that is, the individual F-elements in the codeword sequence are linear combinations (over F) of the F-elements in the information being encoded. Equivalently, for every $\alpha', \alpha'' \in F$ and every $p', p'' \in P_{m,d}$, it holds that $\mathcal{C}(\alpha' p' + \alpha'' p'')_\ell = \alpha' \mathcal{C}(p')_\ell + \alpha'' \mathcal{C}(p'')_\ell$, for every line ($\Sigma$-coordinate) ℓ. This is the case because $\mathcal{C}(\alpha' p' + \alpha'' p'')_\ell$ equals the univariate polynomial (in z) given by $(\alpha' p' + \alpha'' p'')(\ell(z)) = \alpha' p'(\ell(z)) + \alpha'' p''(\ell(z))$, which in turn equals $\alpha' \mathcal{C}(p')_\ell + \alpha'' \mathcal{C}(p'')_\ell$.

Using the first parameter-setting (i.e., $d = m^m$), we establish Part 1 of Theorem 2.2.

3.3 Decreasing the alphabet size

The above construction uses quite a big alphabet (i.e., $\Sigma = F^{d+1}$). Our aim in this subsection is to maintain the above performance while using a smaller alphabet (i.e., F rather than F^{d+1}). This is achieved by concatenating the above code (which encodes information by a sequence of n degree d univariate polynomials over F) with the following inner-code that maps F^{d+1} to $F^{n'}$, where n' is sub-exponential in $k' \stackrel{\text{def}}{=} d+1$.

For a (suitable) constant d', let $k' = h^{d'}$ and $[h] = \{1, ..., h\}$. As a warm-up, consider the special case of $d' = 2$. In this case, the code $\mathcal{C}'$ maps bilinear forms in x_i's and y_i's (with coefficients $(c_{i,j})_{i,j\in[h]}$) to the values of these forms under all possible assignments. That is, $\mathcal{C}' : F^{h^2} \to F^{|F|^{2h}}$ maps the sequence of coefficients $(c_{i,j})_{i,j\in[h]}$ to the sequence of values $(v_{a_1,...,a_h,b_1,...,b_h})_{a_1,...,a_h,b_1,...,b_h \in F}$ where $v_{a_1,...,a_h,b_1,...,b_h} = \sum_{i,j\in[h]} c_{i,j} \cdot a_i b_j$. In general (i.e., arbitrary $d' \geq 1$), the inner-code $\mathcal{C}' : F^{k'} \to F^{n'}$ maps d'-linear forms in the variables sets $\{z_i^{(1)} : i \in [h]\}, ..., \{z_i^{(d')} : i \in [h]\}$ to the values of these

d'-linear forms under all possible assignments to these $d'h$ variables. That is, $\mathcal{C}'$ maps the sequence of coefficients $(c_{i_1,...,i_{d'}})_{i_1,...,i_{d'}\in[h]}$ to the sequence of values $(v_{a_1^{(1)},...,a_h^{(1)},...,a_1^{(d')},...,a_h^{(d')}})_{a_1^{(1)},...,a_h^{(1)},...,a_1^{(d')},...,a_h^{(d')}\in F}$ where $v_{a_1^{(1)},...,a_h^{(1)},...,a_1^{(d')},...,a_h^{(d')}} = \sum_{i_1,...,i_{d'}\in[h]} c_{i_1,...,i_{d'}} \cdot \prod_{j=1}^{d'} a_{i_j}^{(j)}$. Thus, ($k' = h^{d'}$ and) $n' = |F|^{d'h} = \exp(d' \cdot (k')^{1/d'} \log |F|)$.

Testing the inner-code: A valid codeword is a multi-linear function (in the variable sets $\{z_i^{(1)} : i \in [h]\}, ..., \{z_i^{(d')} : i \in [h]\}$); that is, for each j, a valid codeword is linear in the variables $z_i^{(j)}$'s. Thus, testing whether a sequence belongs to the inner-code amounts to d' linearity checks. Specifically, for each j, we randomly select $\overline{r} = (r_1^{(1)}, ..., r_h^{(1)}, ..., r_1^{(d')}, ..., r_h^{(d')})$ and $s_1^{(j)}, ..., s_h^{(j)}$, and compare $v_{\overline{r}} + v_{0,...,0,s_1^{(j)},...,s_h^{(j)},0,...,0}$ to $v_{(t_1^{(1)},...,t_h^{(1)},...,t_1^{(d')},...,t_h^{(d')})}$, where $t_i^{(j)} = r_i^{(j)} + s_i^{(j)}$ and $t_i^{(j')} = r_i^{(j')}$ for $j' \neq j$. To simplify the analysis, we also let the test employ a total low-degree test (to verify that the codeword is a multi-variate polynomial of total-degree d').[4] (The total-degree test uses $d' + 2$ queries, and so our codeword test uses $3d' + d' + 2$ queries.)

Lemma 3.2 *If the distance of $w' \in F^{n'}$ from $\mathcal{C}'$ is $\epsilon n'$ then the probability that the codeword test for $\mathcal{C}'$ rejects is $\Omega(\epsilon)$.*

Testing the concatenated-code: In order to test the concatenated code, we first test (random instances of) the inner-code and next use self-correction on the latter to emulate the testing of the outer-code. Specifically, for the information q' (i.e., a univariate polynomial of degree $d = h^{d'} - 1$ over F) encoded in a letter of the outer-code, we need to recover from the (inner) encoding the value $q'(t)$, where t is some element of F (which is determined by the outer test). However, the value $q'(t) = \sum_{i_1,...,i_{d'}\in[h]} q'_{i_1,...,i_{d'}} t^{(i_1-1)+(i_2-1)h+\cdots+(i_{d'}-1)h^{d'-1}}$ equals the entry of $\mathcal{C}'(q')$ that is associated with the sequence $(t^0, ..., t^{h-1}, t^0, ..., t^{(h-1)h}, ..., t^0, ..., t^{(h-1)h^{d'-1}})$. Self-correction of the desired entry is performed via polynomial interpolation, and requires only $d' + 1$ queries. Thus, the concatenated code can be tested by making $O(d')$ queries.

Notes: Observe that the inner-code is linear (over F), and thus so is also the concatenated code. Furthermore, the codeword test is a conjunction of $(O(d'))$ linear tests. Alternatively, we may perform one of these linear tests, selected at random (with equal probability). Regarding the parameters of the concatenated code, suppose that in the outer-code we use the setting $d = m^e$ (for constant $e > 1$), and in the inner-code we use $d' = 2e$. Then, we obtain a code that maps $F^{kk'}$ to $F^{nn'}$, where $n \approx k^{e/(e-1)}$ and $n' \approx \exp(d^{1/d'}) \approx \exp((\log k)^{e/d'}) = \exp(\sqrt{\log k}) = k^{o(1)}$ (using $d \approx (\log k)^e$). Thus, $nn' \approx (kk')^{e/(e-1)}$ and $|F| = O(d) \approx (\log k)^e$ (as before).

3.4 A Binary Code

The last step is to derive a binary code. This is done by concatenating the above code with a Hadamard code, while assuming that $F = GF(2^{k''})$. The Hadamard code is used to code elements of F by binary sequences of length $n'' \stackrel{\text{def}}{=} 2^{k''}$. See details in our technical report [14]. Part 2 of Theorem 2.2 follows.

4 Nearly linear-sized PCPs

In this section we give a probabilistic construction of nearly-linear sized PCPs for SAT. More formally, we reduce SAT probabilistically to a promise problem recognized by a PCP verifier tossing $(1 + o(1)) \log n$ random bits (on inputs of length n) and queries a proof oracle in a constant number of bits and has perfect completeness and soundness arbitrarily close to $\frac{1}{2}$. We stress that the constant number of bits is explicit and small. Specifically, if the $o(1)$ function in the randomness is allowed to be as large as $1/\text{poly} \log \log n$, then the number of queries can be reduced to 16 bits. The little $o(1)$ function can be reduced to $O(\sqrt{\log \log n / \log n})$ for a small cost in the number of queries, which now goes up to 19 bits. These improvements are obtained by using and improving results of Harsha and Sudan [16].

We get our improvements by applying the "random truncation" method (introduced in Section 3) to certain *constant-prover one-round proof systems*, which are crucial ingredients in the constructions of PCPs. Typically, these proof systems use provers of very different sizes, and by applying the "random truncation" method we obtain an equivalent system in which all provers have size roughly equal to the size of the smallest prover in the original scheme. At this point, we reduce the randomness complexity to be logarithmic in the size of the provers (i.e., and thus logarithmic in the size of the smallest original prover).

Recall that typical PCP constructions are obtained by the technique of proof composition introduced by Arora and Safra [2]. In this technique, an "outer verifier", typically a verifier for a constant prover one round proof system, is composed with an "inner verifier" to get a new PCP verifier. The new verifier essentially inherits the randomness complexity of the outer verifier and the query complexity of the

[4] We conjecture that the codeword test operates well also without employing the total-degree test, but the augmented codeword test is certainly easier to analyze.

inner verifier. Since our goal is to reduce the randomness complexity of the composed verifier, we achieve this objective by reducing the randomness complexity of the outer verifier.

We show how to apply the random truncation to the verifier of a specific 3-prover one-round proof system used by Harsha and Sudan [16]. Their verifier is a variant of the one constructed by Raz and Safra [19] (see also, Arora and Sudan [3]), which are, in turn, variants of a verifier constructed by Arora et al. [1]. All these verifiers share the common property of working with provers of "imbalanced" sizes. We manage to reduce the size of the provers to the size of the smallest one, and consequently reduce the randomness of the verifier to $(1 + o(1)) \log n$ (i.e., logarithmic in the prover size). We stress that this part is not generic but relies on properties of the proof of soundness in, say, [16], which are abstracted below. Applying the composition lemmas used/developed in [16] to this new verifier gives us our efficient PCP constructions.

4.1 Improved 3-Prover Proof System for NP

We recall the notion of a constant-prover one-round interactive proof system (MIP).

Definition 4.1 *For positive reals c, s, integer p and functions $r, a : \mathsf{Z}^+ \to \mathsf{Z}^+$, we say that a language $L \in$* $\mathrm{MIP}_{c,s}[p, r, a]$ (or, L has a p-prover one-round proof system with answer length a) *if there exists a probabilistic polynomial-time verifier V interacting with p provers $P_1, \ldots, P_p$ such that*

Operation: *On input x of length n, the verifier tosses $r(n)$ coins, generates queries $q_1, \ldots, q_p$ to provers $P_1, ..., P_p$, obtain the corresponding answers $a_1, \ldots, a_p \in \{0, 1\}^{a(n)}$, and outputs a Boolean verdict that is a function of x, its randomness and the answers $a_1, \ldots, a_p$.*

Completeness: *If $x \in L$ then there exist strategies $P_1, \ldots, P_p$ such that V accepts their response with probability at least c.*

Soundness: *If $x \notin L$ then for every sequence of prover strategies $P_1, \ldots, P_p$, machine V accepts their response with probability at most s, which is called the* soundness error.

Harsha and Sudan [16] presented a randomness efficient 3-prover one-round proof system with answer length $\mathrm{poly}(\log n)$ and randomness complexity $(3 + \epsilon) \log_2 n$, where $\epsilon > 0$ is an arbitrary constant and n denotes the length of the input. Here we reduce the randomness required by their verifier to $(1 + o(1)) \log n$.

Before going on we introduce a notion that will be useful in this section — namely, the notion of a *length preserving reduction*. For a function $\ell : \mathsf{Z}^+ \to \mathsf{Z}^+$, a reduction is $\ell(n)$-length preserving if it maps instances of length n to instances of length at most $\ell(n)$.

Lemma 4.2 *For every $\epsilon > 0$ and functions $m(n)$, $\ell(n)$ satisfying $\ell(n) = \Omega(m(n)^{\Omega(m(n))} n^{1+\Omega(1/m(n))})$, SAT reduces in probabilistic polynomial time, under $\ell(n)$-length preserving reductions to a promise problem $\Pi \in \mathrm{MIP}_{1,\epsilon}[3, (1 + 1/m(n)) \log n + O(m(n) \log m(n)), m(n)^{O(1)} n^{O(1/m(n))}]$.*

Before proving this lemma, let us see some special cases obtained by setting $m(n) = \mathrm{poly}(\log \log n)$ and $m(n) = \sqrt{\log n}$, respectively in the above lemma.

Corollary 4.3 *For every $\mu > 0$ and every polynomial p, there exists a promise problem $\Pi \in \mathrm{MIP}_{1,\mu}[3, (1 + 1/p(\log \log n)) \cdot \log n, 2^{\mathrm{poly}(\log \log n)}]$ such that SAT reduces probabilistically to Π under $n^{1+(1/p(\log \log n))}$-length preserving reductions.*

Corollary 4.4 *For every $\mu > 0$, there exists a promise problem $\Pi \in \mathrm{MIP}_{1,\mu}[3, (1 + O((\log \log n)/\sqrt{\log n})) \cdot \log n, 2^{O(\sqrt{\log n} \log \log n)}]$, such SAT reduces probabilistically to Π under $n^{1+O((\log \log n)/\sqrt{\log n})}$-length preserving reductions.*

We defer the proof of Lemma 4.2 to Section 4.1.4. Here we give an overview of the proof steps. We modify the proof of [16] improving it in two steps. The proof of [16] first reduces SAT to a parametrized problem they call GapPCS under $\ell'(n)$-length preserving reductions for $\ell'(n) = n^{1+\gamma}$ for any $\gamma > 0$. Then they give a 3-prover MIP proof system for the reduced instance of GapPCS where the verifier tosses $(3 + \gamma) \log \ell'(n)$ random coins.

Our first improvement shows that the reduction of [16] actually yields a stronger reduction than stated there, in two ways. First we note that their proof allows for smaller values of $\ell(n)$ than stated there, allowing in particular for the parameters we need. Furthermore, we notice that their result gives rise to instances from a restricted class, for which slightly more efficient protocols can be designed. In particular, we can reduce the size of the smallest prover in their MIP protocol to roughly $\ell(n)$ (as opposed to their result which gives a prover of size $\ell(n)^{1+\gamma}$ for arbitrarily small γ).

The second improvement is more critical to our purposes. Here we improve the randomness complexity of the MIP verifier of [16], by applying a random truncation. To get this improvement we need to abstract the verifier of [16]. This is done in Section 4.1.1. We then show how to transform such a verifier into one with $(1 + o(1)) \log n$ randomness. This transformation comes in three stages, described in Sections 4.1.2-4.1.4.

4.1.1 Abtracting the verifier of [16]

The verifier of [16] interacts with three provers which we'll denote P, P_1, and P_2. We will let Q, Q_1, and Q_2 denote the question space of the provers respectively; and we'll let A, A_1, and A_2 denote the space of answers of the provers respectively. Denote by $V_x(r, a, a_1, a_2)$, the acceptance predicate of the verifier on input x, where r denotes the verifier's coins, and a (resp., a_1, a_2) the answer of prover $f = P$ (resp., P_1, P_2). (Note: The value of V_x is 1 if the verifier accepts.) We'll usually drop the subscript x unless needed. Let us denote by $q(r)$, (resp. $q_1(r)$, $q_2(r)$) the verifier's query to P (resp., P_1, P_2) on random string $r \in \Omega$, where Ω denotes the space of verifier's coins. We note that the following properties hold for the 3-prover proof system of [16].

1. The acceptance-predicate decomposes: $V(r, a, a_1, a_2) = V_1(r, a, a_1) \wedge V_2(r, a, a_2)$, where V_1 and V_2 are predicates.

2. Sampleability: The verifier only tosses $O(\log n)$ coins. Thus, it is feasible to sample from various specified subsets of the space of all possible coin outcomes. For example, given $S_1 \subseteq Q_1$, we can uniformly generate in $\mathrm{poly}(n)$-time a sequence of coins r such that $q_1(r) \in S_1$.

3. Uniformity: The verifier's queries to prover P (resp. P_1, P_2) are uniformly distributed over Q (resp. Q_1, Q_2).

4. If x is a NO-instance, then for $V = V_x$, for small ϵ and every possible P strategy, there exists a subset $Q' = Q'_P \subseteq Q$ such that for every P_1, P_2 the following two conditions hold:

$$\Pr_r[q(r) \in Q' \wedge V_1(r, f(Q(r)), P_1(Q_1(r)))] < \frac{\epsilon}{2} \qquad (2)$$

$$\Pr_r[q(r) \notin Q' \wedge V_2(r, f(Q(r)), P_2(Q_2(r)))] < \frac{\epsilon}{2} \qquad (3)$$

4.1.2 The 3-prover MIP: Stage I

We start by modifying the verifier of [16] so that its questions to provers P_1 and P_2 are "independent" (given the question to the prover P). That is, we define a new verifier, denoted W, that behaves as follows

- On input x, let $V = V_x$ be the verifier's predicate and let V_1 and V_2 be as given in Property (1).

- Pick $q \in Q$ uniformly and pick coins r_1 and r_2 uniformly and independently from the set $\{r \in \Omega | q(r) = q\}$. [Here we use sampleability with respect to a specific set of r's.]

- Make queries q (which indeed equals $q(r_1) = q(r_2)$), $q_1 = q_1(r_1)$ and $q_2 = q_2(r_2)$, to P, P_1 and P_2, receiving answers $a = P(q)$, $a_1 = P_1(q_1)$ and $a_2 = P_2(q_2)$.

- Accept if and only if $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

Claim 4.5 *W has perfect completeness and soundness at most ϵ.*

Proof: The completeness is obvious, and so we focus on the soundness. Fix a NO-instance x and any set of provers P, P_1 and P_2. Let $Q' = Q'_P$ be the subset of Q as given by Property (4) of the MIP. The probability that W accepts is given by

$$\Pr_{q,r_1,r_2}[EV_1(r_1) \wedge EV_2(r_2)] \qquad (4)$$

where $EV_1(r_1) = V_1(r_1, P(q), P_1(q_1(r_1)))$ and $EV_2(r_2) = V_2(r_2, P(q), P_2(q_2(r_2)))$. Note that $q = q(r_1) = q(r_2)$, where (q and) r_1, r_2 are selected as above. Thus, EV_i only depends on r_i, and the shorthand above is legitimate. Note that the process of selecting r_1 and r_2 in (4) is equivalent to selecting each of them uniformly (though not independently). We thus upper bound (4) by

$$\Pr_{r_1}[q(r_1) \in Q' \wedge EV_1(r_1)] + \Pr_{r_2}[q(r_2) \notin Q' \wedge EV_2(r_2)].$$

Using Property (4), each term above is bounded by $\epsilon/2$ and thus the sum above is upper-bounded by ϵ. ■

4.1.3 The 3-prover MIP: Stage II

In the next stage, the crucial one in our construction, we reduce the size of the provers P_1 and P_2 by a random truncation. For sets $S_1 \subseteq Q_1$ and $S_2 \subseteq Q_2$, we define the (S_1, S_2)-restricted verifier W_{S_1,S_2} as follows:

- On input x, let $V = V_x$ be the verifier's predicate and let V_1 and V_2 be as given in Property (1).

- Pick $q \in Q$ uniformly and for $i \in \{1, 2\}$ pick coins r_i's uniformly and independently from the sets $\{r \in \Omega | q(r) = q \wedge q_i(r) \in S_i\}$. If either of the sets is empty, then the verifier simply accepts. [Here, again, we use sampleability of subsets of the verifier coins.]

- Make queries $q = q(r_1) = q(r_2)$, $q_1 = q_1(r_1)$ and $q_2 = q_2(r_2)$, to P, P_1 and P_2, receiving answers $a = P(q)$, $a_1 = P_1(q_1)$ and $a_2 = P_2(q_2)$.

- Accept if and only if $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

As usual it is clear that the verifier W_{S_1,S_2} has perfect completeness (for every S_1 and S_2). We bound the soundness of this verifier, for most choices of sufficiently large sets S_1 and S_2:

Lemma 4.6 *For randomly chosen sets S_1, S_2 of size $O(|Q| \max\{\log |A|, \log |Q|\})$, with probability at least $4/5$, the soundness error of the verifier W_{S_1,S_2} is at most 6ϵ.*

Proof: We start with some notation: Recall that Ω denotes the space of random strings of the verifier V (of Section 4.1.1). For $i \in \{1, 2\}$ and a fixed set S_i, let X_i denote the distribution on Ω induced by picking a random string $r \in \Omega$ uniformly, conditioned on $q_i(r) \in S_i$ (i.e., uniform in $\{r \in \Omega | q_i(r) \in S_i\}$). Similarly, let Y_i denote the distribution on Ω induced by picking a query $q \in Q$ uniformly and then picking r_i uniformly at random from the set $\{r \in \Omega | q(r) = q \wedge q_i(r) \in S_i\}$. We use the notation $r_i \leftarrow D$ to denote that r_i is picked according to distribution D. Note that the verifier W_{S_1,S_2} picks $r_1 \leftarrow Y_1$ and $r_2 \leftarrow Y_2$ (depending on the same random $q \in Q$). In our analysis, we will show that, for a random S_i, the distributions X_i and Y_i are statistically close, where as usual the statistical difference between X_i and Y_i is defined as $\max_{T \subseteq \Omega} \{\Pr_{r_i \leftarrow X_i}[r_i \in T] - \Pr_{r_i \leftarrow Y_i}[r_i \in T]\}$. We will then show that the verifier has low soundness error if it works with the distributions X_1 and X_2. This informal description can be made rigorous by considering the following "bad" events (over the probability space defined by the random choices of S_1 and S_2):

BE1: The statistical difference between X_1 and Y_1 is more than ϵ.

BE2: The statistical difference between X_2 and Y_2 is more than ϵ.

BE3: There exist P and P_1 such that for $Q' = Q'_P$ (as in Property (4) of Section 4.1.1)

$$\Pr_{r_1 \leftarrow X_1} [(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] > 2\epsilon.$$

BE4: There exist P and P_2 such that for $Q' = Q'_P$ (as in Property (4) of Section 4.1.1)

$$\Pr_{r_2 \leftarrow X_2} [(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] > 2\epsilon.$$

In the full version of this paper [14], we show that the probabilities of the bad events are each at most $1/20$. Further we show that if none of the bad events occur, then the soundness error of the verifier W_{S_1,S_2} is at most 6ϵ. ■

4.1.4 The 3-prover MIP: Stage III

Having reduced the sizes of the three prover oracles, it is straightforward to reduce the amount of randomness used by the three provers. Below we describe a reduced randomness verifier $W_{S_1,S_2,T}$ where $S_i \subseteq Q_i$ and $T \subseteq \{(r_1, r_2) | (q(r_1) = q(r_1)) \wedge (q_i(r_i) \in S_i, \forall i \in \{1, 2\})\}$.

- On input x, let $V = V_x$ be the verifier's predicate and let V_1 and V_2 be as given in Property (1).
- Pick $(r_1, r_2) \in T$ uniformly at random. [This uses the sampleability property.]
- Compute $q = q(r_1) = q(r_2)$, and make queries q, $q_1 = q_1(r_1)$ and $q_2 = q_2(r_2)$, to P, P_1 and P_2, receiving answers $a = P(q)$, $a_1 = P_1(q_1)$ and $a_2 = P_2(q_2)$.
- Accept if and only if $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

It is obvious that the verifier uses $\log_2 |T|$ random bits. It is also easy to see that if T is chosen randomly of sufficiently large size then its soundness remains low. We skip this proof, stating the resulting lemma.

Lemma 4.7 *If S_1, S_2 are chosen randomly of size $O(|Q| \max\{\log |A|, \log |Q|\})$ and T is chosen randomly of size $O(|Q| \log |A| + |S_1| \log |A_1| + |S_2| \log |A_2|)$, then, with probability at least $\frac{2}{3}$, the verifier $W_{S_1,S_2,T}$ has soundness error at most 7ϵ.*

Lemma 4.2 follows from Lemma 4.7 in a straightforward manner. See full version [14] for details.

4.2 Implication on PCP

Applying state-of-the-art composition lemmas from [2, 16], leads to the following theorem, where Part 2 implies Theorem 2.3.

Theorem 4.8 (Our main PCP result):

1. *For every $\epsilon > 0$, SAT reduces probabilistically, under $n^{1+O(1/\log\log n)}$-length preserving reductions to a promise problem $\Pi \in \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}[(1 + O(1/\log\log n)) \cdot \log n, 16]$.*
2. *For every $\epsilon > 0$, SAT reduces probabilistically, under $n^{1+O(\sqrt{\log n}\log\log n)}$-length preserving reductions to a promise problem $\Pi \in \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}[(1 + O(\log\log n/\sqrt{\log n})) \cdot \log n, 19]$.*

Part 2 implies Theorem 2.3.

5 Conclusions and Open Problems

Our code constructions are randomized, and so we do not obtain fully-explicit codes. The randomization amounts to selecting a random subspace of random-tapes for certain low-degree tests, and the probabilistic analysis shows that almost all choices of the subspace will do. A natural (derandomization) goal is to provide an explicit construction of a good subspace. For example, in case of the low-degree test, the goal is to provide an explicit set of $\tilde{O}(|F|^m)$ lines that can be used (as R_m in the construction of Section 3.2).

As a seemingly easier goal, consider the linearity test of Blum, Luby and Rubinfeld [7]: To test whether $f: G \to H$ is linear, one uniformly selects $(x, y) \in G \times G$ and accepts if and only if $f(x) + f(y) = f(x + y)$. Now, by the probabilistic method, there exists a set $R \subset G \times G$ of size $O(|G| \log |H|)$ such that the test works well when (x, y) is uniformly selected in R (rather than in $G \times G$).[5] The goal is to present an explicit construction of such a set R. Recent progress on this special case (i.e., derandomization of the BLR test) is reported in [15].

Another natural question that arises in this work refers to obtaining locally-testable codes for coding $k' < k$ information symbols out of codes that apply to k information symbols. The straightforward idea of converting k'-symbol messages into k-symbol messages (via padding) and encoding the latter by the original code, preserves many properties of the code but does not necessarily preserve local-testability.[6]

We have presented locally testable codes and PCP schemes of almost-linear length, where $\ell : \mathsf{N} \to \mathsf{N}$ is called almost-linear if $\ell(n) = n^{1+o(1)}$. For PCP, this improved over a previous result where for each $\epsilon > 0$ a scheme of length $n^{1+\epsilon}$ was presented (with query complexity $O(1/\epsilon)$). Recall that our schemes have length $\ell(n) = \exp(\log n)^c) \cdot n$, for any $c > 0.5$. We wonder whether length $\ell(n) = \text{poly}(\log n) \cdot n$ (or even linear length) can be achieved. Similarly, the number of queries in our proof system is really small, say 16, while simultaneously achieving nearly linear-sized proofs. Further reduction of this query complexity is very much feasible and it is unclear what the final limit may be. Is it possible to achieve nearly-linear (or even linear?) proofs with 3 query bits and soundness nearly $1/2$?

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *JACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.

[2] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *JACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.

[3] S. Arora and M. Sudan. Improved low degree testing and its applications. In *29th STOC*, pages 485–495, 1997.

[5] For every $f: G \to H$, with probability $1 - \exp(-|R|)$ a random set R will be good for testing whether f is linear, and the claim follows using the union bound for all $|H|^{|G|}$ possible functions $f: G \to H$.

[6] Indeed, this difficulty (as well as other difficulties regarding the gap between PCPs and codes) disappears if one allows probabilistic coding. That is, define a code $\mathcal{C}: \Sigma^k \to \Sigma^n$ as a randomized algorithm (rather than a mapping), and state all code properties with respect to randomized codewords $\mathcal{C}(a)$'s.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd STOC*, pages 21–31, 1991.

[5] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *26th STOC*, 1994.

[6] E. Ben-Sasson, O. Goldreich, and M. Sudan. Impossibility results for 2-query codeword testing. In preparation, 2002.

[7] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *JCSS*, Vol. 47, No. 3, pages 549–595, 1993.

[8] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Press, Philadeplhia, PA, USA, March 2001.

[9] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. *JACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.

[10] G.D. Forney. *Concatenated Codes*. MIT Press, Cambridge, MA 1966.

[11] K. Friedl and M. Sudan. Some Improvements to Low-Degree Tests. In the *3rd Israel Symp. on Theory and Computing Systems (ISTCS)*, 1995.

[12] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, pages 653–750, July 1998.

[13] O. Goldreich, H. Karloff, L.J. Schulman and L. Trevisan. Lower Bounds for Linear Locally Decodable Codes and Private Information Retrieval. In the *Proc. of the 17th IEEE Conference on Computational Complexity*, 2002.

[14] O. Goldreich and M. Sudan. Locally Testable Codes and PCPs of Almost-Linear Length. *ECCC Technical Report*, TR02-050, August 2002.

[15] O. Goldreich and A. Wigderson. On derandomizing the BLR test. Private communication, June 2002.

[16] P. Harsha and M. Sudan. Small PCPs with Low Query Complexity. *Computational Complexity*, 9(3-4):157-201, 2000.

[17] J. Katz and L. Trevisan. On The Efficiency Of Local Decoding Procedures For Error-Correcting Codes. In the *32nd STOC*, 2000.

[18] A. Polishchuk and D.A. Spielman. Nearly-linear size holographic proofs. In *26th STOC*, pages 194–203, 1994.

[19] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *29th STOC*, 1997.

[20] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, Vol. 25 (2), pages 252–271, 1996.