Electronic Colloquium on Computational Complexity, Report No. 42 (2009)

Composition of low-error 2-query PCPs using decodable PCPs

Irit Dinur[*] Prahladh Harsha[†]

April 27, 2009

Abstract

The main result of this paper is a simple, yet generic, composition theorem for low error two-query probabilistically checkable proofs (PCPs). Prior to this work, composition of PCPs was well-understood only in the constant error regime. Existing composition methods in the low error regime were non-modular (i.e., very much tailored to the specific PCPs that were being composed), resulting in complicated constructions of PCPs. Furthermore, until recently, composition in the low error regime suffered from incurring an extra 'consistency' query, resulting in PCPs that are not 'two-query' and hence, much less useful for hardness-of-approximation reductions.

In a recent breakthrough, Moshkovitz and Raz [In *Proc. 49th IEEE Symp. on Foundations of Comp. Science (FOCS)*, 2008] constructed almost linear-sized low-error 2-query PCPs for every language in NP. Indeed, the main technical component of their construction is a novel composition of certain specific PCPs. We give an alternate, modular and, considerably simpler proof of their result by repeatedly applying the new composition theorem to known PCP components.

To facilitate the new modular composition, we introduce a new variant of PCP, which we call a *decodable PCP (dPCP)*. A dPCP is an *encoding* of an NP witness that is both locally checkable and locally decodable. The dPCP verifier in addition to verifying the validity of the given proof like a standard PCP verifier, also locally decodes the original NP witness. Our composition is generic in the sense that it works regardless of the way the component PCPs are constructed.

[*] Weizmann Institute, Israel

[†] Technion, Israel

ISSN 1433-8092

1 Introduction

Probabilistically checkable proofs (PCPs) provide a proof format that enables verification with only a constant number of queries into the proof. This is formally captured by the (by now standard) notion of a probabilistic verifier.

Definition 1.1 (PCP Verifier)**.** *A PCP verifier V for a language L is a polynomial time probabilistic algorithm that behaves as follows: On input x, and oracle access to (proof) string π (over an alphabet Σ), the verifier reads the input x, tosses some random coins r, and based on x and r computes a window $I = (i_1, \ldots, i_q)$ of indices to read from π, and a predicate $f : \Sigma^q \to \{0,1\}$. The verifier then accepts iff $f(\pi_I) = 1$.*

- *The verifier is* complete *if for every $x \in L$ there is a proof π accepted with probability 1. I.e., $\exists \pi,\ \Pr_{I,f}[f(\pi_I) = 1] = 1$.*
- *The verifier is* sound *with* soundness error *$\delta < 1$ if for any $x \notin L$, every proof π is accepted with probability at most δ. I.e., $\forall \pi,\ \Pr_{I,f}[f(\pi_I) = 1] \le \delta$.*

The celebrated PCP Theorem [AS98, ALM$^+$98] states that every language in NP has a verifier that is complete and sound with a constant $\delta < 1$ soundness error while using only a logarithmic number of random coins, and reading only $q = O(1)$ proof symbols. Naturally, (and motivated by the fruitful connection to inapproximability due to [FGL$^+$96]), much attention has been given to obtaining PCPs with "good" parameters, such as $q = 2$, smallest possible soundness error δ, and smallest possible alphabet size $|\Sigma|$. These are the parameters of focus in this paper.

How does one construct PCPs with such remarkable proof checking properties? In general, it is easier to construct such PCPs if we relax the alphabet size $|\Sigma|$ to be large (typically super-constant, but sub-linear). This issue is similar to a well-known issue that arises in coding theory; wherein it is relatively easy to construct codes with good error-correcting properties over a large, super constant sized, alphabet (e.g., Reed-Solomon codes). Codes over a constant-sized alphabet (e.g., GF(2)) are then obtained from these codes by (repeatedly) applying the "code-concatenation" technique of Forney [For66]. The equivalent notion in the context of PCP constructions is the paradigm of "proof composition", introduced by Arora and Safra [AS98]. Informally speaking, proof composition is a recursive procedure applied to PCP constructions to reduce the alphabet size. Proof composition is applied (possibly several times over) to PCPs over the large alphabet to obtain PCPs over a small (even binary) alphabet.

Proof composition is an essential ingredient of all known constructions of PCPs. Composition of PCPs with high soundness error (greater than 1/2) is by now well understood due to the works of Szegedy [Sze99], Ben-Sasson et al. [BGH$^+$06] and Dinur and Reingold [DR06]. These works introduce the notion of *PCPs of proximity* [BGH$^+$06] (called *assignment testers* in [DR06]) which allow for extremely modular composition, in the high soundness error regime which in turn led to alternate proofs of the PCP Theorem and constructions of shorter PCPs [BGH$^+$06, Din08, BS08]. However, these composition theorems are inapplicable when constructing PCPs with low-soundness error (arbitrarily small soundness error or even any constant less than 1/2). (See survey on constructing low error PCPs by Dinur [Din08] for a detailed explanation of this limitation).

Our first contribution is a definition of an object which we call a *decodable PCP*, which allows for clean and modular composition in the low error regime.

1.1 Decodable PCPs (dPCPs)

Consider a probabilistically checkable proof for the language CircuitSat (the language of all satisfiable circuits). The natural NP proof for CircuitSat is simply a satisfying assignment. An intuitive way to construct a PCP for CircuitSat is to *encode* the assignment in a way that enables probabilistic checking. This intuition guides all known constructions, although it is not stipulated in the definition.

In this work, we make the intuitive notion of proof encoding explicit by introducing the notion of a *decodable PCP (dPCP)*. A dPCP for CircuitSat is an encoding of the satisfying assignment that can be both verified and decoded locally in a probabilistic manner. In this setting, the verifier is supposed to both verify that the dPCP is encoding an *satisfying* assignment, as well as to decode a symbol in that assignment. More precisely, we define a *PCP decoder* for CircuitSat to be (along the lines of Definition 1.1) a probabilistic algorithm that is given an input circuit C, oracle access to a dPCP π, and, in addition, an index i. Based on C, i and the randomness r it computes a window I and a *function* f (rather than a predicate). This function is supposed to evaluate to the i-th symbol of a satisfying assignment for C; or to reject.

- The PCP decoder is *complete* if for every y such that $C(y) = 1$ there is a dPCP π such that $\Pr_{i,I,f}[f(\pi_I) = y_i] = 1$.
- The PCP decoder has *soundness error* δ and list size L if for any (purported) dPCP π there is a list of $\leq \mathsf{L}$ valid proofs such that $\Pr_{i,I,f}[f(\pi_I)$ inconsistent with the list, but not reject$] \leq \delta$.

The list of valid proofs can be viewed as a "list decoding" of the dPCP π. Since we are interested in the low soundness error regime, list-decoding is unavoidable. Of course, we can define dPCPs for any NP language and not just CircuitSat, but we focus on CircuitSat since it suffices for the purpose of composition.

The notion of dPCPs allows for clean and modular composition in the case of low soundness error (described next) in analogy to the way PCPPs and assignment testers [BGH$^+$06, DR06] allow for modular composition in the case of high soundness error. Moreover, using dPCPs we show a two query composition that yields an alternate completely modular proof of the recent breakthrough result of Moshkovitz and Raz [MR08b].

Finally, we note that decodable PCPs are not hard to come by. Decodable PCPs or variants of them are implicit in many PCP constructions [AS03, RS97, DFK$^+$99, BGH$^+$06, DR06, MR07, MR08b] and existing PCP constructions can often be easily adapted to yield decodable PCPs (c.f., Section 6).

1.2 Composition with dPCPs

There is a natural and modular way to compose a PCP verifier[1] V with a PCP decoder $\mathcal{D}$. The composed PCP verifier V' begins by simulating V on a probabilistically checkable proof Π. It determines a set of queries into Π (a local window I), and a local predicate f. Instead of directly querying Π and testing if $f(\Pi_I) = 1$, V' relies on the inner PCP decoder $\mathcal{D}$ to perform this action. For this task, the inner PCP decoder $\mathcal{D}$ is supplied with a dedicated proof that is supposedly an encoding of the relevant local view Π_I. The main issue is consistency: the composed verifier V' must ensure that the dedicated proofs supposedly encoding the various local views are consistent

[1] The verifier needs to be a *robust* PCP as in Definition 2.3, but we gloss over this issue in the introduction.

with the same Π (i.e. they should be encodings of local views coming from a single valid PCP for V). This is achieved easily with PCP decoders: the composed verifier V' asks $\mathcal{D}$ to decode a random value from the encoded local view, and compares it to the appropriate symbol in Π.

The above description of composition already appears[2] to lead to a modular presentation of the composition performed in earlier low-error PCP constructions [AS03, RS97, DFK+99, MR07]. But at the same time, like these compositions, it incurs an additional query per composition, namely the "consistency" query to the outer PCP Π. (The queries made by V' are the queries of $\mathcal{D}$ plus the one additional consistency query to Π).

Nevertheless, inspired by [MR08b] and equipped with a cleaner understanding of composition in the low soundness error case, we are in a better position to remove this extra consistency query.

1.3 Composition with only two queries

Our main contribution is a surprisingly simple composition theorem that does not incur an extra query. The extra query was incurred in the above described composition as the composed verifier needed to verify that all the inner PCP decoders decode to the same symbol. This check was performed by comparing the decoded symbol to the symbol in the outer PCP Π. Instead, we verify consistency by invoking *all* the inner PCP decoders that involve this symbol *in parallel*, and then checking that they all decode to the same symbol. This avoids the necessity to query the outer PCP Π for this symbol and saves us the extra query.

We describe our new composed verifier V' more formally below. As before, let V be a PCP verifier, and $\mathcal{D}$ a PCP decoder.

1. The composed PCP verifier simulates V on a hypothetical PCP Π; it chooses a random index i in Π, and then determines *all* the possible random strings $R_1, \ldots, R_D$ that cause V to query this index.

2. For each random string R_j ($j = 1 \ldots D$), V' needs to check that the corresponding local view of Π would have lead V to accept. This is done by running $\mathcal{D}$, for each $j = 1 \ldots D$, on a dedicated proof $\pi(R_j)$ that is supposedly the encoding of the j-th local view (i.e., the one generated by V on random string R_j) into Π. Furthermore, V' expects $\mathcal{D}$ to decode the symbol Π_i.

3. Finally V' accepts if and only if *all* the D parallel runs of $\mathcal{D}$ accept and output the same symbol.

Observe that the composed verifier V' does not access the PCP for V (i.e., Π) at all, rather only the dedicated proofs for the inner PCP decoders. The outer PCP Π is only "mentally" present in order to compute $R_1, \ldots, R_D$. A few important points are in order.

- **Two Queries and Robust Soundness:** As described, V' makes many queries rather than just two. This is fixed by the following easy transformation: the first query will supposedly be answered by the complete local view V' expects to read, and the second query will consist of one random symbol in the local view of V'. The soundness of the resulting two-query PCP is equal to the *robust soundness* of V': an upper bound on the average agreement between a local view read by V' and an accepting local view. This interesting correspondence between two query PCPs and robust PCPs is true in general and described in full in Section 2.2.

[2] We have not verified the details.

Thus, drawing on the above correspondence, the fact that V' has low robust soundness implies the required two-query composition. Of course, the composition could have been described entirely in the 2-query PCP language.

- **Size of alphabet or window size:** The purpose of composition is to reduce the alphabet size, or, in the language of robust PCPs, to reduce the window size, that is, the number of queries made by V'. Recall that V' runs $\mathcal{D}$ in parallel on all D local views corresponding to $R_1, \ldots, R_D$. Thus, the window size equals the query complexity of $\mathcal{D}$ multiplied by the number D of local views (which we refer to as the *proof degree* of V). Hence composition is meaningful only if the proof degree is small to begin with (otherwise, the local window of V' is not smaller than that of V and we haven't gained anything from composition). In general PCPs, the proof degree is very high. In fact, this has been one of the obstacles to achieving this result prior to [MR08b]. However, a key observation of [MR08b] is that it is easy to reduce the proof degree using standard tools from derandomization (i.e., expander replacement).

  Alternatively, one can handle V of arbitrarily high proof degree by making the following change to V'. Instead of running $\mathcal{D}$ to verify the local tests corresponding to *all* of $R_1, \ldots, R_D$, V' can *pseudo-randomly* choose a small number of these and run $\mathcal{D}$ only on the selected ones.

  The fact that the query complexity is at least D is an inherent bottleneck in our composition method. Combined with the bound of $D \geq 1/\delta$, this poses a limitation of this technique towards achieving exponential dependence of the error probability on alphabet size, a point discussed later in this introduction.

The new composition is generic in the sense that it works regardless of how the original components V and $\mathcal{D}$ are constructed.

1.4 Background and Motivation

Let us step back to give some motivation for obtaining PCPs with small soundness and two queries (for a more comprehensive treatment, see [MR08b]). Two is the absolute minimal number of queries possible for a non-trivial PCP. Thus, it is interesting to find what are the strongest 2-query PCPs that still capture NP. However, the main motivation for two query PCPs is for proving hardness of approximation results.

Two query PCPs with soundness error δ are (more or less) equivalent to $\textsc{Label-Cover}_\delta$, which is a promise problem defined as follows[3]: The input is a bipartite graph and an alphabet Σ, and for each edge e there is a function $f_e : \Sigma \to \Sigma$, which we think of as a *constraint* on the labels of the vertices. The constraint is satisfied by values a and b iff $f_e(a) = b$. The problem is to distinguish between two cases: (1) there exists a labeling of the vertices satisfying all constraints, or (2) every labeling satisfies at most δ fraction of the constraints.

$\textsc{Label-Cover}_\delta$ is probably the most popular starting point for hardness of approximation reductions. In particular, even though there are 3-query PCPs with much smaller soundness error, they currently have far fewer applications to inapproximability.

The fact that $\textsc{Label-Cover}_\alpha$ is NP-hard for some constant $\alpha < 1$ (and constant alphabet size) is nothing but a reformulation of the PCP Theorem [AS98, ALM$^+$98]. Strong inapproximability

[3] We focus on the important special case of projection constraints. For a more accurate definition, see Definition 2.2.

results, however, require[4] NP-hardness of Label-Cover$_\delta$ for arbitrarily small, sometimes even sub-constant soundness error δ. There are two known routes to obtaining hardness results for Label-Cover$_\delta$ with small soundness δ. The first, is via an application of the parallel repetition theorem of Raz [Raz98] to the Label-Cover$_\alpha$ instance produced by the PCP Theorem. However, this application of the repetition theorem blows up the size of the problem instance from n to $n^{O(\log(1/\delta))}$ and thus remains polynomial only for constant, though arbitrarily small, δ. One might try to get a polynomial sized construction by carefully choosing a subset of the entire parallel repetition construction. This is known as the problem of "derandomizing the parallel repetition theorem". Feige and Kilian [FK95] showed that such derandomization is impossible under certain (rather general) conditions. Nevertheless, in a recent paper, Impagliazzo et. al. [IKW09] obtained a related derandomization. While their derandomization result applies only to direct products and not to the construction of PCPs, this direction seems promising. Another potential direction is to use the gap-amplification technique of Dinur [Din07], however as shown by Bogdanov [Bog05] gap-amplification fails below a soundness error of 1/2.

The second route to sub-constant δ goes through the classical (algebraic) construction of PCPs. Indeed, hardness for label cover with sub-constant error can be obtained from the low soundness error PCPs of [RS97, AS03, MR08a], more or less by omitting the composition steps, and carefully combining queries. The following "manifold vs. point" PCP construction has been folklore since [RS97, AS03], and formally described in [MR08b].

Theorem 1.2 (Manifold vs. Point PCP). *There exists a constant $c > 1$ such that the following holds: For every $\frac{1}{n} \leq \delta \leq \frac{1}{(\log n)^c}$, there exists an alphabet Σ of size at most $\exp(\text{poly}(1/\delta))$ such that* Label-Cover$_\delta$ *over Σ is NP-hard.*

The above result is unsatisfactory as the size of the alphabet $|\Sigma|$ is super-polynomial. Combined with the fact that hardness-of-approximation reductions are usually exponential in $|\Sigma|$ (and always at least polynomial in $|\Sigma|$) the super polynomial size of Σ renders the above theorem useless. The situation can be redeemed if the theorem could be extended to the entire range of smaller $|\Sigma|$ (with a corresponding increase in δ).

A natural way to perform this extension would be to apply the composition paradigm to the PCPs constructed in Theorem 1.2 and reduce the alphabet size. Indeed, this is how one constructs PCPs with sub-constant error and *a constant* number of queries for the entire range of $\Omega(1) \leq |\Sigma| \leq \exp((\log n)^{1-\varepsilon})$ [RS97, AS03, DFK$^+$99]. However, the composition a la [RS97, AS03, DFK$^+$99] incurs at least one additional query, which means that the final PCP is no longer "two-query", so it does not lead to a hardness result for label cover. Alternatively, the composition technique of [BGH$^+$06, DR06] using PCPs of proximity or assignment testers is inapplicable in this context as it fails to work for soundness error less than 1/2. Thus, all earlier composition techniques are either inapplicable in the low error regime or if applicable, incur an extra query and thus, are no longer in the framework of the Label-Cover problem.

1.5 The Two-Query PCP of Moshkovitz and Raz [MR08b]

In a recent breakthrough, [MR08b] show that the above theorem can in fact, be extended to the entire range of δ and $|\Sigma|$ (and maintaining $|\Sigma| \approx \exp(\text{poly}(1/\delta))$). This is done by composing

[4] In some cases the hardness gap is inversely proportional to δ, and in others, it is the sum of two terms: a problem-dependent term (e.g. 7/8 in Håstad's hardness result [Hås01] for 3-SAT), and a "low order" term that is polynomial in δ.

certain specific 2-query PCPs with low soundness error without incurring an additional query per composition.

Theorem 1.3 ([MR08b]). *For every $\delta \in (1/\text{polylog}n, 1)$, there exists an alphabet Σ of size at most* $\exp(\text{poly}(1/\delta))$ *such that* $\textsc{Label-Cover}_\delta$ *over Σ is NP-hard (in fact, even under nearly length preserving reductions).*

The main technical component of their construction is a novel composition of certain specific PCPs. However, the construction is so organically tied to the specific algebraic components that are being composed, as to make it extremely difficult to differentiate between the details of the PCP, and what it is that makes the composition go through.

We give an alternate, modular and, in our view, a considerably simpler proof of this theorem using our composition theorem in Section 6. Our proof relies on a PCP system based on the manifold vs. point construction (as in Theorem 1.2). The parameters we need are rather weak: it is enough that on input size n the PCP decoder / verifier makes n^α queries and has soundness error $\delta = 1/n^\beta$, for small constants α, β. After one composition step the number of queries goes (roughly) from n^α to n^{α^2}, and so on. After each composition step we add a combinatorial step, consisting of degree and alphabet reduction, that prepares the verifier for the next round of composition. After i rounds the number of queries is about n^{α^i}, and the soundness error is about $\delta = 1/n^{O(\alpha^i)}$. Choosing $1 \leq i \leq \log\log n$ appropriately gives us the result.

The modular composition theorem allows us to easily keep track of a super-constant number of steps, thus avoiding the need for another tailor-made Hadamard-based PCP which was required in the proof of [MR08b]. (The later approach could also be implemented in our setting).

Generic transformations on $\textsc{Label-Cover}$: We also give generic transformations on $\textsc{Label-Cover}$, such as alphabet reduction, degree reduction, and regularization, which are needed before applying composition. These transformations incur only a moderate cost to the other parameters. To the best of our knowledge, the alphabet reduction and the regularizing transformations are new, and may be of independent interest.

Randomness and the length of the PCP: The above discussion completely ignores the randomness complexity of the underlying PCPs. However, it is easy to verify that the composition described above is, in fact, randomness efficient; this is because the same inner randomness can be used for all the D parallel runs of the inner PCP decoder. Thus, if we start from a version of the Theorem 1.2 (the manifold vs. point PCP) based on an almost linear-size low-degree test (c.f., [MR08a]), we obtain a nearly length preserving version of Theorem 1.3 (i.e., a reduction taking instances of size n to instances of size almost linear in n). Furthermore, the fact that we account for the input index i separately from the inner randomness r of the PCP decoder leads to an even more randomness-efficient composition, however, we do not exploit this fact in the proof of Theorem 1.3.

Polynomial dependence of soundness error on alphabet size: Theorem 1.3 suffers from the following bottleneck: the error probability δ is inverse logarithmic (and not inverse-polynomial) with respect to the size of the alphabet Σ. This limitation is inherent in our composition method as discussed above. Thus, the "sliding-scale conjecture" of Bellare et al. [BGLR93] that for every $|\Sigma| \in (1, n)$, $\textsc{Label-Cover}_\delta$ over Σ is NP-hard for $\delta = \text{poly}(1/|\Sigma|)$ remains open.

Organization

The rest of the paper is organized as follows. In Section 2 we define the known notions of robust PCPs and label cover, and describe the syntactic equivalence between them. We introduce decodable PCPs in Section 3. The main result of the paper, two-query composition theorem, is then presented in Section 4. This is then followed by Section 5 which contains various basic transformations of label cover such as degree reduction, alphabet reduction, etc. In Section 6, we construct the building blocks for composition and then repeatedly compose them to obtain Theorem 1.3. Various extensions of decodable PCPs are discussed in Appendix A.

2 Preliminaries

2.1 Notation

We begin by formalizing our notation while dealing with strings over some alphabet Σ. For any string $\pi \in \Sigma^n$ and $I \subseteq [n]$, a subset of indices, we refer by π_I, the restriction of π to the indices in I. In other words, if $I = \left\{i_1 < i_2 < \ldots < i_{|I|}\right\}$, then $\pi_I \triangleq \pi_{i_1}\pi_{i_2}\cdots\pi_{i_{|I|}}$. For any subset of indices $I = \left\{i_1 < i_2 < \ldots < i_{|I|}\right\}$ and index $i \in I$ such that $i_k = i$, we refer to k as the index of i within I and denote the same by $\mathrm{index}_{i\in I}$. Observe that this re-indexing satisfies the property that for any string $\pi \in \Sigma^n$, we have $(\pi_I)_{(\mathrm{index}_{i\in I})} = \pi_i$. We will reserve the symbol $\perp$, which will not be a member of any of the alphabets we use, to denote "reject" or "fail".

For any two strings $x, y \in \Sigma^n$, the (relative) agreement between x and y, denoted by $\mathrm{agr}(x,y)$, is defined as the fraction of locations on which x and y agree (i.e., $\mathrm{agr}(x,y) \triangleq \mathrm{Pr}_{i\in[n]}[x_i = y_i]$). The agreement between a string and a set of strings $L \subseteq \Sigma^n$ is defined in the natural manner: $\mathrm{agr}(x,L) = \max_{y\in L}(\mathrm{agr}(x,y))$. For any set of strings $L \subseteq \Sigma^n$ and index $i \in [n]$, we denote by L_i the set of symbols obtained by restricting the strings in L to the i^{th} index, i.e., $L_i = \{w_i \mid\ w \in L\}$. The following fact about agreement of strings will come useful.

Fact 2.1. *Let $L \subseteq \Sigma^n$ and $s \in \Sigma^n$. Then $\mathrm{agr}(s,L) \geq |L|^{-1} \cdot \mathrm{Pr}_i[s_i \in L_i]$.*

Proof. The event $s_i \in L_i$ is the union of the events $\{s_i = w_i\}$ for all $w \in L$, hence

$$|L|^{-1} \cdot \Pr_i[s_i \in L_i] \leq |L|^{-1} \cdot \sum_{w\in L} \Pr_i[s_i = w_i] = \mathop{\mathbb{E}}_{w\in L}[\mathrm{agr}(s,w)] \leq \mathrm{agr}(s,L)$$

□

Now, for some terminology for circuits. Unless otherwise stated, all *circuits* in this paper will have fan-in 2 and fan-out 2 and we allow arbitrary unary and binary Boolean operations as internal gates. The *size* of a circuit is the number of gates. The typical NP-complete language we will refer to is CircuitSat, the set of satisfiable Boolean circuits, defined as follows: $\textsc{CircuitSat} = \{C \mid\ \exists w, C(w) = 1\}$. Note that the instance C is specified as a circuit and not a truth-table in the above definition.

Sometimes, we will refer to circuits computing a function over a non Boolean alphabet Σ and outputting a symbol from a (possibly different) non-Boolean alphabet σ, such as $f : \Sigma^n \to \sigma$. This is merely shorthand for the equivalent function $f' : \{0,1\}^{n\cdot\log|\Sigma|} \to \{0,1\}^{\log|\sigma|}$, where Σ and σ are viewed as bit-strings of length $\log|\Sigma|$ and $\log|\sigma|$ respectively. The circuit complexity of such a function f is defined to be the circuit complexity of f'. When working with the alphabet Σ, we will frequently refer to the corresponding NP-complete language, $\textsc{CircuitSat}_\Sigma$, the set of satisfiable Boolean circuits over the alphabet Σ, defined as follows: $\textsc{CircuitSat}_\Sigma = \{f : \Sigma^n \to \{0,1\} \mid \exists w \in \Sigma^n, f(w) = 1\}$. As in the Boolean setting, the instance $f : \Sigma^n \to \{0,1\}$ is specified as a circuit $C_f : \{0,1\}^{n \cdot \log|\Sigma|} \to \{0,1\}$.

2.2 Label Cover and Robust PCPs

In this section, we point to an interesting correspondence between two known objects, namely, the Label-Cover problem and robust PCPs. We first define these two objects (in Section 2.2.1 and Section 2.2.2), and then (in Section 2.2.3) show the equivalence of the following two statements (a) a language L is reducible to $\textsc{Label-Cover}_\delta$ and (b) L has a robust PCP with soundness error δ. This equivalence is very important in this paper, as we move back and forth between the two views: the composition theorem is more natural to describe in terms of robust PCPs, while the other manipulations (such as degree and alphabet reduction) are easier to describe in terms of Label-Cover. (The application of the final result for inapproximability also requires the Label-Cover formulation).

A weak equivalence of this nature has been implicitly observed (at least in one direction) earlier, but, to the best of our knowledge, this is the first time a formal syntactic equivalence between the two notions has been established.

2.2.1 Label Cover

We begin with the definition of the Label-Cover problem. Formally defined by Arora et al. [ABSS97], but implicit in several earlier hardness reductions, the Label-Cover problem has been the starting point of a long list of hardness reductions.

Definition 2.2 (Label-Cover)**.** *An instance of the* Label-Cover *problem is specified by a quadruple* $(G, \Sigma_1, \Sigma_2, F)$ *where* $G = (U, V, E)$ *is a bipartite graph,* Σ_1 *and* Σ_2 *are two finite sized alphabets and* $F = \{f_e : \Sigma_1 \to \Sigma_2 \mid e \in E\}$*, is a set of functions (also called projections), one for each edge.*

A labeling $L = (\Sigma_1^U, \Sigma_2^V)$*, (i.e., a pair of labelings* $L_1 : U \to \Sigma_1$ *and* $L_2 : V \to \Sigma_2$*) is said to satisfy an edge* (u, v) *iff* $f_{(u,v)}(L_1(u)) = L_2(v)$*. The* value *of an instance is the maximal fraction of edges satisfied by any such labeling.*

For any $\delta \in (0, 1)$*, the gap problem* $\textsc{Label-Cover}_\delta$ *is the promise problem of deciding if a given instance has value* 1 *or at most* δ*.*

We refer to U and V as the "left" and "right" vertices, and to Σ_1 and Σ_2 as the "left" and "right" alphabets. The *left degree* of an instance (resp. the *right degree*) is defined naturally as the maximum degree of a left vertex (resp. of a right vertex). In general, we will assume that all the Label-Cover instances we construct are regular (i.e, the left (right) degree of all left (right) vertices are the same), unless explicitly stated otherwise. In fact, in Section 5 we show how to "regularize" any Label-Cover instance without altering its other parameters very much

The Label-Cover problem is often viewed as a "two-query" PCP. This is because a reduction from L to Label-Cover can be converted into a two-query PCP verifier: the verifier expects a labeling as a proof and checks that a random edge is satisfied by reading its two endpoints.

2.2.2 Robust PCPs

Next, we recall the notion of robust PCPs, which has been very useful in PCP constructions. Formally defined in [BGH$^+$06, DR06], robust PCPs have been implicit in all PCP constructions since the original proof of the PCP Theorem [AS98, ALM$^+$98] (especially in PCP constructions which involve composition). The only difference between robust PCPs and regular PCPs is in the soundness condition: while the standard soundness condition measures how often the PCP verifier accepts a false proof, the robust soundness condition measures the *average distance* between the local view of the verifier and an accepting local view.

Definition 2.3 (robust PCPs)**.** *For functions* $\mathsf{r}, \mathsf{q}, \mathsf{m}, a, s : \mathbb{Z}^+ \to \mathbb{Z}^+$ *and* $\delta : \mathbb{Z}^+ \to [0, 1]$*, a* verifier V *is a* robust probabilistically checkable proof (robust PCP) system *for a language* L *with* randomness complexity r*,* query complexity q*,* proof length m*,* alphabet size a*,* decision complexity s *and* robust soundness error δ *if* V *is a probabilistic polynomial-time algorithm that behaves as follows: On input* x *of length* n *and oracle access to a proof string* $\pi \in \Sigma^{\mathsf{m}(n)}$ *over the (proof) alphabet* Σ *where* $|\Sigma| = a(n)$*,* V *reads the input* x*, tosses at most* $\mathsf{r} = \mathsf{r}(n)$ *random coins, and generates a sequence of locations* $I = (i_1, \ldots, i_{\mathsf{q}}) \in [\mathsf{m}]^{\mathsf{q}(n)}$ *and a predicate* $f : \Sigma^{\mathsf{q}} \to \{0, 1\}$ *of decision complexity* $\mathsf{s}(n)$*, which satisfy the following properties.*

Completeness: *If* $x \in L$ *then there exists* π *such that*

$$\Pr_{(I,f)} [f(\pi_I) = 1] = 1.$$

(Robust) Soundness: *If* $x \notin L$ *then for every* π*,*

$$\mathop{\mathbb{E}}_{(I,f)} \left[\operatorname{agr}\left(\pi_I, f^{-1}(1)\right)\right] \leq \delta. \tag{2.1}$$

where the distribution over (I, f) *is determined by* x *and the random coins of* V*.*

Robust soundness must be contrasted with soundness of standard PCP verifiers in which (2.1) is replaced by

$$\Pr_{I,f}[f(\pi_I) = 1] \leq \delta$$

(in fact, this is the *only* difference between the above definition and the standard definition of a PCP system). The robust soundness states that not only does the local view violate the local predicate f, but in fact has very little agreement with any of the satisfying assignments of f.

Remark 2.4. For readability, our notation does not reflect the fact that I and f depend on both x and the random coins r. When not clear from the context we may write $I(x, r)$ or $I(r)$ to highlight this dependence. Note that as usual, all of the parameters $(\mathsf{r}, \mathsf{q}, \mathsf{m}, |\Sigma|, s, \delta)$ are functions of the input length $|x| = n$, but not of the input itself. We will find it convenient to refer to the sequence of locations $I = I(r)$ as the *local window*, f as the *local predicate* and the proof restricted to the local window, i.e., π_I, as the *local view* of the proof.

2.2.3 Correspondence between Label-Cover and robust PCPs

We now proceed to describe the correspondence between the notions of Label-Cover and robust PCPs.

If a language L has a robust PCP, then here is a reduction from L to Label-Cover: the set of left vertices is the set of random strings of the robust PCP, the set of right vertices is the set of the proof locations. An edge (r, i) exists if the proof location i is probed on random string r. The label to a left vertex r is an accepting local view of the verifier on random string r while a label to the right vertex i is the proof symbol in the corresponding proof location i. An edge (r, i) is consistent if the local view is consistent with the proof symbol.

Conversely, a reduction from L to Label-Cover defines a robust PCP verifier as follows: the verifier expects as proof a labeling of the set of right vertices, the verifier chooses a random left vertex, queries all its neighbors and accepts iff there exists a label to the left vertex that satisfies all the corresponding edges.

This correspondence is summarized more formally in the following lemma statement. Note that this correspondence is akin to the correspondence between bipartite graphs with left degree q and q-uniform hyper-graphs. The proof is straightforward. One direction is proved along the lines of Fortnow, Rompel and Sipser's result [FRS94] that every language in MIP has a 2-prover MIP. (cf., [BGH$^+$06, Proposition 2.14]).

Lemma 2.5 (Robust PCP $\equiv$ Label-Cover)**.** *For every $\delta : \mathbb{Z}^+ \to \mathbb{R}^+$, and $\mathsf{r}, \mathsf{q}, \mathsf{m}, a : \mathbb{Z}^+ \to \mathbb{Z}^+$, the following two statements are equivalent:*

1. Label-Cover$_\delta$ *is NP-hard for instances with the following parameters*
   - *left degree at most* $\mathsf{q}(n)$,
   - *right alphabet* $\Sigma(n)$ *with* $|\Sigma| = a(n)$,
   - *left alphabet* $\Sigma'(n)$,
   - *size of right vertex set at most* $\mathsf{m}(n)$, *and*
   - *size of left vertex set at most* $2^{\mathsf{r}(n)}$.
2. *Every $L \in NP$ has a robust PCP with robust soundness error δ and the following parameters*
   - *query complexity* $\mathsf{q}(n)$,
   - *proof alphabet* $\Sigma(n)$ *with* $|\Sigma| = a(n)$,
   - *maximum number of accepting local views*[5] $|\Sigma'(n)|$,
   - *proof length* $\mathsf{m}(n)$, *and*
   - *randomness complexity at most* $\mathsf{r}(n)$.

Proof Sketch: $(1 \to 2)$: Given a reduction from $L \in NP$ to Label-Cover$_\delta$, we construct a verifier for L as follows. The verifier, on input x, computes (using the reduction) a Label-Cover instance $I = ((U, V, E), \Sigma, \Sigma', F)$. The verifier expects the proof to contain a labeling of V, and uses its random bits to select a random left vertex $u \in U$ and reads the labels of every neighbor of u. It accepts iff there exists a label for u that, together with labels of its neighbors given by the proof,

[5] This is sometimes called the free bit complexity. More precisely, $|\Sigma'(n)| = 2^{\mathsf{fb}}$ where fb is the free bit complexity.

satisfies all the constraints adjacent to u. Given a proof, i.e., a right labeling $L_2 : V \to \Sigma'$ which has robust soundness error δ, then there exists a left labeling $L_1 : U \to \Sigma$ such that the labeling $L = (L_1, L_2)$ satisifies exactly δ fraction of the edge constraints.

$(2 \to 1)$: Given a robust verifier for L we construct a reduction from L to Label-Cover. The reduction maps an input x to an instance $I = ((U, V, E), \Sigma, \Sigma', F)$ where U has a vertex per random string of the verifier, and V has a vertex per proof symbol. A vertex $u \in U$ will be adjacent to all proof symbols that the verifier reads when given the corresponding random string. A label $a \in \Sigma$ will describe an entire accepting view of the verifier, and the constraints will check consistency. Given a labeling $L = (L_1, L_2)$ of the Label-Cover instance that satisfies at least δ fraction of the edges, it is easy to see that the proof given by $L_2 : V \to \Sigma'$ has robust soundness error at least δ. □

It is important to note that this is a *syntactic* correspondence between the notions of Label-Cover and robust PCPs and there is no loss of parameters in going from one framework to another. In particular, going from Label-Cover to a robust PCP and back, one gets back the original Label-Cover instance.

To get comfortable with this correspondence, let us see how composition of two-query PCPs (i.e., verifiers derived from Label-Cover) looks in terms of robust PCPs. In the Label-Cover world the aim of composition is to reduce the alphabet size. (In fact, the main issue is to reduce the *left* alphabet, since reducing the right alphabet is much easier, see Section 5). When translating to a robust PCP, the alphabet size is the free bit complexity. So the aim of composition for robust PCPs would be to reduce the *free bit complexity*. We will actually be more stringent in our demands from composition of robust PCPs and expect composition to reduce the *query complexity* which upper bounds the free-bit complexity.

We end this section with a definition.

Definition 2.6 (Proof degree)**.** *Given a robust PCP system, we will refer to the maximum number of local windows any index in the proof participates in, as the* proof degree*, denoted by $d(n)$. More precisely, for each $i \in [\mathsf{m}(n)]$, if we let*

$$\mathcal{R}_i = \left\{ r \in \{0,1\}^{\mathsf{r}(n)} \,\middle|\, i \in I(r) \right\},$$

then $d(n) = \max_i |\mathcal{R}_i|$. Furthermore, if $|\mathcal{R}_i| = d(n)$ for all i, we will say the PCP system is regular*.*

Observe that the notion of proof degree exactly corresponds to the right degree of the Label-Cover instance according to the equivalence in Lemma 2.5. Furthermore, the PCP system is regular iff the corresponding Label-Cover instance is *right-regular*. In general, all the PCP systems (and hence Label-Cover instances) we will dealing with will be regular, unless explicitly stated otherwise. In fact, in Section 5, we give a reduction that "regularizes" a robust PCP.

3 Decodable PCPs

Consider a PCP for some language in NP. Known PCP constructions have the property that the PCP π is an encoding of the original NP proof. In fact, some constructions have the additional property that every bit of the NP-proof can be *locally decoded* from the PCP π. We make this notion explicit, in the form of *PCP decoders* and *decodable PCPs*. For example, consider the language $\textsc{CircuitSat}_\Sigma$, which consists of circuits $C : \Sigma^k \to \{0,1\}$ that are satisfiable (i.e., there exists a string y that causes C to evaluate to true). The PCP for checking satisfiability of an instance C of $\textsc{CircuitSat}$ is typically a probabilistically checkable encoding of a string y such that $C(y) = 1$. Such a y is called the NP-witness of the fact "$C \in \textsc{CircuitSat}$". A PCP verifier for the language $\textsc{CircuitSat}$ would verify that the input circuit is satisfiable, with the help of a PCP, which is typically (but not-necessarily) an encoding of the NP-witness y. A PCP decoder for $\textsc{CircuitSat}$*expects* the PCP to be an encoding of the NP witness. Like a PCP verifier, the PCP decoder verifies with the help of the PCP that "$C \in \textsc{CircuitSat}$", and *furthermore* decodes the PCP back to the NP witness. Formally, the PCP decoder gets as additional input an index j, and is supposed to either reject or return the j^{th} symbol of the NP witness.

Definition 3.1 (PCP Decoders). *A* PCP decoder *for* $\textsc{CircuitSat}_\Sigma$ *over a proof alphabet* σ *is a probabilistic polynomial-time algorithm* $\mathcal{D}$ *that on input a circuit* $C : \Sigma^k \to \{0,1\}$ *of decision complexity* n *and an index* $j \in [k]$*, tosses* $\mathsf{r} = \mathsf{r}(n)$ *random coins and generates (1) a sequence of* $\mathsf{q} = \mathsf{q}(n)$ *locations* $I = (i_1, \dots, i_\mathsf{q})$ *in a proof of length* $\mathsf{m}(n)$ *and (2) a (local decoding) function* $f : \sigma^\mathsf{q} \to \Sigma \cup \{\perp\}$ *of decision complexity at most* $\mathsf{s}(n)$*.*

For readability, our notation does not reflect the fact that I and f depend on C, r and j. When not clear from the context we may write $I(C, r, j)$ or $I(r, j)$ to highlight this dependence (and similarly for f). Clearly, neither I nor f depend on the proof string π.

We think of the PCP decoder $\mathcal{D}$ as representing a probabilistic oracle machine that based on its input C, the index j and random coins queries its proof oracle $\pi \in \sigma^\mathsf{m}$ for the positions in the local window I, receives the local view π_I consisting of the q symbols $(\pi_{i_1}, \dots, \pi_{i_\mathsf{q}})$ and outputs $f(\pi_I)$.

All of the parameters $|\sigma|, |\Sigma|, \mathsf{m}, k, \mathsf{r}, \mathsf{q}, \mathsf{s}$ are understood to be functions of the input length n, and not of the input itself. We call r the *randomness complexity*, q the *query complexity*, m the *proof length*, $a = |\sigma|$ the *proof alphabet size*, and s the *decoding complexity* of the PCP decoder $\mathcal{D}$. We refer to Σ as the input alphabet and σ as the proof alphabet of $\mathcal{D}$.

Definition 3.2 (Decodable PCPs). *For functions* $\delta : \mathbb{Z}^+ \to [0,1]$ *and* $\mathsf{L} : \mathbb{Z}^+ \to \mathbb{Z}^+$*, we say that a PCP decoder* $\mathcal{D}$ *is a* decodable probabilistically checkable proof (dPCP) system *for* $\textsc{CircuitSat}_\Sigma$ *with* soundness error δ *and* list size L *if the following completeness and soundness properties holds for every circuit* $C : \Sigma^k \to \{0,1\}$*:*

Completeness: *For any* $y \in \Sigma^k$ *such that* $C(y) = 1$ *there exists a proof* $\pi \in \sigma^\mathsf{m}$*, also called a decodable PCP, such that*

$$\Pr_{j,I,f}[f(\pi_I) = y_j] = 1$$

where $j \in [k]$ *is chosen uniformly at random and* I, f *are distributed according to* C, j *and the verifier's random coins.*

Soundness: *For any* $\pi \in \sigma^m$*, there is a list of* $0 \le \ell \le \mathsf{L}$ *strings* $y^1, \dots, y^\ell$ *satisfying* $\forall i,\ C(y^i) = 1$ *such that*

$$\Pr_{j,I,f}\left[f(\pi_I) \notin \left\{\perp, y^1_j, \dots, y^\ell_j\right\}\right] \le \delta, \tag{3.1}$$

Robust Soundness: *We say that* $\mathcal{D}$ *is a* robust dPCP system *for* $\textsc{CircuitSat}_\Sigma$ *with* robust soundness error δ*, if the soundness criterion in (3.1) can be strengthened to the following robust soundness criterion,*

$$\mathop{\mathbb{E}}_{j,I,f}\left[\operatorname{agr}\left(\pi_I, \textsc{bad}(f)\right)\right] \le \delta,$$

where

$$\textsc{bad}(f) \triangleq \left\{ w \in \sigma^{\mathsf{q}} \;\middle|\; f(w) \notin \left\{ \perp, y_j^1, \ldots, y_j^\ell \right\} \right\}.$$

Note that the parameters δ, L are allowed to be functions of the input length n, but not the input. As in the case of PCPs vs. robust PCPs, the only difference between a dPCP and a robust dPCPs is that the soundness condition for dPCP is $\Pr[\pi_I \in \textsc{bad}(f)] \leq \delta$ while that for robust dPCPs is $\mathbb{E}[\mathrm{agr}(\pi_I, \textsc{bad}(f))] \leq \delta$.

The above definition of dPCP can be naturally extended to any *pair* language, where the first part of the input should be viewed as the original input, and the second part as the NP witness (see Appendix A). However, for the purpose of composition it suffices to work with dPCPs for CircuitSat.

Decodable PCPs or its variants are implicit in most PCP constructions [AS03, RS97, DFK+99, BGH+06, DR06, MR07] and can be easily obtained by adapting the existing PCP constructions as we do in Section 6).

Decodable PCPs are very closely related to the locally decode/reject codes (LDRCs), introduced by Moshkovitz and Raz [MR08b] and can be viewed as a natural extension of their definition. The following summarizes the salient differences and similarities between these two objects.

1. LDRCs are a special case of dPCPs in the sense that LDRCs consider only those circuits C which check membership in a particular code (eg., Reed-Muller, Hadamard) while dPCPs consider any predicate C. This is the *main* difference between LDRCs and dPCPs. However, it is to be added that [MR08b] did not require such a general definition that works with any predicate C as they were interested in the composition of some very specific PCPs, while we need to work with the more general definition as we need to be able to compose arbitrary PCP verifiers.

2. LDRCs are defined in terms of the Label-Cover problem, while dPCPs are defined in the language of proofs, but this is merely a difference in the underlying language used (see Lemma 2.5).

3. LDRCs decode a k-tuple of elements from the proof while dPCPs decode just one symbol of the proof. However, the definition of dPCP can be extended from decoding symbols of the proof to decoding any function of the proof (and in particular k-tuples of the proof), as long as the set of functions to be decoded is known in advance (see Appendix A).

4 Composition Theorem

In this section, we show how to compose an outer robust PCP verifier with an inner robust PCP decoder, such that the resulting PCP verifier has *low* robust soundness. This gives a composition theorem for two-query PCPs simply by the equivalence between robust PCPs and two-query PCPs (see Lemma 2.5).

Before moving to our composition theorem, let us first explain why the earlier "natural" composition techniques [BGH+06, DR06, Sze99, RS97, AS03] did not give the result we claim here. As described in Section 1.2, the straightforward way to compose an outer robust PCP verifier V with an inner robust PCP decoder $\mathcal{D}$ is as follows. The composed PCP verifier V' begins by simulating V on a probabilistically checkable proof Π. It determines a set of queries into Π (a local window I), and a local predicate f. Instead of directly querying Π and testing if $f(\Pi_I) = 1$, V' relies on the inner PCP decoder $\mathcal{D}$ to perform this action. For this task, the inner PCP decoder $\mathcal{D}$ is supplied with a dedicated proof that is supposedly an encoding of the relevant local view Π_I. To ensure consistency (i.e. that the various dedicated proofs for $\mathcal{D}$ are encodings of local views coming from a single valid PCP for V) V' asks $\mathcal{D}$ to decode a value from the encoded local view, and compares it to the appropriate symbol in Π.

The problem is that the robust soundness of V' is always at least $1/2$, even if both V and $\mathcal{D}$ had very small robust soundness parameters. The reason is that the local view of V' has two distinct parts: the outer PCP part, and the inner dPCP part. Having fixed the view in one of the two parts, it is easy to modify the second part to make the verifier accept. Thus, by taking completely inconsistent inner dPCPs, still the average agreement of V' with an accepting view (namely, the robust soundness) is at least $1/2$, even if we allow for different weights on each part.

An alternate approach is to have V' check consistency by decoding the i-th symbol Π_i from two different randomly selected (encodings of) local views of Π, and avoiding the need for Π altogether. Here too the robust soundness is at least $1/2$, but now it is easy to correct: simply read Π_i simultaneously from many different local views, rather than just 2 ! This is the approach we describe next.

Theorem 4.1 (Composition Theorem)**.** *Suppose L has a regular*[6] *robust PCP verifier V with proof alphabet Σ and robust soundness error Δ, and* $\textsc{CircuitSat}_\Sigma$ *has a robust PCP decoder $\mathcal{D}$ with input alphabet Σ, robust soundness error δ and list size L. Then, L has a robust PCP verifier $V' = V \circledast \mathcal{D}$, with robust soundness error $\Delta\mathsf{L} + \delta$ and other parameters as stated in Figure 1. Furthermore, if the PCP decoder $\mathcal{D}$ is regular, then so is the composed verifier V'.*

	V	$\mathcal{D}$	$V' = V \circledast \mathcal{D}$
proof alphabet	Σ	σ	σ
randomness complexity	R	r	$\log M + \mathsf{r}$
query complexity	Q	q	Dq
decision complexity	S	s	$Ds + s(equal)$
proof degree	D	d	d
proof length	M	m	$2^{\mathsf{R}} \cdot \mathsf{m}$
robust soundness error	Δ	δ	$\Delta\mathsf{L} + \delta$
list size	-	L	-
input size	n	$S(n)$	-

Figure 1: Parameters for Composition. All parameters in the V column are functions of n, and all parameters in the $\mathcal{D}$ column are functions of $S(n)$. For example, $\Delta\mathsf{L} + \delta$ should be read as $\Delta(n) \cdot \mathsf{L}(S(n) + \delta(S(n)))$.

Proof. The proof $\pi \in \sigma^{2^{\mathsf{R}} \cdot \mathsf{m}}$ of the composed verifier V' is interpreted as a concatenation of the proofs $\pi(R)$ for each $R \in \{0,1\}^{\mathsf{R}}$. V' acts as follows:

[6] The composition theorem works even if the robust PCP is not regular as long as one works with a suitable weighted version of PCPs and chooses the probability distribution according to these weights instead of the uniform distribution. However, we find it easier to work with the regular case and not worry about weights. Lemma 5.9 contains a generic reduction transforming any non-regular PCP system into a regular one.

1. Choose $i \in [M]$ uniformly at random (recall that M is the length of the outer PCP). Let $R_1, \ldots, R_D$ be all the random strings of the outer verifier V that generate local windows $I_1, \ldots, I_D$ respectively such that $i \in I_k$ for every $k = 1, \ldots, D$. Denote by $f_1, \ldots, f_D : \Sigma^Q \to \{0,1\}$ the corresponding local predicates computed by V and by $j_1, \ldots, j_D \in [Q]$ the corresponding re-indexing of i within each I_k (i.e., $j_k = \text{index}_{i \in I_k}$ as defined in Section 2.1).

2. Choose $r \in \{0,1\}^{\mathsf{r}}$ uniformly at random. For each $k = 1, \ldots, D$ run the inner PCP decoder $\mathcal{D}$ on input f_k, index j_k, random coins r, and proof $\pi(R_k)$. Let (J_k, g_k) be the local window and local predicate computed by $\mathcal{D}$.

3. Accept if and only if
$$g_1(\pi(R_1)_{J_1}) = \cdots = g_D(\pi(R_D)_{J_D}) \neq \perp.$$
In other words the local window is $I' = \cup_{k=1}^{D} J_k$ and the local predicate $f' : \sigma^{Dq} \to \{0,1\}$ is defined by
$$f'(w_1, \ldots, w_D) = \begin{cases} 1, & g_1(w_1) = \cdots = g_D(w_D) \neq \perp \\ \\ 0, & \text{Otherwise.} \end{cases}$$

The claims about V''s parameters (randomness, query, decision complexities, proof length and proof degree) can be verified by inspection. Thus, we only need to check completeness and soundness.

Completeness: Suppose $x \in L$. Then, by completeness of V, there exists a proof Π causing V to accept with probability 1. In other words, for every $R \in \{0,1\}^{\mathsf{R}}$ and corresponding (I, f) computed by V, we have $f(\Pi_I) = 1$. We now invoke the inner PCP decoder $\mathcal{D}$ on the (input) circuit $f(R)$. By completeness of $\mathcal{D}$, there exists a proof $\pi(R)$ which encodes Π_I, causing $\mathcal{D}$ to always accept and output the correct symbol of Π. More specifically for each i and for every $r \in \{0,1\}^{\mathsf{r}}$, the verifier computes J and g such that $g(\pi(R)_J) = \Pi_i$. Since all the proofs $\pi(R)$ of the various inner verifiers encode different local views of *the same* outer proof Π, we have that the local view of the composed verifier satisfies the computed predicate f' with probability 1.

Soundness: Suppose that $x \notin L$. To prove soundness of the composed verifier V_{comp}, we need to show that for all proofs π,
$$\mathop{\mathbb{E}}_{(I',f') \sim V'}[\text{agr}\left(\pi_{I'}, (f')^{-1}(1)\right)] \leq \delta + \mathsf{L} \cdot \Delta.$$

Assume (for the purpose of contradiction) that this is not the case. In other words, there exists a proof $\pi \in \sigma^{\mathsf{m}2^{\mathsf{R}}}$, such that $\mathbb{E}_{(I',f') \sim V'}[\text{agr}\left(\pi_{I'}, (f')^{-1}(1)\right)] > \delta + \mathsf{L} \cdot \Delta$. We will then show that there exists a proof Π for the outer verifier V such that $\mathbb{E}_{(I,f) \sim V}\left[\text{agr}\left(\Pi_I, f^{-1}(1)\right)\right] > \Delta$, contradicting the soundness claim of the outer verifier V.

Let us write $\pi = (\pi(R))_{R \in \{0,1\}^{\mathsf{R}}}$. Fix some $R \in \{0,1\}^{\mathsf{R}}$, and let (I, f) be the local window and local predicate generated by the outer verifier V on input x and randomness R. Consider the inner PCP decoder $\mathcal{D}$ when run on the input f and the proof $\pi(R)$.

It follows from the soundness of $\mathcal{D}$, that for each $\pi(R)$ there exist a set
$$\text{list}(R) = \{y^1, y^2, \ldots, y^\ell\} \subseteq f^{-1}(1),$$
with $0 \leq \ell \leq \mathsf{L}$, of supposed "plausible" decodings of $\pi(R)$.

Let us recall the following notation from the description of V'. The random string of V' is $(i, r) \in [M] \times \{0,1\}^{\mathsf{r}}$. The pairs $(I_1, f_1), \ldots, (I_D, f_D)$ are such that $i \in I_k$ for all $1 \leq k \leq D$ and they are generated by the outer verifier on random strings $R_1, \ldots, R_D \in \{0,1\}^{\mathsf{R}}$ respectively. Furthermore, recall that $\{(J_k, g_k)\}_{k\in[D]}$ were the pairs generated by $\mathcal{D}$ in step 2, i.e., on input f_k, random string r, and index j_k where j_k is the re-indexing of i within I_k (i.e., $j_k = \text{index}_{i\in I_k}$). Finally, we denoted by (I', f') the local view and local predicate of V'.

We will view all of $I_k, f_k, R_k, J_k, g_k, f', I'$ as random variables over the probability space $[M] \times \{0,1\}^{\mathsf{r}}$ (i.e., that depend on i, r).

The following captures the set of *accepting* local views of V':

$$(f')^{-1}(1) = \left\{ w_1 w_2 \ldots w_D \in \sigma^{qD} \;\middle|\; g_1(w_1) = \cdots = g_D(w_D) \neq \bot \right\}.$$

Let $w = w_1 w_2 \cdots w_D \in (f')^{-1}(1)$ be an accepting view that is closest (in Hamming distance) to $\pi_{I'}$ (breaking ties lexicographically) and α the corresponding decoded value, i.e., $\alpha = g_1(w_1) = \cdots = g_D(w_D)$. Note that both w and α are random variables as well (i.e., $w = w(i, r)$ and $\alpha = \alpha(i, r)$). By assumption,

$$\mathop{\mathbb{E}}_{i,r}[\text{agr}(\pi_{I'}, w)] > \delta + \mathsf{L}\Delta. \tag{4.1}$$

Recall that $I' = \cup_{k=1}^{D} J_k$ so

$$\text{agr}(\pi_{I'}, w) = \mathop{\mathbb{E}}_{k\in[D]}\left[\text{agr}(\pi(R_k)_{J_k}, w_k)\right]. \tag{4.2}$$

Hence,

$$\mathop{\mathbb{E}}_{i,r}\mathop{\mathbb{E}}_{k\in[D]}\left[\text{agr}(\pi(R_k)_{J_k}, w_k)\right] > \delta + \mathsf{L}\Delta. \tag{4.3}$$

We will split the above expression according to whether or not α is "consistent" with the list $\text{list}(R_k)$. For each $k \in [D]$, let $c_k = c_k(i, r)$ be an indicator random variable defined by

$$c_k = \begin{cases} 1, & \alpha \in \text{list}(R_k)_{j_k} \\ 0, & \text{otherwise.} \end{cases}$$

Surely,

$$\begin{aligned} \mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, w_k)] &= \mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, w_k) \cdot c_k] + \mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, w_k) \cdot (1 - c_k)] && (4.4) \\ &\leq \mathop{\mathbb{E}}_{i,r,k}[c_k] + \mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, w_k) \cdot (1 - c_k)]. && (4.5) \end{aligned}$$

where the last inequality follows since $\text{agr}(\cdot) \leq 1$.

We will now upper bound the second quantity in the above expression by δ, the robust soundness of the inner PCP decoder $\mathcal{D}$. For each outer random string R, the soundness of the inner PCP decoder states that $\mathbb{E}_{r,j}[\text{agr}(\pi(R)_J, \textsc{bad}(g))] \leq \delta$, where $\textsc{bad}(g) = \{u \mid g(u) \notin \{\bot\} \cup \text{list}(R)_j\}$. Applying this to the outer random string R_k, we have

$$\mathop{\mathbb{E}}_{r,j}[\text{agr}(\pi(R_k)_{J_k}, \textsc{bad}(g_k))] \leq \delta,$$

and by regularity of the outer verifier V, also,

$$\mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, \textsc{bad}(g_k))] \leq \delta.$$

On the other hand, whenever $c_k = 0$, we have by definition that $\alpha \notin \text{list}(R_k)_{j_k}$ whereas $g_k(w_k) = \alpha \neq \bot$ which implies $w_k \in \text{BAD}(g_k)$. Hence we have

$$\mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, w_k) \cdot (1 - c_k)] \leq \mathop{\mathbb{E}}_{i,r,k}[\text{agr}(\pi(R_k)_{J_k}, \text{BAD}(g_k))] \leq \delta.$$

Combining the above inequality with (4.5) and (4.3), we have

$$\mathop{\mathbb{E}}_{i,r,k}[c_k] > \mathsf{L}\Delta.$$

Or equivalently,

$$\mathop{\mathbb{E}}_{r}\Pr_{i,k}[\alpha \in \text{list}(R_k)_{j_k}] > \mathsf{L}\Delta.$$

Recall that $\alpha = \alpha(i, r)$ was defined independently of k, and hence of R_k and yet, the above inequality shows that often α is consistent with the list-decoding $\text{list}(R_k)$ of the proof $\pi(R_k)$. This reveals that the soundness assumption of the composed verifier translates into an underlying *consistency* among the various $\text{list}(R_k)$'s. Stating the same in more words, we have that for average i and r, $\alpha = \alpha(i, r)$ often agrees with the list-decoding $\text{list}(R_k)_j$ where R_k is a random outer random string that involves i and j is the re-indexing of i within $I(R_k)$. This leads to the following definition of a (randomized) proof Π for the outer verifier V. Choose a random $r \in \{0,1\}^{\mathsf{r}}$ and set $\Pi_i = \alpha(i, r)$. In other words, Π_i is the common value decoded by the components of the closest accepting view to the local view $\pi_{I'}$ of the composed verifier V' on coin toss (i, r). From above, we have that this proof Π satisfies

$$\mathop{\mathbb{E}}_{r}\Pr_{R,i}[\Pi_i \in \text{list}(R)_j] > \mathsf{L}\Delta \tag{4.6}$$

where R is a uniformly chosen random string in $\{0,1\}^{\mathsf{R}}$, i a random location in $I(R)$ and j the re-indexing of i within $I(R)$ (i.e., $j = \text{index}_{i \in I(R)}$). However, any proof Π that satisfies (4.6) necessarily contradicts the soundness of the outer verifier V as seen from the following argument.

$$\begin{aligned}
\mathop{\mathbb{E}}_{r}\mathop{\mathbb{E}}_{R}[\text{agr}(\Pi_I, f^{-1}(1))] &\geq \mathop{\mathbb{E}}_{r}\mathop{\mathbb{E}}_{R}[\text{agr}(\Pi_I, \text{list}(R))] \\
&\geq \mathop{\mathbb{E}}_{r}\mathop{\mathbb{E}}_{R}\left[\mathsf{L}^{-1} \cdot \Pr_{j \in [|I|]}[(\Pi_I)_j \in \text{list}(R)_j]\right] \\
&= \mathop{\mathbb{E}}_{r}\mathop{\mathbb{E}}_{R}\left[\mathsf{L}^{-1} \cdot \Pr_{i \in I}[\Pi_i \in \text{list}(R)_j]\right] \quad [\text{where } j = \text{index}_{i \in I}] \\
&= \mathsf{L}^{-1} \cdot \mathop{\mathbb{E}}_{r}\Pr_{R,i}[\Pi_i \in \text{list}(R)_j] > \Delta
\end{aligned}$$

where the first inequality follows since $\text{list}(R) \subseteq f^{-1}(1)$, the second inequality is a consequence of Fact 2.1, and the rest follows by changing summation order and (4.6). This completes the proof of soundness of V'. □

5 Transformations on Label-Cover

In this section, we describe generic transformations on Label-Cover (or in its equivalent formulation, robust PCPs): degree reduction, alphabet reduction and regularization. To the best of our knowledge the alphabet reduction and regularization transformations are new, and may be of independent interest.

5.1 Degree Reduction

In this section, we show how we can lower the proof degree of a robust PCP at a very nominal cost to the other parameters. To this end, we use *expanders*, specifically we employ the *expander mixing lemma*. This proof is along the lines of degree reduction of LDRCs [MR08b], and is included for completeness.

Definition 5.1 (Expanders). *A d-regular graph $G = (V, E)$ on $n(= |V|)$ vertices is said to be a $[n, d, \lambda]$-expander if the second eigen-value of its normalized adjacency matrix is at most λ.*

Remark 5.2 (Ramanujan graphs). There exists a uniform algorithm (see [HLW06]) that when given as input n, $d > 3$, and λ construct a $[n, d, \lambda]$-expander in time polynomial in n and d as long as $1/\lambda = O(D^{\frac{1}{2}})$. (Such graphs are called *Ramanujan graphs.*)

Lemma 5.3 (expander mixing lemma). *Let $G = (V, E)$ be a $[N, d, \lambda]$-expander. Then for any two (possibly intersecting) sets $X, Y \subseteq V$, we have $|E(X, Y) - d|X||Y|/N| \leq \lambda \cdot \sqrt{|X||Y|}$.*

We state (and prove) degree reduction in the language of the Label-Cover problem (as opposed to that of robust PCPs) as this is a more natural setting (in our view) to perform degree reduction.

Theorem 5.4 (degree reduction). *Suppose $[n, d, \lambda]$-expanders are efficiently constructible (in polynomial time in the output length) where $\lambda = \lambda(d)$ as stated in Remark 5.2. Then there exists a polynomial time reduction transforming instances $I = (G = (U, V, E), \Sigma_1, \Sigma_2, F)$ of $\textsc{Label-Cover}_\delta$ of average right degree D to right-regular instances $I' = (G' = (U, V', E'), \Sigma_1, \Sigma_2, F')$ of $\textsc{Label-Cover}_{\delta+\lambda}$ of right degree d, such that $|V'| = D\,|V| = |E|$.*

Proof. Let $I = (G = (U, V, E), \Sigma_1, \Sigma_2, F)$ be the input instance with average right degree D. Let us assume some numbering of the neighbors of each left vertex, and denote the i-th neighbor of v by $\Gamma_G(v, i)$.

For each left vertex $v \in V$, let D_v denote the degree of v and let $H_v = ([D_v], E'')$ be a $[D_v, d, \lambda]$-expander as stated in the hypothesis of the theorem. Assume that the edges E'' of H are specified by the neighborhood function $\Gamma_{H_v} : [D_v] \times [d] \to [D_v]$, where $\Gamma_{H_v}(i, k)$ denotes the k^{th} neighbor of i.

The target instance I' is obtained by replacing each vertex $v \in V$ in the left vertex set by the vertices of the expander H_v. Thus, each vertex v is replaced by a cloud of vertices $[D_v]$ and V' is the disjoint union of $[D_v]$ for all $v \in V$. Clearly $|V'| = D\,|V|$. The edges E' and the constraints F' are specified below.

The edges E' of the (target) bipartite graph $G' = (U, V', E')$ are defined by defining the k-th neighbor of (v, i) in G' to be the j-th neighbor of v in G, where j itself is the k-th neighbor of i in H_v. In other words:

$$\Gamma_{G'}((v, i), k) \triangleq \Gamma_G\left(v, \Gamma_{H_v}(i, k)\right).$$

The constraints $F' = \{f_{e'} \mid e' \in E'\}$ of the target instance I' are defined as follows: for each edge $e' = (u, (v, i)) \in E'$, the corresponding edge $e = (u, v)$ is by definition an edge in E. The constraint $f_{e'} : \Sigma_1 \to \Sigma_2$ is then defined to be exactly the same as the corresponding function $f_e : \Sigma_1 \to \Sigma_2$.

The target instance $I' = (G', \Sigma_1, \Sigma_2, F')$, by definition, is right regular with right degree d. Furthermore, it is easy to see that if $L = (L_1 : U \to \Sigma_1, L_2 : V \to \Sigma_2)$ is a labeling of vertices in

G that satisfies all the edges in E, then the labeling $L' = (L_1 : U \to \Sigma_1, L_2' : V' \to \Sigma_2)$ given by $L_2'(v,i) = L_2(v)$ satisfies all the edges in E'. Thus, instances of value 1 are transformed to instances of value 1.

We only need to show that instances I of value at most δ are transformed to instances I' of value at most $\delta + \lambda$. Suppose (for contradiction) that the target instance I' has value greater than $\delta + \lambda$ while the input instance I has value less than or equal to δ. In other words, there exist labelings $L_1' : U \to \Sigma_1$ and $L_2' : V' \to \Sigma_2$ that satisfy more than $\delta + \lambda$ fraction of the edges in E'.

For each $v \in V$ and $\sigma \in \Sigma_2$, define sets $X_{v,\sigma}, Y_{v,\sigma} \subseteq [D_v]$ as follows.

$$\begin{aligned} X_{v,\sigma} &= \left\{ i \in [D_v] \,\middle|\, L_2'(v,i) = \sigma \right\} \\ Y_{v,\sigma} &= \left\{ i \in [D_v] \,\middle|\, f_{(u,v)}(L_1'(u)) = \sigma \text{ where } u \text{ is defined by } u = \Gamma_G(v,i) \right\}. \end{aligned}$$

Thus, for each $v \in V$, the sets $\{X_{v,\sigma} \mid \sigma \in \Sigma_2\}$ corresponds to the partition of the cloud of vertices $[D_v]$ based on the labeling L_2' while the sets $\{Y_{v,\sigma} \mid \sigma \in \Sigma_2\}$ corresponds to the partition of the cloud of vertices $[D_v]$ based on the opinion (i.e., value of the function f) along the edge $(\Gamma_G(v,i), v)$

It follows from a counting argument, that the fraction of edges satisfied by the labeling L' is given by the following expression, which by assumption is greater than $\delta + \lambda$.

$$\frac{1}{|E'|} \sum_{v \in V} \sum_{\sigma \in \Sigma_2} E(X_{v,\sigma}, Y_{v,\sigma}).$$

We now define a (randomized) labeling L for the input instance I as follows: for each $u \in U$, set $L_1(u) = L_1'(u)$ and for each $v \in V$, choose $i \in [D_v]$ randomly and set $L_2(v) = L_2'(v,i)$. The fraction of edges satisfied by this labeling is given by the following expression.

$$\frac{1}{|E'|} \sum_{v \in V} \sum_{\sigma \in \Sigma_2} |X_{v,\sigma}| \cdot |Y_{v,\sigma}| \frac{d}{D_v}$$

Since this quantity lower bounds the value of the instance I, we have

$$\begin{aligned} \text{value}(I) &\geq \frac{1}{|E'|} \sum_{v \in V} \sum_{\sigma \in \Sigma_2} |X_{v,\sigma}| \cdot |Y_{v,\sigma}| \frac{d}{D_v} \\ &\geq \frac{1}{|E'|} \sum_{v \in V} \sum_{\sigma \in \Sigma_2} \left(E(X_{v,\sigma}, Y_{v,\sigma}) - \lambda \cdot \sqrt{|X_{v,\sigma}| \cdot |Y_{v,\sigma}|} \right) \\ &> (\delta + \lambda) - \frac{\lambda}{|E'|} \cdot \sum_{v \in V} \sum_{\sigma \in \Sigma_2} \sqrt{|X_{v,\sigma}|} \cdot \sqrt{|Y_{v,\sigma}|} \\ (Cauchy - Schwarz) \quad &\geq (\delta + \lambda) - \frac{\lambda}{|E'|} \cdot \sum_{v \in V} \sqrt{\sum_{\sigma \in \Sigma_2} |X_{v,\sigma}|} \cdot \sqrt{\sum_{\sigma \in \Sigma_2} |Y_{v,\sigma}|} \\ &= (\delta + \lambda) - \lambda = \delta \end{aligned}$$

The inequality in the second line follows from the expander mixing lemma, and the last equality follows from $\sum D_v = |E'|$. Thus, value of the input instance I is greater than δ, which is a contradiction. Hence, instances I of value at most δ are transformed to instances I' of value at most $\delta + \lambda$.

□

5.2 Alphabet Reduction

In this section, we show how to reduce the proof alphabet of the robust PCP at a nominal cost to other parameters. First we need some preliminaries from coding theory.

A mapping $\mathcal{C} : \Sigma \to \sigma^k$ is called a code with minimum relative distance $1 - \delta$ if for every $a \neq b \in \Sigma$ the strings $\mathcal{C}(a)$ and $\mathcal{C}(b)$ differ in at least $(1-\delta)k$ coordinates.

Fact 5.5. *Suppose $\mathcal{C} \subseteq \sigma^k$ is a code with (relative) distance at least $1 - \delta$ and $\eta > 2\sqrt{\delta}$. Then, there are at most $2/\eta$ codewords in $\mathcal{C}$ that agree with a given word w on at least η fraction of the coordinates.*

Proof. Suppose there exist a word w and a list of $l = \lfloor 2/\eta \rfloor + 1$ codewords in $\mathcal{C}$ that agree with w on at least η fraction of the locations. Then, by inclusion exclusion,

$$\Pr_i[\exists c \in \text{list}, w_i = c_i] \geq \sum_{c \in \text{list}} \Pr_i[w_i = c_i] - \sum_{c_1 \neq c_2 \in \text{list}} \Pr_i[w_i = (c_1)_i = (c_2)_i] \geq l\eta - \binom{l}{2}\delta.$$

Since $\eta > 2\sqrt{\delta}$, the above expression is greater than 1 for every $l \in [2/\eta, 2/\eta + 1]$ and in particular for $l = \lfloor 2/\eta \rfloor + 1$, which is a contradiction. □

Remark 5.6. For every $0 < \delta < 1$ and alphabet Σ, there exists a code $\mathcal{C} : \Sigma \to \sigma^k$ with relative distance $1 - \delta$ where $|\sigma| = O(1/\delta^2)$ and $k = O(\log|\Sigma|/\delta^2)$.

Such codes can be constructed by concatenating the following two codes. As outer codes, we take the rate optimal codes of [ABN+92] with relative distance $1 - \delta/2$, rate $\Omega(\delta)$ and alphabet size $2^{O(1/\delta)}$. As inner codes, we take Reed-Solomon codes with relative distance $1 - \delta/2$, rate δ, and alphabet $O(1/\delta^2)$.

Theorem 5.7 (alphabet reduction)**.** *Suppose $\mathcal{C} : \Sigma \to \sigma^k$ is a code with (relative) distance $1 - \eta^3$ for some $\eta < 1/4$. Then there exists a polynomial time reduction transforming instances $I = (G = (U, V, E), \Sigma', \Sigma, F)$ of* Label-Cover$_\delta$ *to instances $I' = (G' = (U, V \times [k], E'), \Sigma, \sigma, F')$ of* Label-Cover$_{\delta+3\eta}$.

Proof. The reductions maps instances $I = (G = (U, V, E), \Sigma', \Sigma, F)$ to instances $I' = (G' = (U, V \times [k], E'), \Sigma, \sigma, F')$ where the set of edges E' and the set of projections F' are defined as follows: $E' = \{(u, (v, i)) \mid (u, v) \in E, i \in [k]\}$ and for each $e = (u, (v, i)) \in E$, the function $f'_e : \Sigma' \to \sigma$ is defined as $f'_e(\alpha) = \mathcal{C}(f_{(u,v)}(\alpha))_i$.

Completeness is easy as instances of value 1 are mapped to instances of value 1. We will prove soundness by showing that value$(I') \leq$ value$(I) + 3\eta$. To this end, let $\Pi' = (\pi_1 : U \to \Sigma', \pi'_2 : V \times [k] \to \sigma)$ be any labeling of the target instance I'. For each $v \in V$ and $\beta \in \Sigma$, define $\delta_v(\beta)$ as follows:

$$\delta_v(\beta) = \Pr_{u \in \Gamma(v)} \left[f_{(u,v)}(\pi_1(u)) = \beta \right].$$

It can be easily checked that the fraction of edges satisfied by the labeling Π' is $\mathbb{E}_{v \in V}[\delta'_v]$ where δ'_v is the following expression:

$$\delta'_v = \sum_{\beta \in \Sigma} \delta_v(\beta) \cdot \Pr_i \left[\pi'_2(v, i) = \mathcal{C}(\beta)_i \right].$$

Now, define a labeling $\Pi = (\pi_1 : U \to \Sigma', \pi_2 : V \to \Sigma)$ for the instance I by taking the same π_1 and setting $\pi_2(v) = \mathrm{argmax}_\beta\, \delta_v(\beta)$. Clearly, the expected fraction of edges satisfied by Π is $\mathbb{E}_{v \in V}[\delta_v]$ where $\delta_v = \max_\beta \delta_v(\beta)$.

Since Π is a labeling of I, we have $\mathbb{E}[\delta_v] \leq \mathrm{value}(I)$. Thus, to show $\mathrm{value}(I') \leq \mathrm{value}(I) + 3\eta$, it suffices to show that for each $v \in V$, we have $\delta'_v \leq \delta_v + 3\eta$. Fix a vertex $v \in V$. Define $\mathrm{list}(v) = \{\beta \in \Sigma \,|\, \Pr_i[\pi'_2(v,i) = \mathcal{C}(\beta)_i] \geq \eta\}$. From Fact 5.5, we have that $l = |\mathrm{list}(v)| \leq 2/\eta$ since $\eta > 2\eta^{3/2}$ for $\eta < 1/4$. We now, have

$$\begin{aligned}
\delta'_v &= \sum_{\beta \in \mathrm{list}(v)} \delta_v(\beta) \cdot \Pr_i\left[\pi'_2(v,i) = \mathcal{C}(\beta)_i\right] + \sum_{\beta \notin \mathrm{list}(v)} \delta_v(\beta) \cdot \Pr_i\left[\pi'_2(v,i) = \mathcal{C}(\beta)_i\right] \\
&< \left(\max_{\beta \in \mathrm{list}(v)} \delta_v(\beta)\right) \cdot \sum_{\beta \in \mathrm{list}(v)} \Pr_i\left[\pi'_2(v,i) = \mathcal{C}(\beta)_i\right] + \left(\eta \cdot \sum_{\beta \notin \mathrm{list}(v)} \delta_v(\beta)\right) \\
&\leq \left(\delta_v \cdot \sum_{\beta \in \mathrm{list}(v)} \Pr_i\left[\pi'_2(v,i) = \mathcal{C}(\beta)_i\right]\right) + \eta \\
&\leq \delta_v \cdot \left(\Pr\left[\exists \beta \in \mathrm{list}(v), \pi'_2(v,i) = \mathcal{C}(\beta)_i\right] + \sum_{\beta_1 \neq \beta_2 \in \mathrm{list}(v)} \Pr\left[\pi'_2(v,i) = \mathcal{C}(\beta_1)_i = \mathcal{C}(\beta_2)_i\right]\right) + \eta \\
&\leq \delta_v \cdot \left(1 + \sum_{\beta_1 \neq \beta_2 \in \mathrm{list}(v)} \Pr\left[\mathcal{C}(\beta_1)_i = \mathcal{C}(\beta_2)_i\right]\right) + \eta \\
&\leq \delta_v + \binom{l}{2}\eta^3 + \eta \\
&\leq \delta_v + 3\eta
\end{aligned}$$

The inequality in the third line is due to the fact $\sum_\beta \delta_v(\beta) = 1$, while the inequality in the fourth line is due to the principle of inclusion-exclusion: $\Pr[\exists \beta, A(\beta)] \geq \sum_\beta \Pr[A(\beta)] - \sum_{\beta_1 \neq \beta_2} \Pr[A(\beta_1) \cap A(\beta_2)]$. The inequality in the sixth line follows from the fact that the distance of the code $\mathcal{C}$ is at least $1 - \eta^3$ and the the last line follows from $l \leq 2/\eta$. Thus, proved.

□

5.3 Reduced Verifier

In this section, we summarize the combined actions of performing alphabet reduction and degree reduction on a Label-Cover instance. Let $I = ((U,V,E), \Sigma', \Sigma, F)$ be a label cover instance with $|U| = n, |V| = m$, the average left degree D_A and the average right degree D_B. Figure 2 summarizes the evolution of parameters of the Label-Cover instance when we perform first an alphabet reduction using a code $\mathcal{C} : \Sigma \to \sigma^k$ with distance $1 - \eta^3$ and then a degree reduction using a $[D, d, \lambda]$-expander.

Let $\varepsilon : \mathbb{Z}^+ \to [0,1]$ be any function. For the alphabet reduction, we take $\eta = \varepsilon/4$ and use the codes mentioned in Remark 5.6, with relative distance $1 - (\varepsilon/4)^3$, alphabet size $|\sigma| = O(1/\varepsilon^6)$, and $k = O(\log|\Sigma|/\varepsilon^6)$. For the degree reduction, we take $\lambda = \varepsilon/4$, $d = O(1/\varepsilon^2)$, and use $[D_B, O(1/\varepsilon^2), \varepsilon/4]$-expander, which are constructible for all D_B (see Remark 5.2).

We summarize the outcome transformation in the following lemma (stated in the language of robust PCPs).

Label-Cover	(robust PCP)	I	alphabet $\to \sigma$	degree $\to d$
Number of left vertices	(randomness)	n	n	n
Number of right vertices	(proof length)	m	mk	mkD_B
Left Degree	(query complexity)	D_A *	$D_A k$ *	$D_A kd$ *
Right Degree	(proof degree)	D_B *	D_B *	d
Left Alphabet	(num. accepting. conf.)	Σ'	Σ'	Σ'
Right Alphabet	(proof alphabet)	Σ	σ	σ
Soundness error	(robust soundness error)	δ	$\delta + 3\eta$	$\delta + 3\eta + \lambda$

Figure 2: The evolution of parameters on performing alphabet reduction followed by degree reduction. An asterisk (*) indicates that the corresponding instance is not necessarily regular (right or left-regular as the case may be) and the quantity mentioned is the average degree.

Lemma 5.8 (degree and alphabet reduced robust PCP)**.** *There exists a constant $C > 0$ such that for all $\varepsilon : \mathbb{Z}^+ \to [0,1]$, the following holds. Suppose L has a robust PCP Verifier V with randomness complexity r, query complexity q, proof length m, proof degree d, robust soundness error δ over a proof alphabet Σ. Then, L has a reduced robust PCP verifier, which we shall denote by $\mathrm{red}_\varepsilon(V)$ with*

- *randomness complexity r,*
- *query complexity $Cq \log |\Sigma| / \varepsilon^8$,*
- *proof length $Cmd \log |\Sigma| / \varepsilon^6$,*
- *proof degree C/ε^2,*
- *proof alphabet σ of size at most C/ε^6,*
- *and robust soundness error $\delta + \varepsilon$.*

5.4 Regularizing a Label Cover Instance

In this section, we show how to regularize a given Label-Cover instance. Since the degree reduction transformation from Section 5.1 makes the graph right-regular, it can be used to make a label cover regular on both sides. Given a Label-Cover instance $I = ((U,V,E),\Sigma',\Sigma,F)$ with $|U| = n, |V| = m$, average left degree D_A and average right degree D_B, we perform the following steps:

1. Degree reduction to make the graph right-regular, with right degree d.
2. Flip sides: This is the only step not already described above. It amounts to mapping the right-d-regular instance $I = ((U,V,E),\Sigma',\Sigma,F)$ to $I = ((V,U,E),(\Sigma')^d,\Sigma',F')$. Note that the underlying graph is almost unchanged, merely U and V are swapped. The constraints F' are as follows. The value $(a_1,\dots,a_d) \in (\Sigma')^d$ assigned to a vertex v is interpreted as an assignment to all of its neighbors in the original instance. The constraint on an edge (v,u) checks that there is some $b \in \Sigma$ that, together with $(a_1,\dots,a_d)$ would have satisfied the edges $(v,u_1),\dots,(v,u_d)$ coming out of v. It also checks that the value actually given to u

is consistent with the appropriate a_i. Soundness and completeness are straightforward. At the end of this operation, the instance is left-regular with left degree d and the average right degree is the earlier average left degree $D_A d$.

3. Degree reduction to make the graph right-regular, with right degree d.

4. Alphabet reduction to reduce alphabet Σ' to σ.

The evolution of the parameters over the four steps is summarized in Figure 3.

	I	degree $\to d$	flip	degree $\to d$	alphabet $\to \sigma$
Number of left vertices	n	n	mD_B	mD_B	mD_B
Number of right vertices	m	mD_B	n	nD_Ad	nD_Adk
Left Degree	D_A *	D_Ad *	d	d^2	d^2k
Right Degree	D_B *	d	D_Ad *	d	d
Left Alphabet	Σ'	Σ'	$(\Sigma')^d$	$(\Sigma')^d$	$(\Sigma')^d$
Right Alphabet	Σ	Σ	Σ'	Σ'	σ
Soundness error	δ	$\delta+\lambda$	$\delta+\lambda$	$\delta+2\lambda$	$\delta+2\lambda+3\eta$

Figure 3: The figure describes the evolution of parameters through the four steps: degree reduction, flipping sides, degree reduction, and alphabet reduction. An asterisk (*) indicates that the corresponding instance is not necessarily regular (right or left-regular as the case may be) and the quantity mentioned is the average degree.

The following lemma summarizes these parameter choices. For the degree reduction, we plug in $\lambda = \varepsilon/5$ and $d = O(1/\varepsilon^2)$ using the $[D, O(1/\varepsilon^2), \varepsilon/5]$-expanders from Remark 5.2. For the alphabet reduction, we use the codes mentioned in Remark 5.6, with $\eta = \varepsilon/5$, distance $1 - O(\varepsilon^3)$, $|\sigma| = O(1/\varepsilon^6)$ and $k = O(1/\varepsilon^6) \cdot \log|\Sigma'| \leq O(1/\varepsilon^6) \cdot q \log|\Sigma|$.

Lemma 5.9 (regularized robust PCP). *There exists a constant $C > 0$ such that for all $\varepsilon : \mathbb{Z}^+ \to [0,1]$, the following holds. Suppose L has a robust PCP Verifier V with randomness complexity r, query complexity q, proof length m, average proof degree d, robust soundness error δ over a proof alphabet Σ. Then, L has a regular reduced robust PCP verifier, which we shall denote by* $\text{regular}_\varepsilon(V)$ *with*

- *randomness complexity* $\log m + \log d$,
- *query complexity* $Cq \log|\Sigma|/\varepsilon^{10}$,
- *proof length* $Cq^2 2^r \log|\Sigma|/\varepsilon^8$,
- *proof degree* C/ε^2,
- *proof alphabet σ of size at most* C/ε^6,
- *and robust soundness error* $\delta + \varepsilon$.

6 Proof of Result of [MR08b]

In this section, we give our proof for the result of [MR08b], namely Theorem 1.3. We first give a more formal statement of this theorem, both in the language of Label-Cover as well as robust PCPs.

Theorem 6.1 (Formal version of Theorem 1.3)**.** *There exists constants $c > 0$ and $0 < \beta < 1$, such that for every function $1 < g(n) \leq 2^{O(\log^\beta n)}$, the following (equivalent) statements hold:*

- *There exists an alphabet Σ of size $\exp(g(n)^c)$ such that* $\text{Label-Cover}_{1/g(n)}$ *over Σ is NP-hard, even under nearly length preserving reductions. Furthermore, the size of the* Label-Cover *instance produced by this reduction is at most $n \cdot 2^{O(\log^\beta n)} \cdot g(n)^c$.*

- CircuitSat *has a robust PCP verifier with robust soundness error $1/g(n)$, query complexity $g(n)^c$, proof length $n^{1+o(1)}$ and randomness complexity $\log n + O(\log^\beta n)$.*

Theorem 6.1 is slightly stronger than the version (Theorem 1.3) stated in the introduction in the sense that it works for the range $1 < g(n) \leq 2^{O(\log^\beta n)}$ and not just $1 < g(n) \leq \text{polylog} n$ as indicated in the introduction. This stronger version is true both of our proof as well as that of [MR08b].

We construct the robust PCP verifier stated in the theorem by repeatedly composing two building blocks, both based on the "Manifold vs. Point" PCP (Theorem 1.2). We describe the building blocks next, and prove the theorem in the following section. The equivalent Label-Cover formulation follows from the equivalence lemma 2.5.

Before we proceed to the proof of the theorem, a couple of remarks regarding the tightness (or lack thereof) of the parameters in this theorem are in order.

Remark 6.2.

- As discussed in the introduction, the relation $\delta = 1/g(n)$ and $|\Sigma| = \exp(\text{poly}(g(n)))$ in the Label-Cover instance is exponentially worse than what is known, for example, for PCPs with $O(1)$ queries.

- Although the verifier is randomness-efficient, still, the relation between the randomness complexity and the soundness error is not "optimal". One could hope for proof length of $n \cdot \text{poly}(g(n))$, which comes closer to the following easy lower bound of $n \cdot \Omega(g(n))$:

  Claim 6.3. *If* $\text{Label-Cover}_\delta$ *is NP-hard, then the produced instance size must be at least $O(n/\delta)$ where n is the size of the shortest NP-witness for* CircuitSat*.*

  The claim holds because if D is the average left degree, it is easy to see that it is always possible to satisfy $O(1/D)$ fraction of edges (one neighbor per $v \in V$), so the proof degree is at least $\Omega(1/\delta)$. On the other hand, the number of left vertices which comprises the PCP, which being a NP-witness itself, is of size at least n (Note if $NP = P$, then $n = 0$). Thus, the total number of edges, which is nD is at least $\Omega(n/\delta)$.

  We wonder whether a result of $n \cdot \text{poly}(g(n))$ is attainable.

6.1 Building Blocks

The two building blocks, we need for our construction, are a robust PCP and a decodable PCP. Both are constructed from variants of the 'Manifold vs. Point' PCP of Theorem 1.2.

6.1.1 The outer (robust) PCP verifier

Theorem 6.4 (Robust PCP)**.** *There exist constants $b_0, b_1, b_2, b_3 > 0$ and $0 < \beta < 1$ such that for $\varepsilon = 1/2^{b_0 \log^\beta n}$, $\textsc{CircuitSat}$ has a robust verifier with robust soundness error ε, query complexity $1/\varepsilon^{b_1}$, proof length $n \cdot 1/\varepsilon^{b_2}$ randomness complexity $\log n + b_2 \log \frac{1}{\varepsilon}$, and proof alphabet size at most $1/\varepsilon^{b_3}$.*

This theorem follows from a combination of known results. We do not provide a complete proof of this theorem, rather an outline of how it is constructed (with pointers to the appropriate known results).

1. **Basic low-error PCP:** Construct a PCP verifier based on the low degree extension over a field $\mathbb{F}$ and the sum-check protocol (as done in [AS98, ALM⁺98, RS97, AS03]). We only need the "basic part" of the construction, i.e. without performing composition at all. A randomness efficient version of this PCP is given in [MR07].

   At this point, the proof oracle has three parts. A "points table" describing a function $f : \mathbb{F}^m \to \mathbb{F}$ supposedly of low degree, a "planes table", supposedly describing the restriction of f to affine planes, and a "curves" table supposedly describing the restriction of f to certain degree d curves.

   The soundness of the verifier says that for every $\varepsilon \geq 1/|\mathbb{F}|^\gamma$ and $\ell = 2/\varepsilon$ the following holds (where $\gamma > 0$ is some absolute constant).

   Soundness: For every function $f : \mathbb{F}^m \to \mathbb{F}$, there exists a list of low degree functions $P^1, \ldots, P^\ell : \mathbb{F}^m \to \mathbb{F}$ such that each P^i is an honest encoding of a legal NP witness for the original $\textsc{CircuitSat}$. In addition, the probability that V accepts even though its queries (point, plane, or curve) disagree with the list $P^1, \ldots, P^\ell$ is at most ε. (In other words, except with probability ε, V either rejects or only accepts values that are consistent with a short list of encodings of valid NP witnesses).

2. **Manifold vs. Point:** It has been "folklore", and formally described in [MR08b] that the plane and the curve queries can be combined into one "manifold" query (where the manifold is the $O(1)$ dimensional manifold containing both the curve and the plane). Now the planes and curves tables are replaced by a single manifold table. This transformation is even possible with small randomness complexity, and we refer the reader to Lemma 8.2 in [MR08b]. (While this theorem constructs an LDRC, it is easy to transform them into PCPs).

3. **Robustness:** The conversion to a robust verifier (from a "manifold vs. point" one) is straightforward, as in Lemma 2.5: the proof now only consists of the function f, and the verifier randomly selects a manifold and reads every point on the manifold (accepting iff the point values are consistent with an accepting value for the entire manifold).

4. **Parameters:** The above construction in general works for a wide range of parameter choices. The randomness efficient version due to [MR07] requires $|\mathbb{F}| = 2^{O((\log n)^\beta)}$ for some β, so we

follow this setting. Both ε and the query complexity are constant powers of $|\mathbb{F}|$, so we choose b_0 small enough and b_1 large enough.

It is to be noted that the randomness-efficient construction of the above robust PCP is *not regular*, actually it is not regular in a very mild sense. We get around this by first regularizing the verifier using the generic regularization transformation stated in Lemma 5.9.

6.1.2 The inner PCP decoder

Theorem 6.5 (dPCP)**.** *There exist constants $a_1, a_2, \alpha, \gamma > 0$ such that for every $\delta \geq n^{-\alpha}$ and input alphabet Σ of size at most n^γ,* CircuitSat$_\Sigma$ *has a robust decodable PCP system with robust soundness error δ and list size $\mathsf{L} \leq 2/\delta$, query complexity $n^{1/8}$, proof alphabets n^γ, proof length n^{a_1} and randomness complexity $a_2 \log n$.*

The construction of such a dPCP is very similar to the construction of the robust verifier from Theorem 6.4 above. The two modifications are as follows.

- First of all, we need to construct a PCP decoder $\mathcal{D}$, rather than a PCP verifier. This means that in addition to the regular input, the decoder also receives an *index* into the original proof (the NP witness) that needs to be decoded. Observe that in the basic PCP described in step 1 above the function f is (by construction) an encoding of the original NP witness in the sense that the restriction of f to certain points in $\mathbb{F}^m$ is the supposed NP witness. So, viewing the input index as a point $x \in \mathbb{F}^m$ all we need is, in addition to the verification, to return the value of $f(x)$. This is done by modifying the manifold to also contain this point x (thereby increasing its dimension by 1). Thus, for each input point x we have a separate collection of manifolds, all of which contain x. By the soundness condition described above we know that for every proof $f : \mathbb{F}^m \to \mathbb{F}$, there is a list of at most $\ell \leq 2/\delta$ valid low degree encodings $P^1, \ldots, P^\ell$ such that the probability that $\mathcal{D}$ does not reject but answers inconsistently with all of $P^1, \ldots, P^\ell$ is at most δ.

- It is to be noted that even though the non-randomness efficient robust PCP verifier described in the earlier section is regular, the PCP decoder is *not regular* because of the bias towards the input points x. One can get around this irregularity by either querying all points in the manifold but for the input point x or by weighting the input and proof points suitably. We can thus assume that the constructed PCP decoder is, in fact, regular.

- The second modification is to the parameters. For this theorem we choose $|\mathbb{F}| = n^\gamma$ for small enough γ so that the query complexity is at most $n^{1/8}$ (recall that it is a fixed power of $|\mathbb{F}|$). This in turn determines α, a_1, a_2. Observe that the proof alphabet is equal to $\mathbb{F}$, which is of size n^γ. Furthermore, note that the PCP decoder can handle any input alphabet as long as its size is at most that of the field $\mathbb{F}$, which is n^γ.

6.2 Putting it Together

Let $\mathcal{D}$ be the PCP decoder from Theorem 6.5, and let V be the robust PCP from Theorem 6.4 with robust soundness error $\varepsilon = 2^{O(\log^\beta n)}$, query complexity $1/\varepsilon^{O(1)}$, randomness complexity $\log n + O(\log 1/\varepsilon)$ and proof length $n \cdot (1/\varepsilon)^{O(1)}$.

Lemma 6.6. *Let $\mathcal{D}, V, \varepsilon$ be as defined above and set $\varepsilon_i = (\varepsilon)^{1/3^i}$. There exist constants $c_0, c_1, c_2, c_3 > 0$ such that for every $i \geq 0$ as long as $\varepsilon_i < c_0$, the following holds.* CircuitSat *has a regular robust PCP verifier V_i with query complexity $1/{\varepsilon_i}^{c_1}$, robust soundness error $2\varepsilon_i$, proof degree $c_3/{\varepsilon_i}^2$, proof alphabet size $c_3/{\varepsilon_i}^6$, randomness complexity $\log n + c_2 \sum_{j=0}^{i} \log 1/\varepsilon_j$ and proof length $n \cdot (\prod_{j=0}^{i} 1/\varepsilon_j)^{c_2}$.*

Proof of lemma. For $i = 0$ the claim follows by taking $\varepsilon_0 = \varepsilon$ and setting $V_0 = \mathrm{regular}_\varepsilon(V)$ for V as in the hypothesis, where $\mathrm{regular}_\varepsilon(V)$ is defined according to Lemma 5.9. Note that this process, regularizes V as the robust PCP verifier V from Theorem 6.4 is not necessarily regular. By choosing c_1, c_2, c_3 large enough the inductive hypothesis is established. Assume that the claim holds for all $i \geq 0$, and let us prove it for $i + 1$. Define

$$V_{i+1} = \mathrm{red}_{\varepsilon_{i+1}}(V_i \circledast \mathcal{D})$$

where $V_i \circledast \mathcal{D}$ stands for the verifier that results from composing V_i with $\mathcal{D}$ as in Theorem 4.1.

We first check that V_{i+1} is well defined and then compute its parameters. Composition requires that both V_i and $\mathcal{D}$ are regular; the former is by the inductive hypothesis and the latter by construction. Hence, both the composed verifier $V_i \circledast \mathcal{D}$ and the reduced verifier $V_{i+1} = \mathrm{red}_{\varepsilon_{i+1}}(V_i \circledast \mathcal{D})$ are also regular. The composition is defined as long as the input alphabet of $\mathcal{D}$ is large enough to be able to encode a symbol from the proof alphabet of V_i. The input size on which $\mathcal{D}$ is run is

$$N = \text{quasi linear}(1/{\varepsilon_i}^{c_1}) \leq (1/\varepsilon_i)^{c_1+1} = (1/\varepsilon_{i+1})^{3(c_1+1)},$$

where we denote quasi linear$(m) = m \cdot \mathrm{poly}\log m$. It will be convenient to assume that $N = (1/\varepsilon_{i+1})^{3(c_1+1)}$ by padding the input. The input alphabet is $N^\gamma = (1/\varepsilon_i)^{\gamma(c_1+1)}$. On the other hand, the proof alphabet of V_i is $c_3/{\varepsilon_i}^6$. This works out as long as $c_3/{\varepsilon_i}^6 \leq 1/{\varepsilon_i}^{\gamma(c_1+1)}$, which for sufficiently small ε_i is true if $6 < \gamma(c_1 + 1)$ which is settled by taking c_1 large enough.

The transformation of Lemma 5.8 gives the required alphabet size and proof degree. We now calculate the remaining parameters.

- Soundness error: Let us first compute the soundness error of $V_i \circledast \mathcal{D}$. It is

  $$\delta + \mathsf{L}\Delta = \delta + \frac{2}{\delta} \cdot 2\varepsilon_i$$

  where we can choose any $\delta = \varepsilon(N) \geq N^{-\alpha} = {\varepsilon_i}^{\alpha(c_1+1)}$. We will bound each term by $\varepsilon_{i+1}/2$, which is equivalent to $8{\varepsilon_i}^{2/3} \leq \delta \leq {\varepsilon_i}^{1/3}/2$. Such a δ exists if $N^{-\alpha} = {\varepsilon_i}^{\alpha(c_1+1)} \leq {\varepsilon_i}^{1/3}/2$, and this holds if $3\alpha(c_1 + 1) > 1$ for sufficiently small ε_i. Applying the transformation of Lemma 5.8 increases the soundness error by another ε_{i+1} which results in $2\varepsilon_{i+1}$.

- Query complexity: For $V_i \circledast \mathcal{D}$ the query complexity is the proof degree of V_i multiplied by the query complexity of $\mathcal{D}$, thus, it is,

  $$c_3/{\varepsilon_i}^2 \cdot N^{1/8} = c_3/{\varepsilon_i}^2 \cdot (1/\varepsilon_i)^{(c_1+1)/8} = c_3(1/\varepsilon_{i+1})^{3\cdot(2+(c_1+1)/8)}$$

  Now, after reducing according to Lemma 5.8, the query complexity of V_{i+1} is multiplied by $C \log |\Sigma|/{\varepsilon_{i+1}}^8$ where $|\Sigma|$ is the size of the proof of alphabet of $V_i \circledast \mathcal{D}$, which is N^γ, Thus, the new query complexity is $Cc_3\gamma(1/\varepsilon_{i+1})^{3(2+(c_1+1)/8)+8} \log N = 3Cc_3(c_1+1)\gamma(1/\varepsilon_{i+1})^{3(2+(c_1+1)/8)+8} \log(1/\varepsilon_{i+}$ Altogether, this is less than $(1/\varepsilon_{i+1})^{c_1}$ if ε_{i+1} is sufficiently small and $c_1 > 3(2+(c_1+1)/8)+8 = \frac{113}{8} + \frac{3}{8}c_1$, or $c_1 > 113/5$.

- Proof length: The proof length of $V_i \circledast \mathcal{D}$ is equal the number of possible random strings for V_i multiplied by the proof length of $\mathcal{D}$, so it is

  $$2^{\log n + c_2 \sum_{j=0}^{i} \log 1/\varepsilon_j} \cdot N^{a_1} = n \cdot \prod_{j=0}^{i} (1/\varepsilon_j)^{c_2} \cdot N^{a_1}$$

  After applying the 'Reduce' transformation of Lemma 5.8, the proof length increases by a factor corresponding to the degree of $\mathcal{D}$ times $C \log N^{\gamma} / {\varepsilon_{i+1}}^6$. Trivially bounding the degree of $\mathcal{D}$ by the number of possible random strings which is N^{a_2}, we get a bound of

  $$n \cdot \prod_{j=0}^{i} (1/\varepsilon_j)^{c_2} \cdot N^{a_1} \cdot N^{a_2} \cdot C \log(N^{\gamma}) / {\varepsilon_{i+1}}^6 = n \cdot \prod_{j=0}^{i} (1/\varepsilon_j)^{c_2} \cdot 3C(c_1+1)\gamma / (\varepsilon_{i+1})^{3(c_1+1)(a_1+a_2)+6} \log(1/\varepsilon_{i+1})$$

  which gives the claimed bound if $c_2 > 3(c_1+1)(a_1+a_2)+6$, as long as ε_{i+1} is sufficiently small.

- Randomness: The 'reduce' transformation of Lemma 5.8 does not change the randomness complexity, so we only need to find the randomness complexity of $V_i \circledast \mathcal{D}$. It is equal to log the proof length of V_i plus the randomness of $\mathcal{D}$,

  $$\log n + c_2 \sum_{j=0}^{i} \log 1/\varepsilon_j + a_2 \log N$$

  Now if $a_2 \log N = 3a_2(c_1+1) \log 1/\varepsilon_{i+1} \leq c_2 \log 1/\varepsilon_{i+1}$we are done.

□

6.3 Proof of Theorem 6.1

Let $\varepsilon_i = \varepsilon_0^{1/3^i}$ and c_0, c_1, c_2, c_3 be as in the statement of Lemma 6.6. We take the verifier to be V_i for i such that $1/\varepsilon_i = \mathrm{poly}(g(n))$. More precisely, such that $\varepsilon_i \leq \min\left\{\frac{1}{2g(n)}, c_0\right\} < \varepsilon_{i+1} = {\varepsilon_i}^{1/3}$. Clearly there is a unique such $i \leq O(\log\log n)$. By Lemma 6.6, $V = V_i$ has robust soundness error $2\varepsilon_i \leq 1/g(n)$ and query complexity $1/{\varepsilon_i}^{c_1} \leq (2g(n))^{3c_1}$. The randomness complexity is

$$\log n + c_2 \sum_{j=0}^{i} \log 1/\varepsilon_j = \log n + c_2 \sum_{j=0}^{i} 3^{-j} \log \varepsilon_0 \leq \log n + O((\log n)^{\beta})$$

Similarly, the proof length is easily seen to be $n^{1+o(1)}$. The equivalent statement about label cover follows from Lemma 2.5. □

We observe that the blowup in proof length and randomness complexity that is incurred by the composition steps is of the same order of the blowup incurred by the initial robust verifier V_0. This gives the following corollary.

Corollary 6.7 (Even shorter PCPs). *If* CircuitSat *has a robust verifier with randomness complexity* $\log n + \ell$*, robust soundness error* δ*, and query complexity* $\mathrm{poly}(1/\delta)$*, then, for every* $\delta' > \delta$*, it also has a robust verifier with query complexity* $\mathrm{poly}(1/\delta')$*, robust soundness error* δ' *and randomness complexity* $\log n + O(\ell)$*.*

Acknowledgements

We would like to thank Eli Ben-Sasson, Oded Goldreich, Venkatesan Guruswami, Dana Moshkovitz, Ran Raz, Omer Reingold, and Salil Vadhan for helpful discussions. Special thanks to Oded Goldreich for many helpful comments on an earlier version of this manuscript.

References

[ABN$^+$92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. *Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs.* IEEE Transactions on Information Theory, 38(2):509–516, 1992. doi:10.1109/18.119713.

[ABSS97] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. *The hardness of approximate optima in lattices, codes, and systems of linear equations.* J. Computer and System Sciences, 54(2):317–331, 1997. (Preliminary Version in *34th FOCS*, 1993). doi:10.1006/jcss.1997.1472.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. *Proof verification and the hardness of approximation problems.* J. ACM, 45(3):501–555, May 1998. (Preliminary Version in *33rd FOCS*, 1992). doi:10.1145/278298.278306.

[AS98] Sanjeev Arora and Shmuel Safra. *Probabilistic checking of proofs: A new characterization of NP.* J. ACM, 45(1):70–122, January 1998. (Preliminary Version in *33rd FOCS*, 1992). doi:10.1145/273865.273901.

[AS03] Sanjeev Arora and Madhu Sudan. *Improved low-degree testing and its applications.* Combinatorica, 23(3):365–426, 2003. (Preliminary Version in *29th STOC*, 1997). doi:10.1007/s00493-003-0025-0.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. *Robust PCPs of proximity, shorter PCPs and applications to coding.* SIAM J. Computing, 36(4):889–974, 2006. (Preliminary Version in *36th STOC*, 2004). doi:10.1137/S0097539705446810.

[BGLR93] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. *Efficient probabilistically checkable proofs and applications to approximation.* In *Proc.* 25*th ACM Symp. on Theory of Computing (STOC)*, pages 294–304. ACM, 1993. doi:10.1145/167088.167174.

[Bog05] Andrej Bogdanov. *Gap amplification fails below 1/2.* Technical Report TR05-046, Electronic Colloquium on Computational Complexity, 2005. (Comment on "Dinur, *The PCP theorem by gap amplification*"). Available from: `http://eccc.hpi-web.de/eccc-reports/2005/TR05-046/index.html`.

[BS08] Eli Ben-Sasson and Madhu Sudan. *Short PCPs with polylog query complexity.* SIAM J. Computing, 38(2):551–607, 2008. (Preliminary Version in *37th STOC*, 2005). doi:10.1137/050646445.

[DFK$^+$99] Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. *PCP characterizations of NP: Towards a polynomially-small error-probability.* In *Proc.* 31*st ACM Symp. on Theory of Computing (STOC)*, pages 29–40. ACM, 1999. doi:10.1145/301250.301265.

[Din07] Irit Dinur. *The PCP theorem by gap amplification.* J. ACM, 54(3):12, 2007. (Preliminary Version in *38th STOC*, 2006). doi:10.1145/1236457.1236459.

[Din08] ———. *PCPs with small soundness error.* SIGACT News, 39(3):41–57, 2008. doi:10.1145/1412700.1412713.

[DR06] Irit Dinur and Omer Reingold. *Assignment testers: Towards a combinatorial proof of the PCP Theorem.* SIAM J. Computing, 36:975–1024, 2006. (Preliminary Version in *45th FOCS*, 2004). doi:10.1137/S0097539705446962.

[FGL$^+$96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. *Interactive proofs and the hardness of approximating cliques*. J. ACM, 43(2):268–292, March 1996. (Preliminary version in *32nd FOCS*, 1991). doi:10.1145/226643.226652.

[FK95] Uriel Feige and Joe Kilian. *Impossibility results for recycling random bits in two-prover proof systems*. In *Proc.* 27*th ACM Symp. on Theory of Computing (STOC)*, pages 457–468. ACM, 1995. doi:10.1145/225058.225183.

[For66] G. David Forney. *Concatenated Codes*. MIT Press, Cambridge, MA, USA, 1966.

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. *On the power of multi-prover interactive protocols*. Theoretical Comp. Science, 134(2):545–557, 21 November 1994. (Preliminary Version in *3rd IEEE Symp. on Structural Complexity*, 1988). doi:10.1016/0304-3975(94)90251-8.

[Hås01] Johan Håstad. *Some optimal inapproximability results*. J. ACM, 48(4):798–859, July 2001. (Preliminary Version in *29th STOC*, 1997). doi:10.1145/502090.502098.

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson. *Expander graphs and their applications*. Bull. Amer. Math. Soc., 43:439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.

[IKW09] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. *Direct product testing: Improved and derandomized*. In *Proc.* 41*st ACM Symp. on Theory of Computing (STOC)*. ACM, 2009. (To appear).

[MR07] Dana Moshkovitz and Ran Raz. *Sub-constant error probabilistically checkable proof of almost linear size*. Technical Report TR07-026, Electronic Colloquium on Computational Complexity, 2007. Available from: `http://eccc.hpi-web.de/eccc-reports/2007/TR07-026/index.html`.

[MR08a] ———. *Sub-constant error low degree test of almost-linear size*. SIAM J. Computing, 38(1):140–180, 2008. (Preliminary Version in *38th STOC*, 2006). doi:10.1137/060656838.

[MR08b] ———. *Two query PCP with sub-constant error*. In *Proc.* 49*th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 314–323. IEEE, 2008. doi:10.1109/FOCS.2008.60.

[Raz98] Ran Raz. *A parallel repetition theorem*. SIAM J. Computing, 27(3):763–803, June 1998. (Preliminary Version in *27th STOC*, 1995). doi:10.1137/S0097539795280895.

[RS97] Ran Raz and Shmuel Safra. *A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP*. In *Proc.* 29*th ACM Symp. on Theory of Computing (STOC)*, pages 475–484. ACM, 1997. doi:10.1145/258533.258641.

[Sze99] Mario Szegedy. *Many-valued logics and holographic proofs*. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, eds., *Proc.* 26*th International Colloquium of Automata, Languages and Programming (ICALP)*, volume 1644 of *LNCS*, pages 676–686. Springer, 1999. doi:10.1007/3-540-48523-6.

A Extensions of dPCPs

In this section, we give various extensions of decodable PCPs. The first, is to define the notion of decodable PCPs for general pair languages, rather than just for CircuitSat. The second, is to allow the PCP decoder to output not just a single symbol of the witness, but rather any polynomial-time computable function of the witness. We provide sketches as to how constructions of dPCPs for CircuitSat can be adapted to give these extensions.

A.1 dPCPs for pair languages and functions

We defined, in Definition 3.1 and Definition 3.2, decodable PCPs for CircuitSat. According to that definition, a PCP decoder for CircuitSat receives a circuit C as explicit input and then locally decoded symbols of a satisfying assignment for C by locally accessing a proof π. However, we might as well have defined dPCPs for any NP language L. The (explicit) input to the PCP decoder in this case is an instance x of L, and the PCP decoder decodes symbols from the corresponding NP witness. More generally, we can define dPCPs for any pair language. A pair language is a language in which the input consists of pairs of strings of the form (x, y). For instance, the pair language corresponding to CircuitSat is the P-complete language, Circuit-Value defined as follows:

$$\textsc{CircuitVal} = \{(C, y) \mid \; C(y) = 1\}\,.$$

Given a pair language L and any x, we define the language $L(x) = \{y \mid \; (x, y) \in L\}$. For instance, for the pair language CircuitVal and any circuit C, CircuitVal(C) refers to the set of satisfying assignments of C.

Maintaining the analogy with NP language and the set of witnesses, we will call the first part, x, the actual input and the second part, y, the witness. Thus, the set $L(x)$ can be viewed as the set of witnesses to the fact that x is "in the language". In general, the two parts x and y need not be strings over the same alphabet. Since the PCP decoder will read the actual input x in full, the alphabet of this part is unimportant and we might as well assume that the alphabet is $\{0, 1\}$. On the other hand, since the PCP decoder will decode symbols of the witness y, the choice of alphabet of the witness is important. To be as general as possible, we will let this alphabet be a function of the length of the first input. More specifically, let $\{\Sigma_n\}_{n=1}^{\infty}$ be a family of alphabets and $N : \mathbb{Z}^+ \to \mathbb{Z}^+$ any function. We will consider pair languages $L \subseteq \bigcup_n \left(\{0,1\}^n \times \Sigma_n^{N(n)}\right)$. For obvious reasons, we will refer to $\{\Sigma_n\}$ as the witness alphabet and $N = N(n)$ as the length of the witness. For readability, we will use shorthand Σ and N for the witness alphabet and witness length, bearing in mind that both Σ and N may depend on n.

Decodable PCPs can be defined in the obvious fashion for any pair language L. A dPCP decoder for a pair language L, gets as input an actual input x of the pair language, it then locally queries a dPCP π and is expected to decode a symbol of a witness $y \in L(x)$. As before, the dPCP π is an encoding of the witness y that enables both local checking and local decoding.

We can further generalize this notion of dPCPs for pair languages to allow local decoding, not only of a single symbol of the witness y, but of an arbitrary function of y. More formally, we wish to decode one of the functions in the vector of functions $\mathbf{h} = (h_1, \ldots, h_k) : \Sigma^N \to \Sigma^k$ on the witness. The PCP decoder explicitly knows $\mathbf{h}$ and, on input x and j and oracle access to a dPCP π, is expected to output $h_j(y)$ where $y \in L(x)$ is the witness supposedly encoded by π.

We refer to these extensions of dPCPs as "functional dPCPs", defined formally below.

Definition A.1 (Functional dPCPs)**.** *Let $H = \left\{\mathbf{h}^{(n)}\right\}_n$ be a family of functions where $\mathbf{h}^{(n)} : \Sigma^N \to \Sigma^k$. An H-*PCP decoder *for a pair language L over witness alphabet Σ and proof alphabet σ is a probabilistic polynomial-time algorithm $\mathcal{D}$ that on input $x \in \{0,1\}^n$ and an index $j \in [k]$, tosses r random coins and computes a window $I = (i_1, \ldots, i_{\mathsf{q}})$ and a (local decoding) function $f : \sigma^{\mathsf{q}} \to \Sigma \cup \{\bot\}$ of decision complexity at most $\mathsf{s}(n)$.*

Completeness: *We say that $\mathcal{D}$ is complete if for every input x and $y \in L(x)$, there exists a proof*

$\pi \in \sigma^{\mathsf{m}}$, also called a decodable PCP, such that

$$\Pr_{j,I,f}[f(\pi_I) = h_j(y)] = 1$$

where $j \in [k]$ is chosen uniformly at random and I, f are distributed according to x, j and the verifier's random coins.

Robust Soundness: *We say that $\mathcal{D}$ has* robust *soundness error δ and list size L, if for every x and for any $\pi \in \sigma^m$, there is a list of $0 \le \ell \le \mathsf{L}$ strings $y^1, \ldots, y^\ell \in L(x)$ such that*

$$\mathop{\mathbb{E}}_{j,I,f}\left[\operatorname{agr}\left(\pi_I, \textsc{bad}(f)\right)\right] \le \delta,$$

where

$$\textsc{bad}(f) \triangleq \left\{ w \in \sigma^{\mathsf{q}} \;\middle|\; f(w) \notin \left\{\bot, h_j(y^1), \ldots, h_j(y^\ell)\right\}\right\}.$$

The special case in which the functions H being decoded are the symbols of the witness corresponds to the case where the vector of functions $\mathbf{h} : \Sigma^N \to \Sigma^k$ is the set of N projections $\mathbf{h}(y_1, \ldots, y_N) = (y_1, \ldots, y_N)$. In this case, we will drop the H and refer to the H-PCP decoder for L as just the PCP-decoder for the pair language L.

A.2 Constructions of functional dPCPs

We now show how existing constructions of dPCPs for CircuitSat yield functional dPCPs for any pair language in NP and any vector of polynomial time computable functions H.

In the terminology of pair languages, a decodable PCP for CircuitSat is actually a decodable PCP for the P-complete pair language CircuitVal. A closer look at the construction of dPCPs (see Section 6.1.2) reveals that the constructions actually gives a dPCP for the NP-complete pair language, non-deterministic Circuit-Value, defined as follows.

$$\textsc{Nondeterministic-CircuitVal} = \{(C, y) \mid \exists z, C(y, z) = 1\}.$$

We now derive the existence of functional dPCPs for any pair language in NP in two steps.

- The existence of a dPCP for Nondeterministic-CircuitVal implies the existence of dPCPs for any pair language $L \in NP$: just take the polynomial size non-deterministic circuit that checks the validity of the witness y for the fact that $x \in L$, and give it as input to the PCP decoder for Nondeterministic-CircuitVal.

- The existence of dPCPs for any pair language in NP in turn implies the existence of functional dPCPs for any pair language in NP and any polynomial time computable vector of functions H. Let L be a pair language and suppose $H = \left\{\mathbf{h}^{(n)}\right\}_n$ is a family of functions $\mathbf{h}^{(n)} : \Sigma^N \to \Sigma^k$ that are (polynomial time computable) functions of the witness y (Σ may also depend on n). Define a pair language by

$$L' = \{(x, z) \mid \exists y \in L(x), s.t.\; z = h_1(y) \circ h_2(y) \circ \ldots \circ h_k(y)\},$$

  where $\mathbf{h}^{(n)} = (h_1, \ldots, h_k)$, $n = |x|$, and $\circ$ denotes string concatenation. Clearly, if $L \in NP$ then $L' \in NP$. A dPCP for L' will give the desired outcome, since decoding the ith symbol of z amounts to decoding the function h_i of y.