Robust PCPs of Proximity, Shorter PCPs and Applications to Coding

Eli Ben-Sasson *
Oded Goldreich †
Prahladh Harsha ‡
Madhu Sudan §
Salil Vadhan ¶

ABSTRACT

We continue the study of the trade-off between the length of PCPs and their query complexity, establishing the following main results (which refer to proofs of satisfiability of circuits of size n):

1. We present PCPs of length $\exp(\tilde{O}(\log\log n)^2) \cdot n$ that can be verified by making $o(\log\log n)$ Boolean queries.

2. For every $\varepsilon > 0$, we present PCPs of length $\exp(\log^{\varepsilon} n) \cdot n$ that can be verified by making a constant number of Boolean queries.

In both cases, false assertions are rejected with constant probability (which may be set to be arbitrarily close to 1). The multiplicative overhead on the length of the proof, introduced by transforming a proof into a probabilistically checkable one, is just quasi-polylogarithmic in the first case (of query complexity $o(\log\log n)$), and $2^{(\log n)^{\varepsilon}}$, for any $\varepsilon > 0$, in the second case (of constant query complexity). In contrast, previous results required at least $2^{\sqrt{\log n}}$ overhead in the length, even to get query complexity $2^{\sqrt{\log n}}$.

Our techniques include the introduction of a new variant of PCPs that we call "Robust PCPs of proximity". These new PCPs facilitate proof composition, which is a central ingredient in construction of PCP systems. (A related notion and its composition properties were discovered independently by Dinur and Reingold.) Our main technical contribution is a construction of a "length-efficient" Robust PCP of proximity. While the new construction uses many of the standard techniques in PCPs, it does differ from previous constructions in fundamental ways, and in particular does not use the "parallelization" step of Arora *et. al.*. The alternative approach may be of independent interest.

We also obtain analogous quantitative results for locally testable codes. In addition, we introduce a relaxed notion of locally decodable codes, and present such codes mapping k information bits to codewords of length $k^{1+\varepsilon}$, for any $\varepsilon > 0$.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems, Complexity of Proof Procedures; E.4 [**Data**]: Coding and Information Theory

General Terms

Theory

Keywords

Probabilistically Checkable Proofs, PCP, Locally Testable Codes, Locally Decodable Codes, Property Testing.

1. INTRODUCTION

Probabilistically Checkable Proofs [13, 2, 1] (a.k.a. Holographic Proofs [3]) are NP witnesses that allow efficient probabilistic verification based on probing few bits of the NP witness. The celebrated PCP Theorem [2, 1] asserts that probing a constant number of bits suffices, and it turned out that three bits suffice for rejecting false assertions with probability almost 1/2 (cf. [21, 19]).

Optimizing the query complexity of PCPs has attracted a lot of attention (see, for example, [5, 4, 21, 19, 27]), but these works only guarantee that the new NP witness (i.e., the PCP) is of length that is upper-bounded by a polynomial

[0] Part of the work was done while the first, second, fourth and fifth authors were fellows at the Radcliffe Institute for Advanced Study of Harvard University.

* Radcliffe Institute for Advanced Study, Cambridge, MA 02139. email: eli@eecs.harvard.edu.

† Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, ISRAEL. email: oded.goldreich@weizmann.ac.il.

‡ Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. email: prahladh@mit.edu. Supported in part by NSF Award CCR-0312575.

§ Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. email: madhu@mit.edu. Supported in part by NSF Award CCR-0312575.

¶ Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138. email: salil@eecs.harvard.edu. Supported in part by NSF grant CCR-0133096 and a Sloan Research Fellowship.

STOC'04, June 13–15, 2004, Chicago, Illinois, USA.

in the length of the original NP witness.[1] The length of the new NP witness was the focus of [3, 24, 20, 18, 8], and in this work we continue the latter research direction.

How short can a PCP be? The answer may depend on the number of bits we are willing to read in order to reject false assertions (say) with probability at least 1/2. It is implicit in the work of Polishchuk and Spielman [24] that, for proofs of satisfiability of circuits of size n, if we are willing to read $n^{0.01}$ bits then the length of the new NP witness may be $\tilde{O}(n)$. That is, stretching the NP witness by only a poly-logarithmic amount, allows to dramatically reduce the number of bits read (from n to $n^{0.01}$). More precisely:[2]

THEOREM 1.1. (implicit in [24]) *Satisfiability of circuits of size n can be probabilistically verified by probing an NP witness of length* $\mathrm{poly}(\log n)\cdot n$ *in* $n^{o(1)}$ *bit locations. In fact, for any value of a parameter* $m \leq \log n$, *there is a PCP having randomness complexity* $(1-m^{-1})\log_2 n + O(\log\log n) + O(m\log m)$ *and query complexity* $O(\mathrm{poly}(\log n)\cdot n^{1/m})$.

Recall that the proof length of a PCP is at most $2^r \cdot q$, where r is the randomness complexity and q is the query complexity of the PCP. Thus, the first part of the above theorem follows by setting $m = \log\log n/\log\log\log n$ in the second part.

Our results show that the query complexity can be reduced dramatically if we are willing to increase the length of the proof slightly. First, with a quasi-polylogarithmic stretch, the query complexity can be made double-logarithmic:

THEOREM 1.2. *Satisfiability of circuits of size n can be probabilistically verified by probing an NP witness of length* $\exp(\tilde{O}(\log\log n)^2)\cdot n$ *in* $o(\log\log n)$ *bit-locations. In fact, it has a PCP having randomness complexity* $\log_2 n + (\log\log n)^2\cdot \mathrm{poly}(\log\log\log n)$ *and query complexity* $O(\log\log n/\log\log\log n)$.

Prior to our work, reducing the query complexity below $\exp(\sqrt{\log n})$ required stretching the NP witness by at least a $\exp(\sqrt{\log n})$ factor. With approximately such a stretch factor, previous works actually achieved constant query complexity (cf. [18, 8]). Thus, Theorem 1.2 represents a vast improvement in the query complexity of PCPs that use very short proofs (i.e., in the range between $\exp(\tilde{O}(\log\log n)^2)\cdot n$ and $\exp(\sqrt{\log n})\cdot n$). On the other hand, considering NP witnesses that allow probabilistic verification by a *constant* number of queries, we reduce the best known stretch factor from $\exp(\log^{0.5+\varepsilon} n)$ (established in [18, 8]) to $\exp(\log^{\varepsilon} n)$, for any $\varepsilon > 0$. That is:

THEOREM 1.3. *For every constant $\varepsilon > 0$, satisfiability of circuits of size n can be probabilistically verified by probing an NP witness of length* $\exp(\log^{\varepsilon} n)\cdot n$ *in a constant number of bit-locations. In fact, it has a PCP having randomness complexity* $\log_2 n + \log^{\varepsilon} n$ *and query complexity* $O(1/\varepsilon)$.

It may indeed be the case that the trade-off (between length blow-up factors and query complexity) offered by Theorems 1.1–1.3 merely reflects our (incomplete) state of knowledge. In particular, we wonder whether circuit satisfiability can be probabilistically verified by a PCP having proof-length $n\cdot\mathrm{poly}(\log n)$ and constant query complexity.

1.1 New notions and main techniques

A natural approach to reducing the query complexity in Theorem 1.1 is via the "proof composition" paradigm of [2]. However, that PCP, as constructed in [24], does not seem amenable to composition (when the parameter m is non-constant). Thus, we begin by giving a new PCP construction whose parameters match those in Theorem 1.1, but is suitable for composition. As we will see, we cannot afford the standard proof composition techniques, and thus also introduce a new, more efficient composition paradigm.

The initial PCP. Our new proof of Theorem 1.1 modifies the constructions of Polishchuk and Spielman [24] and Harsha and Sudan [20]. The latter construction was already improved in [18, 8] to reduce the length of PCPs to $n\cdot 2^{\tilde{O}(\sqrt{\log n})}$. Our results go further by re-examining the "low-degree test" (query-efficient tests that verify if a given function is close to being a low-degree polynomial) and first observing that the small-bias sample sets of [8] give an even more significant savings on the randomness complexity of low-degree tests than noticed in their work. However, exploiting this advantage takes a significant effort in modifying known PCP modules, and redefining the ingredients in "proof composition".

For starters, PCP constructions tend to use many (i.e., a super-constant number of) functions and need to test if each is a low-degree polynomial. In prior results, this was performed efficiently by combining the many different functions on, say m variables, into a single new one on $m+1$ variables, where the extra variable provides an index into the many different old functions. Testing if the new function is of low-degree, implicitly tests all the old functions. Such tricks, which involve introducing a few extra variables, turn out to be too expensive in our context. Furthermore, for similar reasons, we can not use other "parallelization" techniques [14, 23, 1, 17, 25], which were instrumental to the proof composition technique of [2]. In turn, this forces us to introduce a new variant of the proof composition method, which is much more flexible than the one of [2]. Going back to the PCP derived in Theorem 1.1, we adapt it for our new composition method by introducing a "bundling" technique that offers a randomness efficient alternative to parallelization.

Our new "proof composition" method refers to two new notions: the notion of a *PCP of proximity* and the notion of a *robust PCP*. Our method is related to the method discovered independently by Dinur and Reingold [11]. (There are significant differences between the two methods; as explained in Section 1.2.)

PCPs of Proximity. Recall that a standard PCP is given an explicit input (which is supposedly in some NP language) as well as access to an oracle that is supposed to encode a "probabilistically verifiable" NP witness. The PCP verifier uses oracle queries (which are counted) in order to probabilistically verify whether the input, which is explicitly given to it, is in the language. In contrast, a PCP of proximity is given access to two oracles, one representing an input (supposedly in the language) and the other being a redundant encoding of an NP-witness (as in a PCP). Indeed,

[1] We stress that in all the above works as well as in the current work, the new NP witness can be computed in polynomial-time from the original NP witness.

[2] All logarithms in this work are to based 2, but in some places we choose to emphasize this fact by using the notation $\log_2$ rather than log.

the verifier may query both the input oracle and the proof oracle, but *its queries to the input oracle are also counted in its query complexity.* As usual we focus on verifiers having very low query complexity, certainly smaller than the length of the input. Needless to say, such a constrained verifier cannot hope to distinguish inputs in the language from inputs out of the language, but it is not required to do so. A verifier for a PCP of proximity is only required to accept inputs that are in the language and reject inputs that are *far* from the language (i.e., far in Hamming distance from any input in the language). (PCPs of proximity are related to holographic proofs [3] and to "PCP spot-checkers" [12]; see further discussion in Section 1.2.)

Robust PCPs. To discuss robust PCPs, let us recall the soundness guarantee of standard (non-adaptive) PCPs. The corresponding verifier can be thought of as determining, based on its coin tosses, a sequence of oracle positions and a predicate such that evaluating this predicate on the indicated oracle bits always accepts if the input is in the language and rejects with high probability otherwise. That is, in the latter case, we require that the assignment of oracle bits to the predicate does satisfy the predicate. In a robust PCP we strengthen the latter requirement. We require that the said assignment (of oracle bits) not only fails to satisfy the predicate but rather is *far* from any assignment that does satisfy the predicate.

Proof Composition. The key observation is that "proof composition" works very smoothly when we compose an outer "robust PCP" with an inner "PCP of proximity". We need neither worry about how many queries the outer "robust PCP" makes nor care about what coding the inner "PCP of proximity" uses in its proof oracle (much less apply the same encoding to the outer answers). All that we should make sure is that the lengths of the objects match and that the distance parameter in the robustness condition (of the outer verifier) is at least as big as the distance parameter in the proximity condition (of the inner verifier).

Indeed, Theorems 1.2 and 1.3 are proved by first extending Theorem 1.1 to provide a robust PCP of proximity of similar complexities, and then applying the new "proof composition" method. We stress that our contribution is in providing a proof of Theorem 1.1 that lends itself to a modification that satisfies the robustness property, and in establishing the latter property. In particular, the aforementioned "bundling" is applied in order to establish the robustness property. Some care is also due when deriving Theorem 1.2 using a non-constant number of "proof compositions". In particular, Theorem 1.2 (resp., Theorem 1.3) is derived in a way that guarantees that the query complexity is linear rather than exponential in the number of "proof compositions", where the latter is $o(\log\log n)$ (resp., $1/\varepsilon$).

We stress that the flexibility in composing robust PCPs of proximity plays an important role in our ability to derive quantitatively stronger results regarding PCPs. We believe that robust PCPs of proximity may play a similar role in other quantitative studies of PCPs. We note that the standard PCP Theorem of [2, 1] can be easily derived using a much weaker and simpler variant of our basic robust PCP of proximity, and the said construction seems easier than the basic PCPs used in the proof composition of [2, 1].

In addition to their role in our "proof composition" method, PCPs of proximity provide also a good starting point for deriving improved locally testable codes (see discussion in Section 1.3). The relation of PCPs of proximity to "property testing" is further discussed in Section 1.4.

1.2 Related work

As mentioned above, the notion of a PCP of proximity is related to notions that have appeared in the literature. Firstly, the notion of a PCP of proximity generalizes the notion of holographic proofs set forward by Babai, Fortnow, Levin, and Szegedy [3]. In both cases, the verifier is given oracle access to the input, and we count its probes to the input in its query complexity. The key issue is that holographic proofs refer to inputs that are presented in an error-correcting format (e.g., one aims to verify that a graph that is *represented by an error-correcting encoding of its adjacency matrix* (or incidence list) is 3-colorable). In contrast, a PCP of proximity refers to inputs that are presented in any format but makes assertions only about their proximity to acceptable inputs (e.g., one is interested in whether a graph, represented by its adjacency matrix (or incidence list), *is 3-colorable or is far from being 3-colorable*).

PCP of proximity are implicit in the low-degree testers that utilize auxiliary oracles (e.g., an oracle that provides the polynomial representing the value of the function restricted to a queried line); cf. [2, 1].

PCPs of proximity are a special case of the "PCP spot-checkers" defined by Ergün, Kumar and Rubinfeld [12]. On the other hand, PCPs of proximity extend "property testing" [26, 15] by providing the tester with oracle access to a proof (on top of the ordinary input-oracle to which it has access). Thus, the relation of PCPs of proximity to property testing is analogous to the relation of NP to BPP (or RP). Put differently, while property testing provides a notion of approximation for decision procedures, PCP of proximity provides a notion of approximation for (probabilistic) verification procedures. In both cases, approximation means that inputs in the language should be accepted (when accompanied with suitable proofs) while inputs that are far from the language should be rejected (no matter what false proof is provided).

As stated above, our "proof composition" method is related to the method discovered independently by Dinur and Reingold [11]. Both methods use the same notion of PCPs of proximity, but while our method refers to the new notion of robustness (i.e., to the robustness of the outer verifier) the method of Dinur and Reingold refers to the number of (non-Boolean) queries (made by the outer verifier). Indeed, the method of Dinur and Reingold uses a (new) parallelization procedure (which reduces the number of queries by a constant factor), whereas we avoid parallelization altogether (but rather use a related "bundling" of queries into a non-constant number of "bundles" such that robustness is satisfied at the bundle-level).[3] We stress that we cannot afford

[3]The main part of the bundling technique takes place at the level of analysis, without modifying the proof system at all. Specifically, we show that the answers read by the verifier can be partitioned into a non-constant number of (a-priori fixed) "bundles" so that on any no instance, with high probability a constant fraction of the bundles read should be modified to make the verifier accept. We stress that the fact that certain sets of queries (namely those in each bundle) are always made together is a feature that our particular proof system happens to have (or rather it was somewhat massaged to have). Once "robust soundness" is established at the "bundle level," we may just modify the proof system so that the bundles become queries and the answers are placed in

the cost of any known parallelization procedure, because at the very least these procedures increase the length of the proof by a factor related to the answer length, which is far too large in the context of Theorem 1.1 (which in turn serves as the starting point for all the other results in this work). We comment that the parallelization procedure of [11] is combinatorial (albeit inapplicable in our context), whereas our "bundling" relies on the algebraic structure of our proof system.

1.3 Applications to coding problems

The flexibility of PCPs of proximity makes them relatively easy to use towards obtaining results regarding locally testable and decodable codes. In particular, using a suitable PCP of proximity, we obtain an improvement in the rate of locally testable codes (improving over the results of [18, 8]). Loosely speaking, a codeword test (for a code C) is a randomized oracle machine that is given oracle access to a string. The tester may query the oracle at a constant number of bit-locations and is required to (always) accept every codeword and reject with (relatively) high probability every string that is "far" from the code. The locally testable codes of [18, 8] used codewords of length $\exp(\log^{0.5+\varepsilon} k)\cdot k$ in order to encode k bits of information, for any constant $\varepsilon > 0$. Here we reduce the length of the codewords to $\exp(\log^{\varepsilon} k)\cdot k$. That is:

THEOREM 1.4. (loosely stated): *For every constant $\varepsilon > 0$, there exists locally testable codes that use codewords of length* $\exp(\log^{\varepsilon} k)\cdot k$ *in order to encode k bits of information.*

We also introduce a relaxed notion of locally decodable codes, and show how to construct such codes using any PCP of proximity (and ours in particular). Loosely speaking, a code is said to be locally decodable if whenever relatively few location are corrupted, the decoder is able to recover each information-bit, with high probability, based on a constant number of queries to the (corrupted) codeword. This notion was formally defined by Katz and Trevisan [22] and the best known locally decodable code has codeword of length that is sub-exponential in the number of information bits. We relax the definition of locally decodable codes by requiring that, whenever few location are corrupted, the decoder should be able to recover most of the individual information-bits (based on few queries) and for the rest of the locations, the decoder may output a fail symbol (but not the wrong value). That is, the decoder must still avoid errors (with high probability), but is allowed to say "`don't know`" on a few bit-locations. We show that this relaxed notion of local decodability can be supported by codes that have codewords of length that is almost-linear in the number of information bits. That is:

THEOREM 1.5. (loosely stated): *For every $\varepsilon > 0$, there exists relaxed locally decodable codes that use codewords of length $k^{1+\varepsilon}$ in order to encode k bits of information.*

1.4 Relation to Property Testing

Following Ergün *et. al.* [12], we view PCPs of proximity as an extension of property testing [26, 15]. Loosely speaking, a property tester is given oracle access to an input and is required to distinguish the case in which the input has the property from the case in which it is far (say in Hamming distance) from any input having the property. Typically, the interest is in testers that query their input on few bit-locations (or at the very least on a sub-linear number of such locations). In a PCP of proximity such a tester (now called a verifier) is also given oracle access to an alleged proof.

We comment that PCPs of proximity are provably stronger than property testers; that is, there are (natural) separations between property testers and PCPs of proximity (which may be viewed as the "approximation" versions of BPP and NP). For further discussions, refer to Section 2.2

(any) good error-correcting format, which implies robustness at the bit level.

1.5 Organization

Theorems 1.2 and 1.3, which are the work's main results, are proved by constructing and using a Robust PCP of Proximity that achieves a very good trade-off between randomness and query complexity. Thus, this Robust PCP of Proximity is the main building block that underlies our work. Unfortunately, the construction of a very efficient Robust PCP of Proximity is quite involved, and is thus deferred to the full version of the paper [6]. In Section 2 we provide a basic definitional treatment of PCPs of proximity and robust PCPS. The basic definitions as well as some observations and useful transformations are presented in Section 2. Most importantly, we analyze the natural composition of an outer robust PCP with an inner PCP of proximity. In Section 3 we provide an overview of our main construct, a Robust PCP of Proximity that achieves a very good trade-off between randomness and query complexity (whose construction is deferred to the full-version of the paper). This construction is then composed with itself to derive Theorems 1.2 and 1.3.

2. PCPS AND VARIANTS: DEFINITIONS AND COMPOSITION

Notation: The *size* of a circuit is the number of gates. We will refer to the following languages associated with circuits: the **P**-complete language CIRCUIT VALUE, defined as $\text{CKTVAL} = \{(C, w) : C(w) = 1\}$ and the **NP**-complete CIRCUIT SATISFIABILITY, defined as $\text{CKTSAT} = \{C : \exists w C(w) = 1\}$.

We will extensively refer to the *relative* distance between strings/sequences over some alphabet Σ: For $u, v \in \Sigma^{\ell}$, we denote by $\Delta(u, v)$ the fraction of locations on which u and v differ (i.e., $\Delta(u, v) \triangleq |\{i : u_i \neq v_i\}|/\ell$, where $u = u_1 \cdots u_\ell \in \Sigma^{\ell}$ and $v = v_1 \cdots v_\ell \in \Sigma^{\ell}$). We say that u is *δ-close* to v (resp., *δ-far* from v) if $\Delta(u, v) \leq \delta$ (resp., $\Delta(u, v) > \delta$). The relative distance of a string to a set of strings is defined in the natural manner; that is, $\Delta(u, S) \triangleq \min_{v\in S}\{\Delta(u, v)\}$.

Organization of this section: After recalling the standard definition of PCP (in Section 2.1), we present the definitions of PCPs of Proximity and Robust PCPs (in Sections 2.2 and 2.3, respectively). We then turn to discuss (in Section 2.4) the composition of a Robust PCP with a PCP of Proximity.

2.1 Standard PCPs

We begin by recalling the formalism of a PCP verifier. Throughout this work, we restrict our attention to *non-adaptive* verifiers, both for simplicity and because one of our variants (namely robust PCPs) only makes sense for nonadaptive verifiers.

DEFINITION 2.1 (PCP VERIFIERS).

- *A* verifier *is a* probabilistic polynomial-time *algorithm V that, on an input x of length n, tosses $r = r(n)$ random coins R and generates* a sequence *of $q = q(n)$ queries $I = (i_1, \ldots, i_q)$ and a circuit $D : \{0,1\}^q \to \{0,1\}$ of size at most $d(n)$.*

  We think of V as representing a probabilistic oracle machine that queries its oracle π for the positions in I, receives the q answer bits $\pi|_I \triangleq (\pi_{i_1}, \ldots, \pi_{i_q})$, and accepts iff $D(\pi|_I) = 1$.

- *We write $(I, D) \stackrel{R}{\leftarrow} V(x)$ to denote the queries and circuit generated by V on input x and random coin tosses, and $(I, D) = V(x; R)$ if we wish to specify the coin tosses R.*

- *We call r the* randomness complexity*, q the* query complexity*, and d the* decision complexity *of V.*

For simplicity in these definitions, we treat the parameters r, q, and d above (and other parameters below) as functions of only the input length n. However, at times we may also allow them to depend on other parameters, which should be understood as being given to the verifier together with the input. We now present the standard notion of PCPs, restricted to perfect completeness for simplicity.

DEFINITION 2.2 (STANDARD PCPS). *For a function $s : \mathbb{Z}^+ \to [0,1]$, a verifier V is a* probabilistically checkable proof system *for a language L with* soundness error *s if the following two conditions hold for every string x:*

Completeness: *If $x \in L$ then there exists π such that $V(x)$ accepts oracle π with probability 1. Formally,*

$$\exists \pi \qquad \Pr_{(I,D) \stackrel{R}{\leftarrow} V(x)} [D(\pi|_I) = 1] = 1.$$

Soundness: *If $x \notin L$ then for every oracle π, the verifier $V(x)$ accepts π with probability strictly less than s. Formally,*

$$\forall \pi \qquad \Pr_{(I,D) \stackrel{R}{\leftarrow} V(x)} [D(\pi|_I) = 1] < s(|x|).$$

If s is not specified, then it is assumed to be a constant in $(0,1)$.

Our main goal in this work is to construct *short* PCPs that use *very few queries*. Recalling that the length of a (nonadaptive) PCP is upper-bounded by $2^{r(n)} \cdot q(n)$, we focus on optimizing the (trade-off between) randomness and query complexities.

2.2 PCPs of Proximity

We now present a relaxation of PCPs that only verify that the input is *close* to an element of the language. The advantage of this relaxation is that it allows the possibility that the verifier may read only a small number of bits from the input. Actually, for greater generality, we will divide the input into two parts (x, y), giving the verifier x explicitly and y as an oracle, and we only count the verifier's queries to the latter. Thus we consider languages consisting of pairs of strings, which we refer to as a pair language. One pair language to keep in mind is the CIRCUIT VALUE problem: CKTVAL $= \{(C, w) : C(w) = 1\}$. For a pair language L, we define $L(x) = \{y : (x,y) \in L\}$. For example, CKTVAL$(C)$ is the set of satisfying assignments to C. It will be useful below to treat the two oracles to which the verifier has access as a single oracle, thus for oracles π^0 and π^1, we define the concatenated oracle $\pi = \pi^0 \circ \pi^1$ as $\pi_{b,i} = \pi_i^b$.

DEFINITION 2.3 (PCPS OF PROXIMITY (PCPPS)). *For functions $s, \delta : \mathbb{Z}^+ \to [0,1]$, a verifier V is a* probabilistically checkable proof of proximity *(PCPP) system for a pair language L with* proximity parameter *δ and* soundness error *s if the following two conditions hold for every pair of strings (x, y):*

Completeness: *If $(x, y) \in L$, then there exists π such that $V(x)$ accepts oracle $y \circ \pi$ with probability 1. Formally,*

$$\exists \pi \qquad \Pr_{(I,D) \stackrel{R}{\leftarrow} V(x)} [D((y \circ \pi)|_I) = 1] = 1.$$

Soundness: *If y is $\delta(|x|)$-far from $L(x)$, then for every π, the verifier $V(x)$ accepts oracle $y \circ \pi$ with probability strictly less than $s(|x|)$. Formally,*

$$\forall \pi \qquad \Pr_{(I,D) \stackrel{R}{\leftarrow} V(x)} [D((y \circ \pi)|_I) = 1] < s(|x|).$$

If s and δ are not specified, then both are assumed to be constants in $(0,1)$.

Note that the parameters (soundness, randomness, etc.) of a PCPP are measured as a function of the length of x, the explicit portion of the input.

In comparing PCPPs and PCPs, one should note two differences that have conflicting effects. On one hand, the soundness criterion of PCPPs is a relaxation of the soundness of PCPs. Whereas, a PCP is required to reject (with high probability) every input that is not in the language, a PCPP is only required to reject input pairs (x, y) in which the second element (i.e., y) is far from being suitable for the first element (i.e., y is far from $L(x)$). That is, in a PCPP, nothing is required in the case that y is close to $L(x)$ and yet $y \notin L(x)$. On the other hand, the query complexity of a PCPP is measured more stringently, as it accounts also for the queries to the input-part y (on top of the standard queries to the proof π). This should be contrasted with a standard PCP that has free access to all its input, and is only charged for access to an auxiliary proof. To summarize, PCPPs are required to do less (i.e., their performance requirements are more relaxed), but they are charged for more things (i.e., their complexity is evaluated more stringently). Although it may not be a priori clear, the stringent complexity requirement prevails. That is, PCPPs tend to be more difficult to construct than PCPs of the same parameters. For example, while CIRCUIT VALUE has a trivial PCP (since it is in **P**), a PCPP for it implies a PCP for CIRCUIT SATISFIABILITY:

PROPOSITION 2.4. *If* CIRCUIT VALUE *has a PCPP, then* CIRCUIT SATISFIABILITY *has a PCP with identical parameters (randomness, query complexity, decision complexity, and soundness).*

An analogous statement holds for any pair language L and the corresponding projection on first element $L_1 \triangleq \{x : \exists y \text{ s.t. } (x,y) \in L\}$; that is, if L has a PCPP then L_1 has a PCP with identical parameters.

PROOF. A PCP π that C is satisfiable can be taken to be $w \circ \pi'$, where w is a satisfying assignment to C and π' is a PCPP that $(C, w) \in$ CKTVAL. This proof π can be verified using the PCPP verifier. The key observation is that if $C \notin$ CIRCUIT SATISFIABILITY then there exists no w that is 1-close to CIRCUIT VALUE(C), because the latter set is empty. $\square$

Note that we only obtain a standard PCP for CIRCUIT SATISFIABILITY, rather than a PCP of proximity. Indeed, CIRCUIT SATISFIABILITY is not a pair language, so it does not even fit syntactically into the definition of a PCPP.

Relation to property testing: Actually, the requirements from a PCPP for a pair language L refer only to its performance on the ("gap") promise problem $\Pi = (\Pi_Y, \Pi_N)$, where $\Pi_Y = L$ and $\Pi_N = \{(x, y) : y \text{ is } \delta\text{-far from } L(x)\}$. That is, this PCPP is only required to (always) accept inputs in Π_Y and reject (with high probability) inputs in Π_N (whereas nothing is required with respect to inputs not in $\Pi_Y \cup \Pi_N$). Such a gap problem corresponds to the notion of approximation in *property testing* [26, 15].[4] Indeed, property testers are equivalent to PCPP verifiers that have no access to an auxiliary proof π. Thus the relation between property testing and PCPPs is analogous to the relation between **BPP** and **NP** (or **MA**). For example, the problem of testing Bipartiteness can be cast by the pair language $L = \{(n, G) : \text{the } n\text{-vertex graph } G \text{ is bipartite}\}$, where the first (i.e., explicit) input is only used to specify the length of the second (i.e., non-explicit) input G, to which the tester has oracle access (measured in its query complexity). We comment that the formulation of pair languages allows to capture more general property testing problems where more information about the property (to be tested) itself is specified as part of the input (e.g., by a circuit, as in CKTVAL).

In both property testers and PCPs of proximity, the interest is in testers/verifiers that query their input (and proof oracle) in only a small (preferably constant, and certainly sublinear) number of bit-locations. It turns out that PCPPs are provably stronger than property testers; that is, there are (natural) separations between property testers and PCPs of proximity. (Some of the following examples were pointed out in [12].) In the adjacency matrix model (cf. [15]), Bipartiteness has a PCP of proximity in which the verifier makes only $O(1/\delta)$ queries and rejects any graph that is δ-far from being bipartite with probability at least $2/3$. (The proof-oracle consists of an assignment of vertices to the two parts, and the verifier queries the assignment of the end-points of $O(1/\delta)$ random edges. This construction also generalizes to k-colorability, and in fact any generalized graph partition property (cf. [15]) with an efficient one-sided tester.) In contrast, Bogdanov and Trevisan [10] showed that any tester for Bipartiteness that rejects graphs that are δ-far from being bipartite must make $\Omega(\delta^{-3/2})$ queries. More drastic separations are known in in the incidence-lists (bounded-degree) model (of [16]): testing Bipartiteness (resp., 3-colorability) of n-vertex graphs has query complexity $\Omega(\sqrt{n})$ [16] (resp., $\Omega(n)$ [9]), but again a PCP of proximity will only use $O(1/\delta)$ queries.

Another example comes from the domain of codes. For any good code (or "even" any code of linear distance), there exists a PCP of proximity for the property of being a codeword that makes a constant number of queries.[5] This stands in contrast to the linear lower-bound on the query-complexity of codeword testing for some (good) linear codes, proved by Ben-Sasson *et. al.* [7].

Needless to say, there may be interesting cases in which PCPs of proximity do not out-perform property testers.

The relation of PCPP to other works: As discussed in the introduction (see Section 1.2), notions related to (and equivalent to) PCPPs have appeared in the literature before [3, 12]. In particular, holographic proofs are a special case of PCPPs (which refer to pair languages $L = \{(n, \mathcal{C}(x)) : x \in L' \cap \{0,1\}^n\}$, where $\mathcal{C}$ is an error-correcting code and $L' \in \mathbf{NP}$), whereas PCPPs are a special case of "PCP spot-checkers" (when viewing decision problems as a special case of search problems). In addition, PCPPs play an important role also in the work of Dinur and Reingold [11]; again, see Section 1.2. Recall that both our use and their use of PCPPs is for facilitating "proof composition" (of PCP-type constructs). Finally, existing PCP constructions (such as [1]) can be modified to yield PCPPs.

2.3 Robust Soundness

In this section, we present a *strengthening* of the standard PCP soundness condition. Instead of asking that the bits that the verifier reads from the oracle are merely rejected with high probability, we ask that the bits that the verifier reads are *far* from being accepted with high probability. The main motivation for this notion is that, in conjunction with PCPPs, it allows for a very simple composition without the usual costs of "parallelization".

DEFINITION 2.5 (ROBUST SOUNDNESS). *For functions $s, \rho : \mathbb{Z}^+ \to [0,1]$, a PCP verifier V for a language L has* robust-soundness error *s with* robustness parameter *ρ if the following holds for every $x \notin L$: For every oracle π, the bits read by the verifier V are ρ-close to being accepted with probability strictly less than s. Formally,*

$$\forall \pi \quad \Pr_{(I,D) \xleftarrow{R} V(x)} [\exists a \text{ s.t. } D(a) = 1 \text{ and } \Delta(a, \pi|_I) \le \rho] < s(|x|).$$

If s and ρ are not specified, then they are assumed to be constants in $(0,1)$. PCPPs with robust-soundness are defined analogously, with the $\pi|_I$ being replaced by $(y \circ \pi)|_I$.

Note that for PCPs with query complexity q, robust-soundness with any robustness parameter $\rho < 1/q$ is equivalent to standard PCP soundness. However, there can be robust PCPs with large query complexity (e.g. $q = n^{\Omega(1)}$) yet constant robustness, and indeed such robust PCPs will be the main building block for our construction.

Various observations regarding robust PCPs are presented in Section 2.5 of our technical report [6]. We briefly mention here the relation of robustness to parallelization; specifically, when applied to a robust PCP, the simple query-reduction technique of Fortnow *et. al.* [14] performs less poorly than usual (i.e., the resulting soundness is determined by the robustness parameter rather than by the number of queries).

[4]This notion of approximation (of decision problems) should not be confused with the approximation of (search) optimization problems, which is also closely related to PCPs [13, 1].

[5]Indeed, this is a special case of our extension of the result of Babai *et. al.* [3], discussed in Section 1.2. On the other hand, this result is simpler than the locally testable code mentioned in Section 1.3, because here the PCP of proximity is not part of the codeword.

2.4 Composition

As promised, a robust "outer" PCP composes very easily with an "inner" PCPPs. Loosely speaking, we can compose such schemes provided that the decision complexity of the outer verifier matches the input length of the inner verifier, and soundness holds provided that the robustness parameter of the outer verifier upper-bounds the proximity parameter of the inner verifier. Note that composition does not refer to the query complexity of the outer verifier, which is always upper-bounded by its decision complexity.

THEOREM 2.6 (COMPOSITION THEOREM). *Suppose that for functions* $r_{\text{out}}, r_{\text{in}}, d_{\text{out}}, d_{\text{in}}, q_{\text{in}} : \mathbb{N} \to \mathbb{N}$, *and* $\varepsilon_{\text{out}}, \varepsilon_{\text{in}}, \rho_{\text{out}}, \delta_{\text{in}} : \mathbb{N} \to [0,1]$, *the following hold:*

- *Language* L *has a* robust PCP *verifier* V_{out} *with randomness complexity* r_{out}, *decision complexity* d_{out}, *robust-soundness error* $1-\varepsilon_{\text{out}}$, *and robustness parameter* ρ_{out}.
- CIRCUIT VALUE *has a* PCPP *verifier* V_{in} *with randomness complexity* r_{in}, *query complexity* q_{in}, *decision complexity* d_{in}, *proximity parameter* δ_{in}, *and soundness error* $1-\varepsilon_{\text{in}}$.
- $\delta_{\text{in}}(d_{\text{out}}(n)) \le \rho_{\text{out}}(n)$, *for every* n.

Then, L *has a* (standard) *PCP, denoted* V_{comp}, *with*

- *randomness complexity* $r_{\text{out}}(n) + r_{\text{in}}(d_{\text{out}}(n))$,
- *query complexity* $q_{\text{in}}(d_{\text{out}}(n))$,
- *decision complexity* $d_{\text{in}}(d_{\text{out}}(n))$, *and*
- *soundness error* $1-\varepsilon_{\text{out}}(n)\cdot\varepsilon_{\text{in}}(d_{\text{out}}(n))$.

Furthermore, the computation of V_{comp} *(i.e. evaluating* $(I,D) \leftarrow V_{\text{comp}}(x;R)$*) can be performed by some universal algorithm with black-box access to* V_{out} *and* V_{in}. *On inputs of length* n, *this algorithm runs in time* n^c *for some universal constant* c, *with one call to* V_{out} *on an input of length* n *and one call to* V_{in} *on an input of length* $d_{\text{out}}(n)$. *In addition:*

- *If* (instead of being a PCP) *the verifier* V_{out} *is a PCPP with proximity parameter* $\delta_{\text{out}}(n)$ *then* V_{comp} *is a PCPP with proximity parameter* $\delta_{\text{out}}(n)$.
- *If* V_{in} *has robust-soundness with robustness parameter* $\rho_{\text{in}}(n)$, *then* V_{comp} *has robust-soundness with robustness parameter* $\rho_{\text{in}}(d_{\text{out}}(n))$.

PROOF. We will use the inner PCPP to verify that the oracle positions selected by the (robust) outer-verifier are close to being accepted by the outer-verifier's decision circuit. Thus, the new proof will consist of a proof for the outer verifier as well as proofs for the inner verifier, where each of the latter corresponds to a possible setting of the outer verifier's coin tosses (and is intended to prove that the bits that should have been read by the outer-verifier satisfy its decision circuit). We will index the positions of the new (combined) oracle by pairs such that (out, i) denotes the i'th position in the part of the oracle that represents the outer-verifier's proof oracle, and (R, j) denotes the j'th position in the R'th auxiliary block (which represents the R-th possible proof oracle (for the inner verifier's), which in turn is associated with the outer-verifier's coins $R \in \{0,1\}^{r_{\text{out}}}$). For notational convenience, we drop the input length n from the notation below; all parameters of V_{out} are with respect to length n and all parameters of V_{in} with respect to length $d_{\text{out}}(n)$. With these conventions, here is the description of the composed verifier, $V_{\text{comp}}(x)$:

1. Choose $R \xleftarrow{\text{R}} \{0,1\}^{r_{\text{out}}}$.
2. Run $V_{\text{out}}(x;R)$ to obtain $I_{\text{out}} = (i_1, \ldots, i_{q_{\text{out}}})$ and D_{out}.
3. Run $V_{\text{in}}(D_{\text{out}})$ (on random coin tosses) to obtain $I_{\text{in}} = ((b_1, j_1), \ldots, (b_{q_{\text{in}}}, j_{q_{\text{in}}}))$ and D_{in}.

   (Recall that V_{in}, as a PCPP verifier, expects two oracles, an input oracle and a proof oracle, and thus makes queries of the form (b,j), where $b \in \{0,1\}$ indicates which oracle it wishes to query.)
4. For each $\ell = 1, \ldots, q_{\text{in}}$, determine the queries of the composed verifier:
   (a) If $b_\ell = 0$, set $k_\ell = (\text{out}, i_{j_\ell})$; that is, V_{in}'s queries to its input oracle are directed to the corresponding locations in V_{out}'s proof oracle. Recall that the j-th bit in V_{in}'s input oracle is the j-th bit in the input to D_{out}, which in turn is the i_j-th bit in the proof oracle of V_{out}.
   (b) If $b_\ell = 1$, set $k_\ell = (R, j_\ell)$; that is, V_{in}'s queries to its R'th possible proof oracle are directed to the corresponding locations in the auxiliary proof. Recall that the j-th bit in the proof oracle that V_{in} is using to verify the claim referring to the outer-verifier coins R is the j-th bit in the R-th block of the auxiliary proof.
5. Output $I_{\text{comp}} = (k_1, \ldots, k_{q_{\text{in}}})$ and D_{in}.

The claims about V_{comp}'s randomness, query, decision, and computational complexities can be verified by inspection. Thus we proceed to check completeness and soundness.

Suppose that $x \in L$. Then, by completeness of the outer verifier, there exists a proof π_{out} making V_{out} accept with probability 1. In other words, for every $R \in \{0,1\}^{r_{\text{out}}}$, if we set $(I_{\text{out}}, D_{\text{out}}) = V_{\text{out}}(x;R)$, we have $D_{\text{out}}(\pi_{\text{out}}|_{I_{\text{out}}}) = 1$. By completeness of the inner verifier, there exists a proof π_R such that $V_{\text{in}}(D_{\text{out}})$ accepts the oracle $\pi_{\text{out}}|_{I_{\text{out}}} \circ \pi_R$ with probability 1. If we set $\pi(t,\cdot) = \pi_t(\cdot)$ for all $t \in \{\text{out}\} \cup \{0,1\}^{r_{\text{out}}}$, then V_{comp} accepts π with probability 1.

Suppose that $x \notin L$, and let π be any oracle. Define oracles $\pi_t(\cdot) = \pi(t,\cdot)$. By the robust-soundness (of V_{out}), with probability greater than ε_{out} over the choices of $R \in \{0,1\}^{r_{\text{out}}}$, if we set $(I_{\text{out}}, D_{\text{out}}) = V_{\text{out}}(x;R)$, then $\pi_{\text{out}}|_{I_{\text{out}}}$ is ρ_{out}-far from satisfying D_{out}. Fixing such an R, by the PCPP-soundness of V_{in} (and $\delta_{\text{in}} \le \rho_{\text{out}}$), it holds that $V_{\text{in}}(D_{\text{out}})$ rejects the oracle $\pi_{\text{out}}|_{I_{\text{out}}} \circ \pi_R$ (or, actually, any proof oracle augmenting the input oracle $\pi_{\text{out}}|_{I_{\text{out}}}$) with probability greater than ε_{in}. Therefore, $V_{\text{comp}}(x)$ rejects oracle π with probability at least $\varepsilon_{\text{out}} \cdot \varepsilon_{\text{in}}$.

The additional items follow by similar arguments. If V_{out} is a PCPP verifier, then the input is of the form (x,y), where y is given via oracle access. In this case, throughout the proof above we should replace references to the oracle π_{out} with the oracle $y \circ \pi_{\text{out}}$, and for soundness we should consider the case that y is δ_{out}-far from $L(x)$. If V_{in} has robust-soundness, then at the end of the soundness analysis, we note that not only is $\pi_{\text{out}}|_{I_{\text{out}}} \circ \pi_R$ rejected with probability greater than ε_{in} but rather it is ρ_{in}-far from being accepted by V_{in} (and hence also by V_{comp}). $\square$

3. OVERVIEW OF OUR MAIN CONSTRUCT

Throughout this section, n denotes the length of the explicit input given to the PCPP verifier, which in case of CIRCUIT VALUE is defined as the size of the circuit (given as explicit input). As stated in the introduction, our main results rely on the following highly efficient robust PCP of proximity.

THEOREM 3.1 (MAIN CONSTRUCT). There exists a universal constant c such for all $n, m \in \mathbb{Z}^+, \delta, \gamma > 0$ satisfying $n^{1/m} \geq m^{cm}/(\gamma\delta)^3$ and $\delta \leq \gamma/c$, CIRCUIT VALUE has a robust PCP of proximity with the following parameters

- randomness $\left(1 - \frac{1}{m}\right)\log n + O(m \log m) + O(\log\log n) + O(\log(1/\delta))$,
- decision complexity $q = n^{1/m} \cdot \text{poly}(\log n, 1/\delta)$, which also upper-bounds the query complexity.[6]
- perfect completeness, and
- for proximity parameter δ, the verifier has robust-soundness error γ with robustness parameter $(1-\gamma)\delta$.

We comment that the condition $\delta < \gamma/c$ merely means that we present robust PCPs of proximity only for the more difficult cases (when δ is small).

Proof Overview: Theorem 3.1 is proved by modifying a construction that establishes Theorem 1.1. We follow [20] and modify their construction. (An alternative approach would be to start from [24], but that construction does not seem amenable to achieving robust soundness.) The construction of [20] may be abstracted as follows: To verify the satisfiability of a circuit of size n, a verifier expects oracles $\mathrm{F}_i : F^m \to F$, $i \in \{1\ldots, t = \text{poly}\log n\}$, where F is a field and m is a parameter such that $F^m \approx m^m \cdot n$. The verifier then needs to test that (1) each of the F_i's is close to a m-variate polynomial of low degree and (2) the polynomials satisfy some consistency properties which verify that F_i is locally consistent with F_{i-1}.[7] (These consistency checks include tests which depend on the input circuit and verify that F_i's actually encode a satisfying assignment to the circuit.)

We work within this framework — namely our verifier will also try to access oracles for F_i's and test low-degreeness and consistency. Our key modification to this construction is a randomness-reduction in the low-degree test obtained by using the small collection of (small-biased) lines of [8], while using only the "canonical" representations of these lines (and avoiding any complication that was introduced towards "proof composition"). In particular, unlike in [20, 18, 8], we cannot afford to pack the polynomials $\mathrm{F}_1, \ldots, \mathrm{F}_t$ into a single polynomial (by using an auxiliary variable that blows-up the proof length by a factor of the size of the field in use). Instead, we just maintain all these t polynomials separately and test them separately to obtain Theorem 1.1. (In the traditional framework of parallelized PCPs, this would give an unaffordable increase in the number of (non-Boolean) queries. However, we will later ameliorate this loss by a "bundling technique" that will yield robust-soundness.)

The resulting PCP is converted into a PCP of proximity by comparing the input-oracle (i.e. supposed satisfying assignment to the circuit) to the proof-oracle (which is supposed to include an encoding of the said assignment). That is, we read a random location of the input and the corresponding location of the proof oracle, and test for equality. Actually, these locations of the proof-oracle must be accessed via a self-correction mechanism (rather than merely probing at the desired points of comparison), since they constitute only a small part of the proof oracle (and thus corruptions there may not be detected). (This technique was already suggested in [3].)

The most complex and subtle part of the proof of Theorem 3.1 is establishing the robust-soundness property. We sketch how we do this below, first dealing with the low-degree test and the consistency tests separately, and then showing how to reconcile the two "different" fixes.

Low-degree tests of $\mathrm{F}_1, \ldots, \mathrm{F}_t$**:** Selecting a random line $\ell : F \to F^m$ (from the aforementioned sample space), we can check that (for each i) the restriction of F_i to the line ℓ (i.e., the function $f_i(j) \triangleq \mathrm{F}_i(\ell(j))$) is a low-degree (univariate) polynomial. Each of these tests is *individually robust*; that is, if F_i is far from being a low-degree polynomial then with high probability the restriction of F_i to a random line ℓ (in the sample space) is far from being a low-degree polynomial. The problem is that the conjunction of the t tests is not sufficiently robust; that is, if one of the F_i's is δ-far from being a low-degree polynomial then it is only guaranteed that the sequence of t restrictions (i.e., the sequence of the f_i's) is (δ/t)-far from being a sequence of t low-degree (univariate) polynomials. Thus robustness decreases by a factor of t, which we cannot afford for nonconstant t.h

Our solution is to observe that we can "bundle" the t functions together into a function $\overline{\mathrm{F}} : F^m \to F^t$ such that if one of the F_i's is far from being a low-degree polynomial then the restriction of $\overline{F}$ to a random line will be far from being a bundling of t low-degree univariate polynomials. Specifically, for every $x \in F^m$, define $\overline{\mathrm{F}}(x) \triangleq (\mathrm{F}_1(x), ..., \mathrm{F}_t(x))$. To test that $\overline{\mathrm{F}}$ is a bundling of low-degree polynomials, select a random line ℓ (as above), and check that $\overline{f}_\ell(j) = \overline{\mathrm{F}}(\ell(j))$ is a bundling of low-degree univariate polynomials. Thus, we establish *robustness at the bundle level*; that is, if one of the F_i's is far from being low degree then, with high probability, one must modify $\overline{f}_\ell$ on a constant fraction of values in order to make the test accept. The point is that this robustness refers to Hamming distance over the alphabet F^t, rather than alphabet F as before. We can afford this increase in alphabet size, as we later encode the values of $\overline{\mathrm{F}}$ using an error-correcting code in order to derive *robustness at the bit level.*

We wish to highlight a key point that makes the above approach work: when we look at the values of $\overline{\mathrm{F}}$ restricted to a random line, we get the values of the individual F_i's

[6]In fact, we will upper-bound the query complexity by $q = n^{1/m} \cdot \text{poly}(\log n, 1/\delta)$ and show that the verifier's decision can be implemented by a circuit of size $\tilde{O}(q)$, which can also be bounded by $n^{1/m} \cdot \text{poly}(\log n, 1/\delta)$ with a slightly larger unspecified polynomial.

[7]Strictly speaking, the consistency checks are a little more complicated, with the functions really being indexed by two subscripts and consistency tests being between $\mathrm{F}_{i,j}$ and $\mathrm{F}_{i,j-1}$, as well as between $\mathrm{F}_{i,0}$ and $\mathrm{F}_{i+1,0}$. However, these differences don't alter our task significantly — we ignore them in this section to simplify our notation.

restricted to some random line, which is exactly what a low-degree test of each F_i needs. This fact is not very surprising, given that we are subjecting all F_i's to the same test. *But what happens when we need to make two different types of tests?* This question is not academic and does come up in the consistency tests.

Consistency tests: To bundle the t consistency tests between F_i and F_{i+1} we need to look into the structure of these tests. We note that for every i, a random test essentially refers to the values of F_i and F_{i+1} on (random) i-th axis-parallel lines. That is, for every i, and a random $x' = (x_1, ..., x_{i-1}) \in F^{i-1}$ and $x'' = (x_{i+1}, ..., x_m) \in F^{m-i}$, we need to check some relation between $\mathrm{F}_i(x', \cdot, x'')$ and $\mathrm{F}_{i+1}(x', \cdot, x'')$.[8] Clearly, querying $\overline{\mathrm{F}}$ as above on the ith axis-parallel line, we can obtain the relevant values from $\overline{\mathrm{F}}(x', \cdot, x'')$, but this works only for one specific value of i, and other values of i will require us to make other queries. The end result would be that we'll gain nothing from the bundling (i.e., from $\overline{\mathrm{F}}$) over using the individual F_i's, which yields a factor of t loss in the robustness.[9] Fortunately, a different bundling works in this case.

Consider $\overline{\mathrm{F}}'$ such that $\overline{\mathrm{F}}'(x) \triangleq (\mathrm{F}_1(x), GF(2)(S(x)), ..., \mathrm{F}_t(S^{t-1}(x)))$, for every $x \in F^m$, where S denotes a (right) cyclic-shift (i.e., $S(x_1, ..., x_m) = (x_m, x_1 \ldots \ldots, x_{m-1})$ and $S^i(x_1, ..., x_m) = (x_{m-(i-1)}, \ldots, x_m, x_1, x_2, \ldots \ldots x_{m-i})$). Now, if we ask for the value of $\overline{\mathrm{F}}'$ on the first and last axis-parallel lines (i.e., on $(\cdot, x_2, ..., x_m)$ and $(x_2, ..., x_m, \cdot) = S^{-1}(\cdot, x_2, ..., x_m)$), then we get all we need for all the m tests. Specifically, for every i, the i-th component in the bundled function $\overline{\mathrm{F}}'(\cdot, x_2, ..., x_m)$ is $\mathrm{F}_i(S^{i-1}(\cdot, x_2, ..., x_m)) = \mathrm{F}_i(x_{m-i+2}, ..., x_m, \cdot, x_2, ..., x_{m-i+1})$, whereas the $(i+1)$-st component in $\overline{\mathrm{F}}'(S^{-1}(\cdot, x_2, ..., x_m))$ is $\mathrm{F}_{i+1}(S^i(S^{-1}(\cdot, x_2 \ldots, \ldots, x_m))) = \mathrm{F}_{i+1}(x_{m-i+2}, ..., x_m, \cdot, x_2, ..., x_{m-i+1})$. Thus, we need only to query two bundles (rather than t), and robustness only drops by a constant factor.

Reconciling the two bundlings: But what happens with the low-degree tests that we need to do (which were "served" nicely by the original bundling $\overline{\mathrm{F}}$)? Note that we cannot use both $\overline{\mathrm{F}}$ and $\overline{\mathrm{F}}'$, because this will requires testing consistency between them, which will introduce new problems as well as a cost in randomness that we cannot afford. Fortunately, the new bundling (i.e., $\overline{\mathrm{F}}'$), designed to serve the axis-parallel line comparisons, can also serve the low-degree tests. Indeed, the various F_i's will not be inspected on the same lines, but this does not matter, because the property of being a low-degree polynomial is preserved when "shifted" (under S).

Tightening the gap between robustness and proximity: The above description suffices for deriving a weaker version of Theorem 3.1 in which the robustness is only (say) $\delta/3$ rather than $(1-\gamma)\delta$ for a parameter γ that may be set as low as $1/\mathrm{poly}(\log n)$. Such a weaker result yields weaker versions of Theorems 1.2 and 1.3 in which the query complexity is exponentially larger (e.g., for proof-length $\exp(o(\log\log n)^2) \cdot n$, we would have obtained query complexity $\exp(o(\log\log n)) = \log^{o(1)} n$ rather than $o(\log\log n)$). To obtain the stronger bound on the robustness parameter, we take a closer look at the conjunction of the standard PCP test and the proximity test. The PCP test can be shown to have constant robustness $c > 0$, whereas the proximity test can be shown to have robustness $\delta' \triangleq (1-\gamma))\delta$. When combining the two tests, we obtain robustness equal to $\min(\alpha c, (1-\alpha)\delta')$, where α is the relative length of queries used in the PCP test (as a fraction of the total number of queries). A natural choice, which yields the weaker result, is to weight the queries (or replicate the smaller part) so that $\alpha = 1/2$. (This yields robustness of approximately $\min(c, \delta')/2$.) In order to obtain the stronger bound, we assign weights such that $\alpha = \gamma$, and obtain robustness $\min(\gamma c, (1-\gamma)\delta') > \min(\Omega(\gamma), (1-2\gamma)\delta)$, which simplifies to $(1-2\gamma)\delta$ for $\delta < \gamma/O(1)$. (The above description avoids the fact that the PCP test has constant soundness error, but the soundness error can be decreased to γ by using sequential repetitions while paying a minor cost in randomness and *while approximately preserving the robustness*. We comment that the proximity test, as is, has soundness error γ.)

The above completes the overview of the proof of Theorem 3.1 (and the actual proof is provided in our techical report [6]). Composing the above theorem with itself $O(t)$ times yields the following theorem:

THEOREM 3.2. *For every parameter $n, t \in \mathbb{N}$ such that $2 \le t \le \frac{2\log\log n}{\log\log\log n}$ there exists a PCP of proximity for* CIRCUIT VALUE *with randomness complexity* $\log_2 n + O(t + (\log n)^{\frac{1}{t}}) \log\log n + O((\log n)^{\frac{2}{t}}) + t^2 \cdot \mathrm{poly}\log\log\log n$, *query complexity* $O(1)$, *perfect completeness, and soundness error* $1-\Omega(1/t)$ *with respect to proximity parameter* $\Omega(1/t)$. *Alternatively, we can have query complexity* $O(t)$ *and soundness error* $1/2$ *maintaining all other parameters the same.*

Setting $t(n) = 2/\varepsilon$ yields Theorem 1.3 whereas $t(n) = 2\log\log n/\log\log\log n$ yields Theorem 1.2. For obtaining Theorem 3.2, it was essential to have the tight bound on the robustness parameter (as a function of the proximity parameter) in Theorem 3.1. The reason is that when we compose two robust PCPs of proximity the proximity parameter of the second must be upper-bounded by the robustness parameter of the first. Thus, when we compose many robust PCPs of proximity, the robustness parameter deteriorates exponentially in the number of composed systems where the base of the exponent is determined by the tightness of the robustness. Using $\gamma = 1 - (1/t)$, we obtain query complexity that is linear in t (rather than exponential in it).

Acknowledgments

We are grateful to Avi Wigderson for collaborating with us at early stages of this research and to Irit Dinur for inspiring discussions at late stages of this research.

4. REFERENCES

[1] ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. Proof verification and the hardness of approximation problems. *Journal of the ACM 45*, 3 (May 1998), 501–555. (Preliminary Version in *33rd FOCS*, 1992).

[8] Again, this is an oversimplification, but suffices to convey the main idea of our solution.

[9] It turns out that for constant m (e.g., $m = 2$) this does not pose a problem. However, a constant m would suffice only for proving a slightly weaker version of Theorem 1.2 (where $o(\log\log n)$ is replaced by $\log\log n$). but not for proving Theorem 1.3, which requires setting $m = \log^{\varepsilon} n$, for constant $\varepsilon > 0$.

[2] ARORA, S., AND SAFRA, S. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM 45*, 1 (Jan. 1998), 70–122. (Preliminary Version in *33rd FOCS*, 1992).

[3] BABAI, L., FORTNOW, L., LEVIN, L. A., AND SZEGEDY, M. Checking computations in polylogarithmic time. In *Proc.* 23*rd ACM Symp. on Theory of Computing* (New Orleans, Louisiana, 6–8 May 1991), pp. 21–31.

[4] BELLARE, M., GOLDREICH, O., AND SUDAN, M. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal of Computing 27*, 3 (June 1998), 804–915. (Preliminary Version in *36th FOCS*, 1995).

[5] BELLARE, M., GOLDWASSER, S., LUND, C., AND RUSSELL, A. Efficient probabilistically checkable proofs and applications to approximation. In *Proc.* 25*th ACM Symp. on Theory of Computing* (San Diego, California, 16–18 May 1993), pp. 294–304.

[6] BEN-SASSON, E., GOLDREICH, O., HARSHA, P., SUDAN, M., AND VADHAN, S. Robust PCPs of proximity, Shorter PCPs and Applications to Coding. Technical Report (to be posted in ECCC).

[7] BEN-SASSON, E., HARSHA, P., AND RASKHODNIKOVA, S. Some 3CNF properties are hard to test. In *Proc.* 35*th ACM Symp. on Theory of Computing* (San Diego, California, 9–11 June 2003), pp. 345–354.

[8] BEN-SASSON, E., SUDAN, M., VADHAN, S., AND WIGDERSON, A. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proc.* 35*th ACM Symp. on Theory of Computing* (San Diego, California, 9–11 June 2003), pp. 612–621.

[9] BOGDANOV, A., OBATA, K., AND TREVISAN, L. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proc.* 43*rd IEEE Symp. on Foundations of Comp. Science* (Vancouver, Canada, 16–19 Nov. 2002), pp. 93–102.

[10] BOGDANOV, A., AND TREVISAN, L. Lower bounds for testing bipartiteness in dense graphs. Tech. Rep. TR02-064, Electronic Colloquium on Computational Complexity, 2002.

[11] DINUR, I., AND REINGOLD, O. PCP testers: Towards a more combinatorial proof of PCP theorem. (In Preparation), 2003.

[12] ERGÜN, F., KUMAR, R., AND RUBINFELD, R. Fast approximate PCPs. In *Proc.* 31*st ACM Symp. on Theory of Computing* (Atlanta, Georgia, 1–4 May 1999), pp. 41–50.

[13] FEIGE, U., GOLDWASSER, S., LOVÁSZ, L., SAFRA, S., AND SZEGEDY, M. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM 43*, 2 (Mar. 1996), 268–292. (Preliminary version in *32nd FOCS*, 1991).

[14] FORTNOW, L., ROMPEL, J., AND SIPSER, M. On the power of multi-prover interactive protocols. *Theoretical Computer Science 134*, 2 (Nov. 1994), 545–557. (Preliminary Version in *3rd IEEE Symp. on Structural Complexity*, 1988).

[15] GOLDREICH, O., GOLDWASSER, S., AND RON, D. Property testing and its connection to learning and approximation. *Journal of the ACM 45*, 4 (July 1998), 653–750. (Preliminary Version in *37th FOCS*, 1996).

[16] GOLDREICH, O., AND RON, D. Property testing in bounded degree graphs. *Algorithmica 32*, 2 (Jan. 2002), 302–343. (Preliminary Version in *29th STOC*, 1997).

[17] GOLDREICH, O., AND SAFRA, S. A combinatorial consistency lemma with application to proving the PCP theorem. *SIAM Journal of Computing 29*, 4 (2000), 1132–1154. (Preliminary Version in *RANDOM*, 1997).

[18] GOLDREICH, O., AND SUDAN, M. Locally testable codes and PCPs of almost linear length. In *Proc.* 43*rd IEEE Symp. on Foundations of Comp. Science* (Vancouver, Canada, 16–19 Nov. 2002), pp. 13–22. (See ECC Report TR02-050, 2002).

[19] GURUSWAMI, V., LEWIN, D., SUDAN, M., AND TREVISAN, L. A tight characterization of NP with 3-query PCPs. In *Proc.* 39*th IEEE Symp. on Foundations of Comp. Science* (Palo Alto, California, 8–11 Nov. 1998), pp. 18–27.

[20] HARSHA, P., AND SUDAN, M. Small PCPs with low query complexity. *Computational Complexity 9*, 3–4 (Dec. 2000), 157–201. (Preliminary Version in *18th STACS*, 2001).

[21] HÅSTAD, J. Some optimal inapproximability results. *Journal of the ACM 48*, 4 (July 2001), 798–859. (Preliminary Version in *29th STOC*, 1997).

[22] KATZ, J., AND TREVISAN, L. On the efficiency of local decoding procedures for error-correcting codes. In *Proc.* 32*nd ACM Symp. on Theory of Computing* (Portland, Oregon, 21–23 May 2000), pp. 80–86.

[23] LAPIDOT, D., AND SHAMIR, A. Fully parallelized multi prover protocols for NEXP-time (extended abstract). In *Proc.* 32*nd IEEE Symp. on Foundations of Comp. Science* (San Juan, Puerto Rico, 1–4 Oct. 1991), pp. 13–18.

[24] POLISHCHUK, A., AND SPIELMAN, D. A. Nearly-linear size holographic proofs. In *Proc.* 26*th ACM Symp. on Theory of Computing* (Montréal, Québec, Canada, 23–25 May 1994), pp. 194–203.

[25] RAZ, R. A parallel repetition theorem. *SIAM Journal of Computing 27*, 3 (June 1998), 763–803. (Preliminary Version in *27th STOC*, 1995).

[26] RUBINFELD, R., AND SUDAN, M. Robust characterizations of polynomials with applications to program testing. *SIAM Journal of Computing 25*, 2 (Apr. 1996), 252–271. (Preliminary Version in *23rd STOC*, 1991 and *3rd SODA*, 1992).

[27] SAMORODNITSKY, A., AND TREVISAN, L. A PCP characterization of NP with optimal amortized query complexity. In *Proc.* 32*nd ACM Symp. on Theory of Computing* (Portland, Oregon, 21–23 May 2000), pp. 191–199.