The PCP Theorem by Gap Amplification

Irit Dinur[*]

February 13, 2007

Abstract

The PCP theorem [3, 2] says that every language in NP has a witness format that can be checked probabilistically by reading only a constant number of bits from the proof. The celebrated equivalence of this theorem and inapproximability of certain optimization problems, due to [12], has placed the PCP theorem at the heart of the area of inapproximability.

In this work we present a new proof of the PCP theorem that draws on this equivalence. We give a combinatorial proof for the NP-hardness of approximating a certain constraint satisfaction problem, which can then be reinterpreted to yield the PCP theorem.

Our approach is to consider the *unsat value* of a constraint system, which is the smallest fraction of unsatisfied constraints, ranging over all possible assignments for the underlying variables. We describe a new combinatorial amplification transformation that doubles the unsat-value of a constraint-system, with only a linear blowup in the size of the system. The amplification step causes an increase in alphabet-size that is corrected by a (standard) PCP composition step. Iterative application of these two steps yields a proof for the PCP theorem.

The amplification lemma relies on a new notion of "graph powering" that can be applied to systems of binary constraints. This powering amplifies the unsat-value of a constraint system provided that the underlying graph structure is an expander.

We also extend our amplification lemma towards construction of assignment testers (alternatively, PCPs of Proximity) which are slightly stronger objects than PCPs. We then construct PCPs and locally-testable codes whose length is linear up to a *polylog* factor, and whose correctness can be probabilistically verified by making a *constant* number of queries. Namely, we prove $SAT \in PCP_{\frac{1}{2},1}[\log_2(n \cdot \mathrm{poly}\log n),\ O(1)]$.

1 Introduction

A language L is in the class NP if there is a deterministic polynomial-time algorithm called a *verifier* that, in addition to the input, has access to a 'proof' such that the following holds: If $x \in L$ then there is a proof causing the verifier to accept, and if $x \notin L$ the verifier will reject regardless of the proof.

The PCP theorem is a strong characterization of the class NP. The notion of Probabilistically Checkable Proofs (PCPs) extends the power of the verifier by allowing it some randomness (and oracle access to the proof), and simultaneously restricts the verifier to read only a small number of symbols from the proof. More formally, the class $PCP[r, q]$ is defined to contain all languages L for which there is a verifier V that uses $O(r)$ random bits, reads $O(q)$ bits from the proof, and guarantees the following:

- If $x \in L$ then there is a proof π such that $\Pr[V^\pi(x) \text{ accepts}] = 1$. (Here and elsewhere $V^\pi(x)$ denotes the output of V on input x and proof π.)

[*]Hebrew University. Email: `dinuri@cs.huji.ac.il`. Supported by the Israel Science Foundation.

- If $x \notin L$ then for any proof π, $\Pr[V^{\pi}(x) \text{ accepts}] \leq \frac{1}{2}$.

The PCP theorem says that every language in NP has a verifier that uses at most $O(\log n)$ random bits and reads only $O(1)$ bits from the proof. In other words,

Theorem 1.1 (PCP Theorem, [3, 2]) $\mathrm{NP} \subseteq \mathrm{PCP}[\log n, 1]$.

This theorem was a great surprise, as it completely revises our concept of a proof. Rather than the classical notion of a proof as a sequential object that if erroneous in even one place can easily prove a false statement. The PCP theorem provides a new proof notion that is more robust, and must be erroneous in many places when attempting to prove a falsity.

Historically, the class $PCP[r, q]$ stemmed out of the celebrated notion of interactive proofs [20, 4] and the class IP. The original motivation for defining IP was cryptographic, but it soon lead to a list of remarkable complexity-theoretic results, including for example IP=PSPACE (see [24, 32]). We will not give a detailed historic account which can be found in, say, [1]. Let us just mention that an exciting sequence of papers (see [6, 14, 5]) lead to the following theorem: the class of all languages with exponential-sized proofs is equal to the class of all languages that can be verified by a (randomized) polynomial-time verifier. At this point attempts were made to "scale down" this result so as to characterize the class NP in similar terms, through suitable restriction of the verifier. This was especially motivated by the discovery of [12] that connected such a scale-down to an inapproximability result for the clique number (see below). This scale-down was achieved partially in [3] and completed in [2] and came to be known as the PCP theorem.

The techniques that lead to the proof were mainly algebraic, including low-degree extension over finite fields, low-degree test, parallelization through curves, a sum-check protocol, and the Hadamard and quadratic functions encodings.

1.1 PCP and Inapproximability

As mentioned above, the discovery of the PCP theorem came hand in hand with the beautiful and surprising connection, discovered by Feige et. al. [12], between PCP characterizations of NP and the hardness of approximating the clique number in a graph. Predating these developments the situation regarding approximation problems was unclear. There was no clue why different approximation problems seem to exhibit different approximation behavior. The PCP theorem implied, for the first time, that numerous problems (including, for example, max-3-SAT) are hard to approximate. This has had a tremendous impact on the study of combinatorial optimization problems, and today the PCP theorem stands at the heart of nearly all hardness-of-approximation results.

The connection to inapproximability is best described through *constraint satisfaction* problems. Let us begin by defining a *constraint*,

Definition 1.1 *Let $V = \{v_1, \ldots, v_n\}$ be a set of variables, and let Σ be a finite alphabet. A q-ary constraint $(C, i_1, \ldots, i_q)$ consists of a q-tuple of indices $i_1, \ldots, i_q \in [n]$ and a subset $C \subseteq \Sigma^q$ of "acceptable" values. A constraint is* satisfied *by a given assignment $a : V \to \Sigma$ iff $(a(v_{i_1}), a(v_{i_2}), \ldots, a(v_{i_q})) \in C$.*

The constraint satisfaction problem (CSP) is the problem of, given a system of constraints $\mathcal{C} = \{c_1, \ldots, c_n\}$ over a set of variables V, deciding whether there is an assignment for the variables that satisfies every constraint. This problem is clearly NP-complete as it generalizes many well known NP-complete problems such as 3-SAT and 3-colorability. For example, in the equivalent of the 3-colorability problem, the alphabet is $\Sigma = \{1, 2, 3\}$ and the binary constraints are of the form (C, i_1, i_2) where

$C = \{(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)\}$ consists of 6 out of the possible 9 values that exclude equality.

An optimization version of this problem, called max-CSP, is to find an assignment that satisfies a *maximum* number of constraints. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a set of constraints over a set of variables V, in this paper we consider the *unsat-value* of $\mathcal{C}$, denoted $\text{UNSAT}(\mathcal{C})$, defined to be the smallest fraction of unsatisfied constraints, over all possible assignments for V. Clearly $\mathcal{C}$ is satisfiable if and only if $\text{UNSAT}(\mathcal{C}) = 0$. In this notation, the following theorem is a typical inapproximability result.

Theorem 1.2 (Inapproximability version of PCP Theorem) *There are integers $q > 1$ and $|\Sigma| > 1$ such that, given as input a collection $\mathcal{C}$ of q-ary constraints over an alphabet Σ, it is NP-hard to decide whether $\text{UNSAT}(\mathcal{C}) = 0$ or $\text{UNSAT}(\mathcal{C}) \geq \frac{1}{2}$.*

Such a theorem is proven by showing a polynomial-time reduction from an NP-complete language L, reducing an instance x to a constraint system $\mathcal{C}_x$, such that the following gap property holds: if $x \in L$, then $\text{UNSAT}(\mathcal{C}_x) = 0$ and if $x \notin L$ then $\text{UNSAT}(\mathcal{C}_x) \geq \frac{1}{2}$.

The point is that the above Theorem 1.2 is *equivalent* to the PCP theorem (Theorem 1.1).

Lemma 1.3 *Theorem 1.1 and Theorem 1.2 are equivalent.*

Proof: Let us briefly explain this equivalence.

- $(\Rightarrow)$ According to Theorem 1.1 every NP language (and let us fix some language $L \in NP$) has a verification procedure Ver that reads $c \log n$ random bits, accesses $q = O(1)$ bits from the proof and decides whether to accept or reject (where q and c are constants. For each fixed random bit pattern $r \in \{0,1\}^{c \log n}$, Ver (deterministically) reads a fixed set of q bits from the proof: $i_1^{(r)}, \ldots, i_q^{(r)}$. Denote by $C^{(r)} \subseteq \{0,1\}^q$ the possible contents of the accessed proof bits that would cause Ver to accept.

  We present a reduction from L to gap constraint satisfaction. Let $x \overset{?}{\in} L$ be the input, and denote $n = |x|$. Let $\Sigma = \{0,1\}$ and put a Boolean variable for each proof location accessed by Ver on input x (so there are at most $q2^{c \log n} = qn^c$ Boolean variables). Next, construct a system of constraints $\mathcal{C}_x = \{c_r\}_{r \in \{0,1\}^{c \log n}}$ such that the constraint c_r is defined by $c_r = (C^{(r)}, i_1^{(r)}, \ldots, i_q^{(r)})$. It remains to observe that the rejection probability of Ver is exactly equal $\text{UNSAT}(\mathcal{C}_x)$ so it is zero if $x \in L$, and at least $\frac{1}{2}$ if $x \notin L$.

- $(\Leftarrow)$ For the converse, assume there is a reduction taking instances of any NP-language into constraint systems such that the gap property holds. Here is how to construct a verifier. The verifier will first (deterministically) compute the constraint system output by the reduction guaranteed above. It will expect the proof to consist of an assignment for the variables of the constraint system. Next, the verifier will use its random string to select a constraint uniformly at random, and check that the assignment satisfies it, by querying the proof at the appropriate locations.

$\blacksquare$

The $(\Rightarrow)$ direction of the equivalence has so far been the more useful one, as it enables us to derive inapproximability results from the PCP theorem. In this work, we rely on the converse $(\Leftarrow)$ direction, to give a new "inapproximability-based" proof for the PCP theorem. We find it especially pleasing that our proof has a similar flavor to the area in which the theorem has been most applicable.

1.2 Constraint Graphs and Operations on them

Our approach for proving the PCP Theorem (as stated in Theorem 1.2) is based on an iterative gap amplification step. For this, we restrict attention to systems of binary (two-variable) constraints, such as 3-colorability constraints. Any binary constraint system naturally defines an underlying graph, in which the variables are vertices, and two variables are adjacent iff there is a constraint over them. We call this a *constraint graph*.

Definition 1.2 (Constraint Graph) *$G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ is called a constraint graph, if*

1. *(V,E) is an undirected graph, called the underlying graph of G.*
2. *The set V is also viewed as a set of variables assuming values over alphabet Σ.*
3. *Each edge $e \in E$, carries a constraint $c(e) \subseteq \Sigma \times \Sigma$, and $\mathcal{C} = \{c(e)\}_{e \in E}$. A constraint $c(e)$ is said to be satisfied by (a,b) iff $(a,b) \in c(e)$.*

We sometimes use the same letter G to denote the constraint graph and the underlying graph (V,E).

An assignment is a mapping $\sigma : V \to \Sigma$ that gives each vertex in V a value from Σ. For any assignment σ, define

$$\text{UNSAT}_\sigma(G) = \Pr_{(u,v)\in E}\left[(\sigma(u),\sigma(v)) \notin c(e)\right] \quad \text{and} \quad \text{UNSAT}(G) = \min_\sigma \text{UNSAT}_\sigma(G)\,.$$

We call $\text{UNSAT}(G)$ the **unsat-value** of G, or just the unsat of G for short. We define

$$size(G) \overset{\triangle}{=} |V| + |E|\,.$$

Implicit throughout this paper is the notion that we are working with infinite families of constraint graphs. In our context the size of the alphabet Σ will remain fixed independently of the size of the underlying graph structure, so indeed $size(G)$ measures the size of the description of G up to a constant multiplicative factor that depends only on $|\Sigma|$.

Proposition 1.4 *Given a constraint graph $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ with $|\Sigma| = 3$, it is NP-hard to decide if $\text{UNSAT}(G) = 0$.*

Proof: We reduce from graph 3-colorability. Given a graph G, let the alphabet be $\Sigma = \{1,2,3\}$ for the three colors, and equip the edges with inequality constraints. Clearly, G is 3-colorable if and only if $\text{UNSAT}(G) = 0$. ■

Observe that in case $\text{UNSAT}(G) > 0$ it must be that $\text{UNSAT}(G) \ge 1/|G|$. Therefore, it is actually NP-hard to distinguish between the cases (i) $\text{UNSAT}(G) = 0$ and (ii) $\text{UNSAT}(G) \ge 1/|G|$. Our main theorem is the aforementioned 'gap amplification step', where a graph G is converted into a new graph G' whose unsat value is doubled.

Theorem 1.5 (Main) *There exists Σ_0 such that the following holds. For any finite alphabet Σ there exist $C > 0$ and $0 < \alpha < 1$ such that, given a constraint graph $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$, one can construct, in polynomial time, a constraint graph $G' = \langle (V',E'), \Sigma_0, \mathcal{C}' \rangle$ such that*

- *$size(G') \le C \cdot size(G)$.*
- *(Completeness:) If $\text{UNSAT}(G) = 0$ then $\text{UNSAT}(G') = 0$.*

- *(Soundness:)* $\text{UNSAT}(G') \geq \min(2 \cdot \text{UNSAT}(G), \alpha)$.

After applying this step logarithmically many times, the final outcome G_{final} is a constraint graph for which in case (i) still $\text{UNSAT}(G_{final}) = 0$, and in case (ii) we have $\text{UNSAT}(G_{final}) \geq \alpha$ for some constant $\alpha > 0$. Ignoring the fact that instead of $\frac{1}{2}$ we got $\alpha > 0$ (and this can be easily corrected by repetition), this proves the PCP Theorem (as stated in Theorem 1.2). The formal proof of this is given in Section 3.

Let us describe the ideas which come in to the proof of the main theorem. How do we make the unsat value double? this is done through three operations on constraint graphs, which we describe next.

Graph Powering

In order to amplify the unsat value of a constraint graph we simply raise it to the power t, for some constant value of t. This operation is a new operation on constraint systems defined as follows. Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a d-regular constraint graph, and let $t \in \mathbb{N}$. A sequence $(u_0, \ldots, u_t)$ is called a t-step walk in G if for all $0 \leq i < t$, $(u_i, u_{i+1}) \in E$. We define $G^t = \left\langle (V, \mathbf{E}), \Sigma^{d^{\lceil t/2 \rceil}}, \mathcal{C}^t \right\rangle$ to be the following constraint graph:

- The vertices of G^t are the same as the vertices of G.
- Edges: u and v are connected by k parallel edges in $\mathbf{E}$ if the number of t-step walks from u to v in G is exactly k.
- Alphabet: The alphabet of G^t is $\Sigma^{d^{\lceil t/2 \rceil}}$. The meaning of each value is as follows. Let $\Gamma(u) = \left\{ u' \in V \mid (u = u_0, u_1, \ldots, u_{\lceil t/2 \rceil} = u') \text{ is a walk in } G \right\}$. Clearly $|\Gamma(u)| \leq d^{\lceil t/2 \rceil}$ and by choosing some canonical ordering, a value $a \in \Sigma^{d^{\lceil t/2 \rceil}}$ can be interpreted as an assignment $a : \Gamma(u) \to \Sigma$. One may think of this value as describing u's opinion of its neighbors' values.
- Constraints: The constraint associated with an edge $\mathbf{e} = (u, v) \in \mathbf{E}$ is satisfied by a pair of values $a, b \in \Sigma^{d^{\lceil t/2 \rceil}}$ iff the following holds. There is an assignment $\sigma : \Gamma(u) \cup \Gamma(v) \to \Sigma$ that satisfies every constraint $c(e)$ where $e \in E \cap \Gamma(u) \times \Gamma(v)$, and such that

$$\forall u' \in \Gamma(u), v' \in \Gamma(v), \qquad \sigma(u') = a_{u'} \text{ and } \sigma(v') = b_{v'}$$

  where $a_{u'}$ is the value a assigns $u' \in \Gamma(u)$, and $b_{v'}$ the value b assigns $v' \in \Gamma(v)$.

If $\text{UNSAT}(G) = 0$ then clearly $\text{UNSAT}(G^t) = 0$. More interestingly, our main technical lemma asserts that the unsat value is multiplied by a factor of roughly $\sqrt{t}$. This holds as long as the initial underlying graph G is sufficiently well-structured, i.e., the graph is expanding (captured by bounding $\lambda(G)$, defined in Section 2.1) and d-regular for constant d, and has self-loops.

Lemma 1.6 (Amplification Lemma) *Let $0 < \lambda < d$, and $|\Sigma|$ be constants. There exists a constant $\beta_2 = \beta_2(\lambda, d, |\Sigma|) > 0$, such that for every $t \in \mathbb{N}$ and for every d-regular constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ with a self-loop on each vertex and $\lambda(G) \leq \lambda$,*

$$\text{UNSAT}(G^t) \geq \beta_2 \sqrt{t} \cdot \min\left(\text{UNSAT}(G) \, , \, \frac{1}{t} \right) .$$

The advantage of the powering operation is that it amplifies the UNSAT value by factor $\sqrt{t}$ and only incurs a *linear* blowup in the size of the graph (the number of edges is multiplied by d^{t-1}).

Preprocessing

It is quite easy to turn any constraint graph into a 'well-structured' one, as required by the amplification step. This can be done with only a linear blow-up in size, and a constant factor decrease in the unsat value. For example, here is a simple way of turning any constant-degree constraint graph into an expanding one. Simply take the union of the edges of the given graph with edges of any constant-degree expander graph on the same set of vertices. Putting null constraints on the expander edges guarantees that the unsat value only drops by a constant factor.

The following lemma summarizes the preprocessing step:

Lemma 1.7 (Preprocessing Lemma) *There exist constants $0 < \lambda < d$ and $\beta_1 > 0$ such that any constraint graph G can be transformed into a constraint graph G', denoted $G' = \mathrm{prep}(G)$, such that*

- *G' is d-regular with self-loops, and $\lambda(G') \leq \lambda < d$.*
- *G' has the same alphabet as G, and $size(G') = O(size(G))$.*
- *$\beta_1 \cdot \textsc{unsat}(G) \leq \textsc{unsat}(G') \leq \textsc{unsat}(G)$.*

Alphabet Reduction by Composition

The graph powering operation described above has one drawback: it incurs an increase in the alphabet size. In order to repeat the amplification step many times, the alphabet size must be reduced.

Fortunately, this can be achieved through composition. Composition is an essential component in all PCP constructions, starting with [3]. It is most natural in the proof-verification setting (rather than as a gap constraint satisfaction reduction). Recall that a system of q-ary constraints over an alphabet Σ corresponds to a probabilistic verifier that reads q symbols from a proof, where the symbols are taken from Σ.

The basic idea of proof composition is that the verifier, instead of reading the q symbols from Σ (of which we think as a 'large' alphabet) and based on them verifying correctness, can delegate this task to another "inner" verifier. This inner verifier can rely on an additional proof for the fact that *this length-q input would have caused the original verifier to accept*. Thus the verification task can potentially rely on reading even fewer bits than before. Note that there will end up being many additional proofs, one per random string of the original verifier. Consistency between these proofs must be ensured, and this well-studied issue will be discussed in Section 5.

Going back to the language of constraint systems, the "inner verifier" is simply a reduction transforming a single constraint over large-alphabet variables into a system of constraints over new small-alphabet variables. This reduction is applied on every constraint in parallel and is done in a consistent way[1], ensuring that the UNSAT value of the new system doesn't drop by more than a constant factor. We call such a reduction an "assignment tester" and refer the reader to Section 5 and Definition 5.1 for a formal definition of the composition operation.

Lemma 1.8 (Composition Lemma - Informal statement) *Assume the existence of an assignment tester $\mathcal{P}$, with constant rejection probability $\varepsilon > 0$, and alphabet Σ_0, $|\Sigma_0| = O(1)$. There exists $\beta_3 > 0$ that depends only on $\mathcal{P}$, such that given any constraint graph $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$, one can compute, in linear time, the constraint graph $G' = G \circ \mathcal{P}$, such that $size(G') = c(\mathcal{P}, |\Sigma|) \cdot size(G)$, and*

$$\beta_3 \cdot \textsc{unsat}(G) \leq \textsc{unsat}(G') \leq \textsc{unsat}(G).$$

For the sake of self-containedness, we include a construction of an assignment tester $\mathcal{P}$ in Section 7.

[1]This has to do with the consistency issue mentioned earlier, and will be clarified in Section 5.

1.3 The Combined Amplification Step

Assuming we have Lemma 1.7, Lemma 1.6, and Lemma 1.8, the proof of the gap amplification step (Theorem 1.5) is syntactic and is given in Section 3. Altogether, our proof of the PCP theorem takes the following form: Let G be an instance of constraint graph satisfiability (proven NP-hard in Proposition 1.4). Fix $t = O(1)$, set $G_0 = G$, and repeat the following amplification step $\log |G|$ times:

1. Preprocess G_i
2. Raise the result to the t-th power
3. Compose the result with an assignment tester reduction $\mathcal{P}$.

In short,

$$G_{i+1} = (\text{prep}(G_i))^t \circ \mathcal{P}$$

It is not hard to see that taking $G_{final} = G_i$ for $i = \Theta(\log n)$ gives the necessary reduction. Formal details are given in Section 3.

1.4 Related Work

This work follows [17, 11] in the attempt to find an alternative proof for the PCP Theorem that is combinatorial and/or simpler. In [11], a quasi-polynomial PCP Theorem was proven combinatorially. While our proof is different, we do rely on the modular notion of composition due to [7, 11], and in particular on composition with a bounded-input assignment-tester, which has already served as an ingredient in the constructions of [11].

This construction is inspired by the zig-zag construction of expander graphs due to [31] and by Reingold's beautiful proof for $SL = L$ [30]. Although there is no direct technical connection between these works and our construction, our proof has the same overall structure, consisting of a logarithmic number of iterations, where each iteration makes a small improvement in the interesting parameter (be it the UNSAT value in our case, or the spectral gap in Reingold's case).

The steady increase of the UNSAT value is inherently different from the original proof of the PCP Theorem. There, a constant UNSAT value (using our terminology) is generated by one powerful transformation, and then a host of additional transformations are incorporated into the final result to take care of other parameters. Composition is essential in both proofs.

1.5 Further Results

Short PCPs and Locally Testable Codes

The goal of achieving extremely-short Probabilistically Checkable Proofs and Locally-Testable Codes (LTCs) has been the focus of several works [27, 21, 18, 9, 7, 8]. The goal is to convert a standard NP proof into a "robust" PCP proof, with the minimal amount of increase in the proof length. Discussion of Locally Testable Codes is deferred to Section 8.

The shortest PCPs/LTCs are due to [7] and [8], each best in a different parameter setting. For the case where the number of queries is constant, the shortest construction is due to [7], and the proof-length is $n \cdot 2^{(\log n)^{\varepsilon}}$. The construction of [8] has shorter proof-length, $n \cdot \text{poly} \log n$, but the number of queries it requires is $\text{poly} \log n$. Our result combines the best parameters from both of these works. Our starting point is the construction [8]. We first transform this construction into a two-query constraint system

$\mathcal{C}$ whose size is $n \cdot \mathrm{poly}\log n$, such that if the input was a 'no' instance, then $\mathrm{UNSAT}(\mathcal{C}) \geq \frac{1}{\mathrm{poly}\log n}$, and otherwise $\mathrm{UNSAT}(\mathcal{C}) = 0$. Then, by applying our amplification step $O(\log\log n)$ times, we raise the unsat value to a constant, while increasing the size of the system by only another polylogarithmic factor. Using standard notation (which is defined in Section 8), we show that $SAT \in PCP_{\frac{1}{2},1}[\log_2(n \cdot \mathrm{poly}\log n),\ O(1)]$.

Assignment Testers

We further extend our main amplification step (Theorem 1.5) to work for assignment-tester reductions (alternatively called PCPs of Proximity), defined in [7, 11]. This carries over to extend our combinatorial construction of PCPs to that of assignment-testers / PCPs of Proximity. Without getting into the full definition (which can be found in Section 9) we note that this object is syntactically stronger than a PCP reduction. It is known to imply the PCP theorem, but the converse is not known.

We obtain the aforementioned short locally-testable codes by first obtaining short assignment-testers (with comparable parameters to those of the short PCPs described above), and then applying a simple generic construction from [7].

1.6 Organization

Section 2 contains some preliminary facts about expander graphs and probability. In Section 3 we prove the main theorem, relying on Lemmas 1.6, 1.7, and 1.8 stated above, and deduce the PCP Theorem as well. The next three sections (sections 4, 5, and 6) are devoted to the proof of Lemmas 1.7, 1.8 and 1.6, dealing with the three operations on constraint graphs. In Section 7 we describe a concrete (and inefficient) construction of an assignment-tester $\mathcal{P}$ so as to make our proof self-contained.

Sections 8 and 9 contain the results on short PCPs and LTCs. In Section 8 we construct PCPs and locally-testable codes whose length is linear up to a poly-logarithmic factor. In Section 9 we describe how to extend our main amplification step (Theorem 1.5) for assignment-testers. We include a short discussion about our amplification and parallel-repetition in Section 10.

2 Preliminaries

2.1 Expander Graphs

Expander graphs play an important role in a wide variety of results in theoretical computer science. In this section we will state some well-known properties of expander graphs. For an excellent exposition to this subject, we refer the reader to [23].

Definition 2.1 *Let $G = (V, E)$ be a d-regular graph. Let $E(S, \bar{S}) = \left|(S \times \bar{S}) \cap E\right|$ equal the number of edges from a subset $S \subseteq V$ to its complement. The* **edge expansion** *of G is defined as*

$$h(G) = \min_{S:\,|S| \leq |V|/2} \frac{E(S, \bar{S})}{|S|}.$$

Lemma 2.1 (Expanders) *There exist $d_0 \in \mathbb{N}$ and $h_0 > 0$, such that there is a polynomial-time constructible family $\{X_n\}_{n \in \mathbb{N}}$ of d_0-regular graphs X_n on n vertices with $h(X_n) \geq h_0$. (Such graphs are called expanders).*

Proof: It is well known that a random constant-degree graph on n-vertices is an expander. For a deterministic construction, one can get expanders on 2^k vertices for any k from the construction of [31]. For

$n = 2^k - n'$ ($n' < 2^{k-1}$) one can, for example, merge n' pairs of non-neighboring vertices. To make this graph regular one can add arbitrary edges to the non-merged vertices. Clearly, the edge expansion is maintained up to a constant factor. ■

The adjacency matrix of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix A such that $A_{ij} = 1$ iff $(i, j) \in E$ and $A_{ij} = 0$ otherwise. The second eigenvalue of a graph G, denoted $\lambda(G)$, is the second largest eigenvalue of its adjacency matrix in absolute value. The Rayleigh quotient gives a convenient expression for this value.

Lemma 2.2 *Let G be a graph, A its adjacency matrix, and let $\lambda(G)$ denote the second largest eigenvalue in absolute value. Then $\lambda(G) = \max_{x \neq \vec{0}, x \perp \vec{1}} \frac{|\langle x, Ax\rangle|}{\langle x, x\rangle}$.* ■

The following important relation between the edge expansion and the second eigenvalue is well-known, see, e.g., [23],

Theorem 2.3 *Let G be a d-regular graph with eigenvalues $d = \lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{n-1}$, and let $h(G)$ denote the edge expansion of G. Then*

$$\lambda_1 \leq d - \frac{h(G)^2}{2d} .$$

■

As a corollary of Lemma 2.1 and the above theorem we obtain

Corollary 2.4 *There exist ${d_0}' \in \mathbb{N}$ and $0 < \lambda_0 < {d_0}'$, such that there is a polynomial-time constructible family $\{X_n\}_{n \in \mathbb{N}}$ of ${d_0}'$-regular graphs X_n on n vertices with $\lambda(X_n) \leq \lambda_0$.*

Proof: For any $n \in \mathbb{N}$ let X_n be the d_0-regular graph guaranteed by Lemma 2.1. By adding d_0 self-loops to each vertex in X_n we obtain a ${d_0}' = 2d_0$-regular graph X'_n, with the same edge-expansion h_0. However, it is easy to see that now all eigenvalues of X'_n are non-negative, and in particular $\lambda(X'_n)$ equals the second-largest eigenvalue of X'_n. Taking $\lambda_0 = {d_0}' - \frac{(h_0)^2}{2{d_0}'} < {d_0}'$, Theorem 2.3 gives the result. ■

Finally, we prove the following (standard) estimate on the random-like behavior of a random-walk on an expander.

Proposition 2.5 *Let $G = (V, E)$ be a d-regular graph with $\lambda(G) = \lambda$. Let $F \subseteq E$ be a set of edges without self loops, and let K be the distribution on vertices induced by selecting a random edge in F, and then a random endpoint.*

The probability p that a random walk that starts with distribution K takes the $i + 1$st step in F, is upper bounded by $\frac{|F|}{|E|} + \left(\frac{|\lambda|}{d}\right)^i$.

Proof: Let $B \subseteq V$ be the support of K. Let $n = |V|$ and let A be the normalized $n \times n$ adjacency matrix of G, i.e., A_{ij} equals k/d where k is the number of edges between vertices i and j. The first and second largest eigenvalues (in absolute value) of A are 1 and $\tilde{\lambda} = \lambda/d$ respectively.

Let x be the vector corresponding to the distribution K, i.e. $x_v = \Pr_K[v]$ equals the fraction of edges touching v that are in F, divided by 2. Since the graph is d-regular, $\Pr_K[v] \leq \frac{d}{2|F|}$. Let y_v be the probability that a random step from v is in F, so $y = \frac{2|F|}{d} x$. The probability p equals the probability of landing in B after i steps, and then taking a step inside F,

$$p = \sum_{v \in B} y_v (A^i x)_v = \sum_{v \in V} y_v (A^i x)_v = \langle y, A^i x \rangle .$$

Let $\mathbf{1}$ be the all 1 vector. Write $x = x^{\perp} + x^{||}$ where $x^{||} \overset{\triangle}{=} \frac{1}{n}\mathbf{1}$, is an eigenvector of A with eigenvalue 1, and $x^{\perp} \overset{\triangle}{=} x - x^{||}$. The vector $x^{\perp}$ is orthogonal to $x^{||}$ since $\mathbf{1} \cdot x^{\perp} = \sum_v \Pr_K[v] - \sum_v \frac{1}{n} = 1 - 1 = 0$. Denote $\|x\| = \sqrt{\sum_v x_v^2}$. Clearly,

$$\|A^i x^{\perp}\| \leq |\tilde{\lambda}|^i \|x^{\perp}\| \leq |\tilde{\lambda}|^i \|x\| \, .$$

Observe that $\|x\|^2 \leq (\sum_v |x_v|) \cdot (\max_v |x_v|) \leq 1 \cdot (\max_v |x_v|) \leq \frac{d}{2|F|}$. By the Cauchy-Schwarz inequality,

$$\left\langle y, A^i x^{\perp} \right\rangle \leq \|y\| \cdot \|A^i x^{\perp}\| \leq \frac{2\,|F|}{d} \|x\| \cdot |\tilde{\lambda}|^i \|x\| \leq |\tilde{\lambda}|^i \, .$$

Combining the above we get the claim,

$$\left\langle y, A^i x \right\rangle = \left\langle y, A^i x^{||} \right\rangle + \left\langle y, A^i x^{\perp} \right\rangle \leq \frac{2\,|F|}{dn} + |\tilde{\lambda}|^i = \frac{|F|}{|E|} + \left(\frac{|\lambda|}{d} \right)^i$$

■

2.2 Probability

The following easy fact is a Chebychev-style inequality. It is useful for showing that for a non-negative random variable X, $Pr[X > 0] \approx \mathbb{E}[X]$ whenever $\mathbb{E}[X] \approx \mathbb{E}[X^2]$.

Fact 2.6 *For any non-negative random variable $X \not\equiv 0$, $\Pr[X > 0] \geq \frac{\mathbb{E}^2[X]}{\mathbb{E}[X^2]}$.*

Proof: We repeat a proof from [25, Lecture 5].

$$\mathbb{E}\left[X\right] = \mathbb{E}\left[X \cdot 1_{X>0}\right] \leq \sqrt{\mathbb{E}\left[X^2\right]} \sqrt{\mathbb{E}\left[(1_{X>0})^2\right]} = \sqrt{\mathbb{E}\left[X^2\right]} \sqrt{\Pr[X > 0]}.$$

where we have used the Cauchy-Schwarz inequality. Squaring and rearranging completes the proof. ■

2.3 Error Correcting Codes

An *error-correcting code* is a collection of strings $C \subseteq \Sigma^n$, where Σ is some finite alphabet. n is called the block-length of the code, $\log_{|\Sigma|}|C|$ is the dimension of the code, and $\frac{1}{n}\log_{|\Sigma|}|C|$ is the rate of the code. The distance of the code is $\min_{x \neq y \in C} \mathrm{dist}(x, y)$ where $\mathrm{dist}(\cdot, \cdot)$ refers to Hamming distance. We also write $\mathrm{rdist}(x, y) = \frac{1}{n}\mathrm{dist}(x, y)$ for relative distance.

A one-to-one mapping $\mathrm{e} : D \to \Sigma^n$ is also sometimes called an error-correcting code. Its dimension and distance are defined to be the respective dimension and distance of its image $\mathrm{e}(D)$.

It is well known that there exist families of codes $\{C_n \subset \{0,1\}^n\}_{n \in \mathbb{N}}$ for which both the distance and the dimension are $\Theta(n)$, and for which there is a polynomial-sized circuit that checks $x \overset{?}{\in} C_n$, see e.g. [33].

2.4 Assignment Tester

An assignment tester is a certain type of PCP transformation that is useful for composition. We describe below a stripped-down version of the definition of [11], that suffices for our purposes.

Basically, an assignment tester is an algorithm whose input is a Boolean circuit Φ and whose output is a constraint graph G. This graph *contains* the input variables of Φ as some of its vertices, and its unsat value is related to the satisfiability of Φ as follows. Roughly speaking, the only way an assignment for the variables of G can have a small unsat value is if its restriction to the variables of Φ is close to an assignment that satisfies Φ. Here is the formal definition.

For a Boolean circuit Φ over n variables, denote by $\mathrm{SAT}(\Phi) \subseteq \{0,1\}^n$ the set of assignments that satisfy Φ.

Definition 2.2 (Assignment Tester) *An* Assignment Tester *with alphabet Σ_0 and rejection probability $\epsilon > 0$ is an algorithm $\mathcal{P}$ whose input is a circuit Φ over Boolean variables X, and whose output is a constraint graph $G = \langle (V,E), \Sigma_0, \mathcal{C} \rangle$ such that*[2] *$V \supset X$, and such that the following hold. Let $V' = V \setminus X$, and let $a : X \to \{0,1\}$ be an assignment.*

- *(Completeness) If $a \in \mathrm{SAT}(\Phi)$, there exists $b : V' \to \Sigma_0$ such that $\text{UNSAT}_{a \cup b}(G) = 0$.*
- *(Soundness) If $a \notin \mathrm{SAT}(\Phi)$ then for all $b : V' \to \Sigma_0$, $\text{UNSAT}_{a\cup b}(G) \geq \epsilon \cdot \mathrm{rdist}(a, \mathrm{SAT}(\Phi))$.*

Note that we make no requirement on the complexity of the algorithm $\mathcal{P}$.

3 Proofs of the Main Theorem and of the PCP Theorem

Based on the constraint graph operations described in Section 1.2, and on Lemma 1.7, Lemma 1.6, and Lemma 1.8 we can already prove our main theorem.

Proof of Theorem 1.5: We define

$$G' = (\,\mathrm{prep}(G)\,)^t \circ \mathcal{P}$$

for an assignment tester $\mathcal{P}$ whose existence is guaranteed by Theorem 5.1, and a value $t \in \mathbb{N}$ to be determined later. Let us elaborate on the construction of G':

1. (Preprocessing step:) Let $H_1 = \mathrm{prep}(G)$ be the result of applying to G the transformation guaranteed by Lemma 1.7. There exist some global constants $\lambda < d$ and $\beta_1 > 0$ such that H_1 is d-regular, has the same alphabet as G, $\lambda(H_1) \leq \lambda < d$, and $\beta_1 \cdot \text{UNSAT}(G) \leq \text{UNSAT}(H_1) \leq \text{UNSAT}(G)$.
2. (Amplification step:) Let $H_2 = (H_1)^t$, for a large enough constant $t > 1$ to be specified below.

   According to Lemma 1.6, there exists some constant $\beta_2 = \beta(\lambda, d, |\Sigma|) > 0$ for which $\text{UNSAT}(H_2) \geq \beta_2\sqrt{t} \cdot \min(\text{UNSAT}(H_1), \frac{1}{t})$. However, the alphabet grows to $\Sigma^{d^{\lceil t/2 \rceil}}$.
3. (Composition step:) Let $G' = H_2 \circ \mathcal{P}$ be the result of applying to H_2 the transformation guaranteed by Lemma 1.8. Here we rely on the existence of an assignment tester $\mathcal{P}$, as guaranteed in Theorem 5.1.

   The alphabet of G' is reduced to Σ_0 while still $\beta_3 \cdot \text{UNSAT}(H_2) \leq \text{UNSAT}(G') \leq \text{UNSAT}(H_2)$, for a constant $\beta_3 > 0$.

We now verify the properties claimed above. Completeness is clearly maintained at each step, i.e.,

$$\text{UNSAT}(G) = 0 \;\Rightarrow\; \text{UNSAT}(H_1) = 0 \;\Rightarrow\; \text{UNSAT}(H_2) = 0 \;\Rightarrow\; \text{UNSAT}(G') = 0.$$

For soundness, let us choose now

$$t = \lceil (\frac{2}{\beta_1\beta_2\beta_3})^2 \rceil, \quad \text{and} \quad \alpha = \beta_3\beta_2/\sqrt{t}.$$

[2]In a constraint graph, the set V plays a double role of both variables and vertices. By $V \supset X$ it is meant that some of the vertices of V are identified with the X variables.

Altogether,

$$\begin{aligned}\text{UNSAT}(G') &\geq \beta_3 \cdot \text{UNSAT}(H_2) && \text{(step 3, Lemma 1.8)}\\ &\geq \beta_3 \cdot \beta_2\sqrt{t} \cdot \min(\text{UNSAT}(H_1), \frac{1}{t}) && \text{(step 2, Lemma 1.6)}\\ &\geq \beta_3 \cdot \beta_2\sqrt{t} \cdot \min(\beta_1\text{UNSAT}(G), \frac{1}{t}) && \text{(step 1, Lemma 1.7)}\\ &\geq \min(2 \cdot \text{UNSAT}(G), \alpha)\end{aligned}$$

Finally, let us verify that each of the three steps incurs a blowup that is linear in the size of G. In step 1 this is immediate from Lemma 1.7. In step 2, since $deg(H_1) = d$ and t are independent of the size of G, the number of edges in $H_2 = (H_1)^t$ is equal to the number of edges in H_1 times d^{t-1} (this factor depends on $|\Sigma|$ but that is fine). In step 3, the total size grows by a factor c that depends on the alphabet size of H_2, which equals $|\Sigma^{d^{\lceil t/2 \rceil}}|$, and on $\mathcal{P}$. Again, both are independent of the size of G. Altogether, it is safe to write $size(G') \leq C \cdot size(G)$ where the factor C ultimately depends only on $|\Sigma|$ and on some global constants. ■

As a corollary of the main theorem we can immediately prove the PCP theorem,

Theorem 1.2 (Inapproximability version of PCP Theorem) *There are constants $q > 1$ and $|\Sigma| > 1$ such that given a collection $\mathcal{C}$ of q-ary constraints over an alphabet Σ, it is NP-hard to decide whether $\text{UNSAT}(\mathcal{C}) = 0$ or $\text{UNSAT}(\mathcal{C}) \geq \frac{1}{2}$.*

Proof: We reduce from constraint graph satisfiability. According to Proposition 1.4 it is NP-hard to decide if for a given constraint graph G with $|\Sigma| = 3$, $\text{UNSAT}(G) = 0$ or not. So let G be an instance of constraint-graph satisfiability with $|\Sigma| = 3$, and denote $n = size(G)$. The basic idea is to repeatedly apply the main theorem until the unsat-value becomes a constant fraction.

Let $G_0 = G$ and for $i \geq 1$ let G_i be the result of applying to G_{i-1} the transformation guaranteed by Theorem 1.5. Then for $i \geq 1$ G_i is a constraint graph with alphabet Σ_0. Let E_0 be the edge-set of G_0, and let $k = \log|E_0| = O(\log n)$. Observe that the size of G_i for $i \leq k = O(\log n)$ is at most $C^i \cdot size(G_0) = \text{poly}(n)$.

Completeness is easy: if $\text{UNSAT}(G_0) = 0$ then $\text{UNSAT}(G_i) = 0$ for all i. For soundness, assume $\text{UNSAT}(G_0) > 0$. If for some $i^* < k$, $\text{UNSAT}(G_{i^*}) \geq \alpha/2$ then the main theorem implies that for all $i > i^*$ $\text{UNSAT}(G_i) \geq \alpha$. For all other i it follows by induction that

$$\text{UNSAT}(G_i) \geq \min(2^i\, \text{UNSAT}(G_0), \alpha)\,.$$

If $\text{UNSAT}(G_0) > 0$ then $\text{UNSAT}(G_0) \geq \frac{1}{|E_0|}$, so surely $2^k\text{UNSAT}(G_0) > \alpha$. Thus $\text{UNSAT}(G_k) \geq \alpha$.

This proves that gap constraint satisfaction is NP-hard, for two-variable constraints and alphabet size $|\Sigma_0|$.

To get to a gap between $\text{UNSAT}(\mathcal{C}) = 0$ and $\text{UNSAT}(\mathcal{C}) \geq \frac{1}{2}$, one can apply simple (sequential) repetition $u = 1/\log(\frac{1}{1-\alpha}) = O(1)$ times. I.e., create a new constraint system $\mathcal{C}$ consisting of ANDs of all possible u-tuples of the constraints in G_k. This creates a system of $2u$-ary constraints that has the desired gap.

■

4 Preprocessing

In this section we describe how to (rather easily) turn any constraint graph into a 'nicely-structured' one. We define a transformation on constraint graphs, taking G to $\mathrm{prep}(G)$ that consists of two simple steps

$$G \to \mathrm{prep}_1(G) \to \mathrm{prep}_2(\mathrm{prep}_1(G)).$$

These transformation are described in Definitions 4.1 and 4.2 below. The first transformation converts the graph into a constant degree (regular) graph. The second transformation converts it into an expander. The properties of each transformation are stated and proved in Lemmas 4.1 and Lemma 4.2 respectively, which together give an immediate proof for Lemma 1.7.

Definition 4.1 *Let $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $\mathrm{prep}_1(G) = \langle (V',E'), \Sigma, \mathcal{C}' \rangle$ is defined as follows.*

- *Vertices: For each $v \in V$ let $[v] = \{ (v,e) \mid e \in E$ is incident on $v\}$, and set $V' = \cup_{v \in v} [v]$.*
- *Edges: For each $v \in V$ let X_v be a d-regular graph on vertex set $[v]$ and edge expansion at least h_0 (as guaranteed by Lemma 2.1). Let $E_1 = \cup_{v \in V} E(X_v)$ and set*

  $$E_2 = \left\{ \left\{ (v,e), (v',e) \right\} \,\middle|\, e = \left\{ v, v' \right\} \in E \right\}.$$

  Finally let $E' = E_1 \cup E_2$.
- *Constraints: The constraints are $\mathcal{C}' = \{c(e')\}_{e' \in E'}$ where $c(e')$ is defined as follows:*
  - *If $e' \in E_1$ then $c(e')$ is an equality constraint: $c(e') = \{ (a,a) \mid a \in \Sigma\}$.*
  - *If $e' = \{(v,e),(v',e)\} \in E_2$ then $c(e') = c(e) \in \mathcal{C}$.*

In words, the constraint graph $\mathrm{prep}_1(G)$ is obtained from G by blowing up each vertex into a cloud of as many vertices as its degree. Two clouds are connected by one edge if the original vertices were adjacent, and the vertices within a cloud are connected by expander edges. The constraints on external edges (between clouds) remain the same, and 'internal' constraints (within a cloud) enforce equality.

Lemma 4.1 *Let $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. Then $G' = \mathrm{prep}_1(G)$ is a (d_0+1)-regular constraint graph $G' = \langle (V',E'), \Sigma, \mathcal{C}' \rangle$ such that $|V'| \le 2\,|E|$ and*

$$c \cdot \mathrm{UNSAT}(G) \le \mathrm{UNSAT}(G') \le \mathrm{UNSAT}(G) \tag{1}$$

for some global constants $d_0, c > 0$.

Moreover, for any assignment $\sigma' : V' \to \Sigma$ let $\sigma : V \to \Sigma$ be defined according to the plurality value,

$$\forall v \in V, \quad \sigma(v) \overset{\triangle}{=} \arg\max_{a \in \Sigma} \left\{ \Pr_{(v,e)\in[v]} \left[\sigma'(v,e) = a \right] \right\}. \tag{2}$$

Then $c \cdot \mathrm{UNSAT}_\sigma(G) \le \mathrm{UNSAT}_{\sigma'}(G')$.

This lemma relies on a well-known 'expander-replacement' transformation due to [26], and we include a proof for the sake of completeness.

Proof: It is immediate to see that G' is $d = d_0 + 1$ regular. Every non self-loop edge in E gives rise to two endpoints, so clearly $|V'| \le 2\,|E|$. We proceed to prove (1).

The (completeness) upper bound $\text{UNSAT}(G') \le \text{UNSAT}(G)$ is easy: An assignment $\sigma : V \to \Sigma$ can be extended to an assignment $\sigma' : V' \to \Sigma$ by

$$\forall (v,e) \in V', \quad \sigma'(v,e) \stackrel{\triangle}{=} \sigma(v).$$

Clearly σ' does not violate the constraints corresponding to edges in E_1, and it violates exactly $\text{UNSAT}(G)\,|E_2|$ constraints corresponding to E_2. Thus

$$\text{UNSAT}(G') \le \frac{\text{UNSAT}(G)\,|E_2|}{|E_1| + |E_2|} \le \text{UNSAT}(G).$$

The (soundness) lower bound $c \cdot \text{UNSAT}(G) \le \text{UNSAT}(G')$ in (1) follows from the second part of the lemma, which we prove next. The intuitive idea is that the expander edges "penalize" assignments σ' that do not assign the same value to all copies of v; forcing σ' to behave essentially like an assignment σ for G.

Let us first observe that

$$\left|E'\right| \le d\,|E|$$

where the inequality would have been equality were there no self-loops in G.

Fix an assignment $\sigma' : V' \to \Sigma$, and let $\sigma : V \to \Sigma$ be defined according to (2). In other words $\sigma(v)$ is the most popular value among the values occurring in v's cloud. Let $F \subseteq E$ be the edges of G whose constraints reject σ, and let $F' \subseteq E'$ be the edges of G' whose constraints reject σ'. Let $S \subseteq V'$ be the set of vertices of G' whose value disagrees with the plurality,

$$S = \bigcup_{v \in V} \left\{ (v,e) \in [v] \mid \sigma'(v,e) \ne \sigma(v) \right\}.$$

Suppose $e = \{v, v'\} \in F$. Then the edge $\{(v,e),(v',e)\}$ either belongs to F', or has at least one endpoint in S. Hence, for $\alpha \stackrel{\triangle}{=} \frac{|F|}{|E|} = \text{UNSAT}_\sigma(G)$,

$$\left|F'\right| + |S| \ge |F| = \alpha \cdot |E|. \tag{3}$$

There are two cases,

- If $|F'| \ge \frac{\alpha}{2}|E|$ we are done since $\frac{\alpha}{2}|E| \ge \frac{\alpha}{2d}|E'|$ and so $\text{UNSAT}_{\sigma'}(G') \ge \text{UNSAT}_\sigma(G)/2d$.
- Otherwise, $|F'| < \frac{\alpha}{2}|E|$, so by (3), $|S| \ge \frac{\alpha}{2}|E|$. Focus on one v, and let $S^v = [v] \cap S$. We can write S^v as a disjoint union of sets $S^v_a = \{(v,e) \in S^v \mid \sigma'(v,e) = a\}$. Since S is the set of vertices disagreeing with the plurality value, we have $|S^v_a| \le |[v]|/2$, so by the edge expansion of the appropriate expander X_{d_v}, $E(S^v_a, [v] \setminus S^v_a) \ge h_0 \cdot |S^v_a|$. All of the edges leaving S^v_a carry equality constraints that reject σ'. So there are at least $\frac{h_0}{2}\sum_v |S \cap [v]| = \frac{h_0}{2}|S| \ge \frac{\alpha h_0}{4}|E|$ edges that reject σ'. Since $|E| \ge |E'|/d$, we get $\text{UNSAT}_{\sigma'}(G') \ge \frac{h_0}{4d}\text{UNSAT}_\sigma(G)$.

We have completed the proof, with $c = \min(\frac{1}{2d}, \frac{h_0}{4d})$. ■

We now turn to the second transformation, converting a constraint graph into an expander with self-loops.

Definition 4.2 *Let $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $\text{prep}_2(G) = \langle (V,E'), \Sigma, \mathcal{C}' \rangle$ is defined as follows.*

- *Vertices: The vertices remain the same.*

- *Edges: Let X be a ${d_0}'$-regular graph on vertex set V and edge set E_1, with $\lambda(X) < \lambda_0 < {d_0}'$ (as guaranteed by Corollary 2.4). Let $E_2 = \{\{v, v\} \mid v \in V\}$. Finally, let $E' = E \cup E_1 \cup E_2$ (where E' is a multiset allowing parallel edges).*
- *Constraints: The constraints are $\mathcal{C}' = \{c(e')\}_{e' \in E'}$ where $c(e')$ is defined as follows. If $e' \in E$ then $c(e')$ is just like before. Otherwise, $c(e')$ is the null constraint (always satisfied).*

Lemma 4.2 *There are global constants ${d_0}' > \lambda_0 > 0$ such that for any d-regular constraint graph G, the constraint graph $G' = \mathrm{prep}_2(G)$ has the following properties.*

- *G' is $(d + {d_0}' + 1)$-regular, has a self-loop on every vertex, and $\lambda(G') \le d + \lambda_0 + 1 < deg(G')$,*
- *$size(G') = O(size(G))$,*
- *For every $\sigma : V \to \Sigma$, $\frac{d}{d+{d_0}'+1} \cdot \mathrm{UNSAT}_\sigma(G) \le \mathrm{UNSAT}_\sigma(G') \le \mathrm{UNSAT}_\sigma(G)$.*

Proof: Clearly, G' is $d + {d_0}' + 1$-regular and each vertex has a self-loop. To bound $\lambda(G')$ we rely on the Rayleigh quotient (see Lemma 2.2),

$$\lambda(G) = \max_{\|x\|=1, x \perp \vec{1}} |\langle x, A_G x\rangle| ,$$

where A_G is the adjacency matrix of G. Clearly, if we denote adjacency matrix of X, G' by $A_X, A_{G'}$ respectively, then $A_{G'} = A_G + I + A_X$, where I is the identity matrix. Therefore,

$$\begin{aligned}\lambda(G') = \max_{\|x\|=1, x \perp \vec{1}} |\langle x, A_{G'} x\rangle| &\le \max_{\|x\|=1, x \perp \vec{1}} |\langle x, A_G x\rangle| + \max_{\|x\|=1, x \perp \vec{1}} |\langle x, I x\rangle| + \max_{\|x\|=1, x \perp \vec{1}} |\langle x, A_X x\rangle| \\ &= \lambda(G) + \lambda(I) + \lambda(X) \le d + 1 + \lambda_0.\end{aligned}$$

Finally, fix $\sigma : V \to \Sigma$. Since the new edges are always satisfied and since we increased the total number of edges by at most a factor $c' = \frac{d+{d_0}'+1}{d}$, the fraction of unsatisfied constraints cannot increase, and drops by at most c'. ■

Proof:(of Lemma 1.7) Let $G' = \mathrm{prep}_2(\mathrm{prep}_1(G))$. The lemma is proven with $\beta_1 = c \cdot \frac{d}{d+{d_0}'+1}$ by quoting Lemmas 4.1 and 4.2. ■

We conclude with a stronger corollary of Lemmas 4.1 and 4.2 that will be useful in Section 8.

Corollary 4.3 *Let $\beta_1 > 0$ be the constant from Lemma 1.7. Fix a constraint graph G, and let $G' = \mathrm{prep}(G)$. Let V be the vertices of G and let V' be the vertices of G'. For any assignment $\sigma' : V' \to \Sigma$, let $\sigma : V \to \Sigma$ be defined according to Equation (2). Then, $\mathrm{UNSAT}_{\sigma'}(G') \ge \beta_1 \cdot \mathrm{UNSAT}_\sigma(G)$.*

Proof: Let $G_1 = \mathrm{prep}_1(G)$ and $G_2 = \mathrm{prep}_2(G_1)$. By Lemma 4.2, for every assignment $\sigma' : V' \to \Sigma$

$$\frac{d}{d + {d_0}' + 1} \cdot \mathrm{UNSAT}_{\sigma'}(G_1) \le \mathrm{UNSAT}_{\sigma'}(G_2).$$

Moreover, by the second part of Lemma 4.1, we see that if we define $\sigma : V \to \Sigma$ according to Equation (2) then

$$c \cdot \mathrm{UNSAT}_\sigma(G) \le \mathrm{UNSAT}_{\sigma'}(G_1).$$

Combining the two inequalities,

$$c \cdot \frac{d}{d + {d_0}' + 1} \cdot \mathrm{UNSAT}_\sigma(G) \le \frac{d}{d + {d_0}' + 1} \cdot \mathrm{UNSAT}_{\sigma'}(G_1) \le \mathrm{UNSAT}_{\sigma'}(G_2).$$

Noting that $G' = G_2$, and that $\beta_1 = c \cdot \frac{d}{d+{d_0}'+1}$ completes the proof. ■

5 Alphabet Reduction by Composition

In this section we describe a transformation on constraint graphs that reduces the alphabet size, while roughly maintaining the unsat-value. We rely on *composition* which is an essential component in the construction of PCPs. To understand composition let us ignore the underlying graph structure of a constraint graph G, and view it simply as a system of constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$ over a set of variables X.

Let us step back for a moment and recall our overall goal of proving the PCP Theorem. What we seek is a reduction from any NP language L to gap constraint satisfaction. Such a reduction is a polynomial-time algorithm that inputs an instance x, and generates a system of constraints $\mathcal{C}$ with the following gap property: an input $x \in L$ translates to a system $\mathcal{C}$ for which $\text{UNSAT}(\mathcal{C}) = 0$, and an input $x \notin L$ translates to a system $\mathcal{C}$ for which $\text{UNSAT}(\mathcal{C}) > \alpha$, for some $\alpha > 0$.

Suppose we have such a "PCP" reduction $\mathcal{P}$ that is not necessarily efficient: the number of constraints $\mathcal{P}$ generates may be super-polynomial in its input size. Nevertheless, suppose the constraints generated by $\mathcal{P}$ are always over a small alphabet Σ_0, with (say) $|\Sigma_0| = 8$. How would such a "PCP"-reduction $\mathcal{P}$ be used for alphabet reduction?

Let G be a constraint graph with constraints $c_1, \ldots, c_m$ over alphabet Σ. First, we cast the satisfiability of each c_i as an NP statement, and then we feed it to $\mathcal{P}$. The output of $\mathcal{P}$ is a constraint graph G_i with alphabet size 8. It has the property that if c_i is satisfiable then so is G_i, and otherwise $\text{UNSAT}(G_i) \geq \alpha$. The final constraint graph denoted $G \circ \mathcal{P}$ would be some form of union of these G_i's that guarantees a good relation between the unsat value of G and that of the new graph $G \circ \mathcal{P}$. In particular, we would like to have the following properties:

- (Perfect Completeness:) If $\text{UNSAT}(G) = 0$ then $\text{UNSAT}(G \circ \mathcal{P}) = 0$
- (Soundness:) There is some constant ϵ such that $\text{UNSAT}(G \circ \mathcal{P}) \geq \epsilon \cdot \text{UNSAT}(G)$.

There is a subtle issue of consistency which will be discussed shortly below. Before that, let us convince ourselves that the transformation makes sense in terms of efficiency. Surely, our goal of reducing the size of the alphabet from $|\Sigma|$ to $|\Sigma_0| = 8$ has been achieved. What is the size of $G \circ \mathcal{P}$? Note that the size of each c_i that is fed to $\mathcal{P}$ can be bounded by some function that depends on $|\Sigma|$. Thus, the size of each G_i can be bounded by another function of $|\Sigma|$ and $\mathcal{P}$, denoted $c(\mathcal{P}, |\Sigma|)$, that depends on the efficiency of $\mathcal{P}$. In our context $|\Sigma|$ remains bounded as the size of G grows, so asymptotically the output of this procedure is only larger than the input by a constant factor (with $c(\mathcal{P}, |\Sigma|)$ factoring into the constant).

Consistency. The soundness property defined above will not be satisfied if we run $\mathcal{P}$ on each constraint c_i and take the (disjoint) union of the constraint graphs G_i. It is possible that the system of constraints $\{c_1, \ldots, c_m\}$ has a non-zero unsat value which will not carry over to $\cup G_i$ if, say, each constraint c_i is satisfiable on its own. The problem stems from the fact that we are not interested in the satisfiability of each c_i but rather in their satisfiability *simultaneously by the same assignment*. Therefore, when we run $\mathcal{P}$ on each c_i we need a mechanism that causes the assignments for the various graphs G_i to be "consistent" with each other, i.e., to refer to *the same* assignment for the original variables.

This issue has been handled before in a modular fashion by making stronger requirements on the reduction $\mathcal{P}$. Such more-restricted reductions are called PCPs of Proximity in [7] or Assignment Testers in [11]. Below we repeat Definition 2.2 of an assignment tester. Essentially, using an assignment-tester reduction $\mathcal{P}$ will force the different constant-size constraint graphs to have common vertices, and that will ensure consistency. For an exposition as to why assignment-testers are well-suited for composition, as well as a proof of a generic composition theorem, please see [7, 11].

Definition 2.2 (Assignment Tester) *An* Assignment Tester *with alphabet Σ_0 and rejection probability $\epsilon > 0$ is an algorithm $\mathcal{P}$ whose input is a circuit Φ over Boolean variables X, and whose output is a constraint graph $G = \langle (V, E), \Sigma_0, \mathcal{C} \rangle$ such that*[3] *$V \supset X$, and such that the following hold. Let $V' = V \setminus X$, and let $a : X \to \{0, 1\}$ be an assignment.*

- *(Completeness) If $a \in \mathrm{SAT}(\Phi)$, there exists $b : V' \to \Sigma_0$ such that $\mathrm{UNSAT}_{a \cup b}(G) = 0$.*
- *(Soundness) If $a \notin \mathrm{SAT}(\Phi)$ then for all $b : V' \to \Sigma_0$, $\mathrm{UNSAT}_{a \cup b}(G) \geq \epsilon \cdot \mathrm{rdist}(a, \mathrm{SAT}(\Phi))$.*

We remark that our definition of the rejection probability is stronger than the standard definition in the literature. Here it is really the ratio between the the probability of rejection and the distance of the given assignment from a satisfying one.

We prove in Section 7 that such an object exists:

Theorem 5.1 *There is some $\epsilon > 0$ and an explicit construction of an assignment tester $\mathcal{P}$ with alphabet $\Sigma_0 = \{0, 1\}^3$ and rejection probability ϵ.*

Notice that no statement was made on the running time of $\mathcal{P}$, and none will be necessary.

Let us now define the composition between a constraint graph G and an assignment tester $\mathcal{P}$. The definition requires an auxiliary error correcting code $\mathrm{e} : \Sigma \to \{0, 1\}^\ell$. We recall the following standard definitions. An error correcting code is said to have linear dimension if there is some constant $c > 0$ such that $\ell \leq c \cdot \log_2 \Sigma$. It is said to have relative distance $\rho > 0$ if for every $a_1 \neq a_2 \in \Sigma$, the strings $\mathrm{e}(a_1)$ and $\mathrm{e}(a_2)$ differ on at least $\rho\ell$ bits, namely $\mathrm{rdist}(\mathrm{e}(a_1), \mathrm{e}(a_2)) \geq \rho$. Two ℓ-bit strings s_1, s_2 are said to be δ-far (resp. δ-close) if $\mathrm{rdist}(s_1, s_2) \geq \delta$ (resp. if $\mathrm{rdist}(s_1, s_2) \leq \delta$).

Definition 5.1 (Composition) *Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a constraint graph and let $\mathcal{P}$ be an assignment tester. Let $\mathrm{e} : \Sigma \to \{0, 1\}^\ell$ be an arbitrary encoding with linear dimension and relative distance $\rho > 0$. The constraint graph $G \circ \mathcal{P} = \langle (V', E'), \Sigma_0, \mathcal{C}' \rangle$ is defined in two steps.*

1. *(Robustization:) First, we convert each constraint $c(e) \in \mathcal{C}$ to a circuit $\tilde{c}(e)$ as follows. For each variable $v \in V$, let $[v]$ be a fresh set of ℓ Boolean variables. For each edge $e = (v, w) \in E$, $\tilde{c}(e)$ will be a circuit on 2ℓ Boolean input variables $[v] \cup [w]$. The circuit $\tilde{c}(e)$ will output 1 iff the assignment for $[v] \cup [w]$ is the legal encoding via e of an assignment for v and w that would have satisfied c.*
2. *(Composition:) Run the assignment tester $\mathcal{P}$ on each $\tilde{c}(e)$. Let $G_e = \langle (V_e, E_e), \Sigma_0, \mathcal{C}_e \rangle$ denote the resulting constraint graph, and recall that $[v] \cup [w] \subset V_e$. Assume, wlog, that E_e has the same cardinality for each e. Finally, define the new constraint graph $G \circ \mathcal{P} = \langle (V', E'), \Sigma_0, \mathcal{C}' \rangle$, by*

$$V' = \bigcup_{e \in E} V_e, \qquad E' = \bigcup_{e \in E} E_e, \qquad \mathcal{C}' = \bigcup_{e \in E} \mathcal{C}_e \,.$$

Our main lemma in this section is the following,

Lemma 1.8 (Composition) *Assume the existence of an assignment tester $\mathcal{P}$, with constant rejection probability $\varepsilon > 0$, and alphabet Σ_0, $|\Sigma_0| = O(1)$. There exist a constant $\beta_3 > 0$ that depends only on $\mathcal{P}$, and a constant $c(\mathcal{P}, |\Sigma|)$ that depends only on $\mathcal{P}$ and $|\Sigma|$, such that the following holds. Given any*

[3] In a constraint graph, the set V plays a double role of both variables and vertices. By $V \supset X$ it is meant that some of the vertices of V are identified with the X variables.

constraint graph $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$, one can compute, in time that is linear in $size(G)$, the constraint graph $G' = G \circ \mathcal{P}$, such that $size(G') = c(\mathcal{P}, |\Sigma|) \cdot size(G)$, and

$$\beta_3 \cdot \text{UNSAT}(G) \leq \text{UNSAT}(G') \leq \text{UNSAT}(G).$$

Proof: First, let us verify that $G' = G \circ \mathcal{P}$ can be computed in time linear in $size(G)$. The first step (robustization) consists of $|E|$ steps of converting $c(e)$ to a circuit $\tilde{c}(e)$. This circuit computes a Boolean function on 2ℓ Boolean variables. Thus, each conversion can clearly be done in time $2^{O(\ell)}$, which is a factor that depends ultimately only on $|\Sigma|$ and not on $size(G)$. In the second step, we feed each $\tilde{c}(e)$ to $\mathcal{P}$, obtaining the constraint graph G_e. Even if the running time of $\mathcal{P}$ is huge compared to its input length, this makes no difference. The reason is that the size of the input to $\mathcal{P}$ is bounded by some absolute constant, again depending on $|\Sigma|$, and therefore the size of the output is bounded by some other absolute constant (which equals the maximal output length ranging over all of the finitely many different inputs). Since the blowup factor depends only on $|\Sigma|$ and on $\mathcal{P}$ we can write

$$size(G') = c(\mathcal{P}, |\Sigma|) \cdot size(G).$$

It remains to be seen that $\beta_3 \cdot \text{UNSAT}(G) \leq \text{UNSAT}(G') \leq \text{UNSAT}(G)$. The proof is straightforward and follows exactly the proof of the composition theorem in [11].

Let us start with the easy part of proving $\text{UNSAT}(G') \leq \text{UNSAT}(G)$. Let $\sigma : V \to \Sigma$ be an assignment for G such that $\text{UNSAT}(G) = \text{UNSAT}_\sigma(G)$. We construct an assignment $\sigma' : V' \to \Sigma_0$ by following the two steps in Definition 5.1. Recall that each vertex v was replaced by a set of vertices $[v]$. For each $v \in V$, we set

$$\sigma'([v]) = \mathrm{e}(\sigma(v)) \in \{0,1\}^\ell$$

where $\sigma'([v])$ means the concatenation of $\sigma'(y)$ for all $y \in [v]$. It remains to define values for σ' on

$$\bigcup_{e=(u,v)\in E} (V_e \setminus ([u] \cup [v]).$$

If $e = (u,v) \in E$ is such that $c(e)$ is satisfied by σ, then by definition the circuit $\tilde{c}(e)$ is satisfied by σ' restricted to $[u] \cup [v]$. Then, according to the completeness property of $\mathcal{P}$, there is an extension assignment for $V_e \setminus ([u] \cup [v])$ that satisfies all constraints in G_e. In other words, if we let a denote the restriction of σ' to $[u] \cup [v]$, then there is some $b : V_e \setminus ([u] \cup [v]) \to \Sigma_0$ such that $\text{UNSAT}_{a\cup b}(G_e) = 0$. Define σ' to coincide with b on $V_e \setminus ([u] \cup [v])$.

For the remaining vertices (belonging to graphs G_e whose constraint $c(e)$ is unsatisfied by σ) define σ' arbitrarily. Since each E_e has the same cardinality, it is easy to see that $\text{UNSAT}_{\sigma'}(G') \leq \text{UNSAT}_\sigma(G)$. Therefore,

$$\text{UNSAT}(G') \leq \text{UNSAT}_{\sigma'}(G') \leq \text{UNSAT}_\sigma(G) = \text{UNSAT}(G).$$

We now move to the left inequality: $\beta_3 \cdot \text{UNSAT}(G) \leq \text{UNSAT}(G')$. We need to prove that every assignment for G' violates at least $\beta_3 \cdot \text{UNSAT}(G)$ fraction of G''s constraints. So let $\sigma' : V' \to \Sigma_0$ be a best assignment for G', i.e., such that $\text{UNSAT}_{\sigma'}(G') = \text{UNSAT}(G')$. We first extract from it an assignment $\sigma : V \to \Sigma$ for G by letting, for each $v \in V$, $\sigma(v)$ be a value whose encoding via e is closest to $\sigma'([v])$. Let $F \subseteq E$ be the edges of G whose constraints are falsified by σ. By definition, $\frac{|F|}{|E|} = \text{UNSAT}_\sigma(G) \geq \text{UNSAT}(G)$. Now let $e = (u,v) \in F$. We will show that at least a β_3 fraction of the constraints of the graph G_e are falsified by σ'. Recall that the constraint graph G_e is the output of $\mathcal{P}$ on input $\tilde{c}(e)$. Thus, we must analyze the distance of the assignment for $[u] \cup [v]$ from the set $\mathrm{SAT}(\tilde{c}(e))$ of assignments that satisfy $\tilde{c}(e)$. The main observation is that the restriction of σ' to $[u] \cup [v]$, is at least $\rho/4$-far from $\mathrm{SAT}(\tilde{c}(e))$ (where ρ denotes the relative distance of e). The reason is the definition of $\sigma(u)$ (resp. $\sigma(v)$) as the value whose encoding is closest to $\sigma'([u])$ (resp. $\sigma'([v])$). This means that at least a $\rho/2$ fraction of the bits in either $[u]$ or $[v]$ (or both) must be changed in order to change σ' into an assignment that satisfies $\tilde{c}(e)$. So

$$\mathrm{rdist}(\,\sigma'|_{[u]\cup[v]}, \mathrm{SAT}(\tilde{c}(e))\,) \geq \rho/4.$$

By the soundness property of $\mathcal{P}$, at least $\varepsilon \cdot \rho/4 = \Omega(1)$ fraction of the constraints in G_e are unsatisfied, and we set $\beta_3 = \varepsilon\rho/4 > 0$. Altogether,

$$\begin{aligned} \mathrm{UNSAT}(G') &= \mathrm{UNSAT}_{\sigma'}(G') \\ &= \frac{1}{|E|}\sum_{e\in E} \mathrm{UNSAT}_{\sigma'|_{V_e}}(G_e) \\ &\geq \frac{1}{|E|}\sum_{e\in F} \mathrm{UNSAT}_{\sigma'|_{V_e}}(G_e) \\ &\geq \beta_3\frac{|F|}{|E|} = \beta_3\mathrm{UNSAT}_{\sigma}(G) \geq \beta_3\mathrm{UNSAT}(G) \end{aligned}$$

where the second equality follows since $|E_e|$ is the same for all $e \in E$. ■

6 Amplification Lemma

In this section we prove the amplification lemma. In fact, we prove the following slightly stronger lemma from which Lemma 1.6 follows as an immediate corollary.

Lemma 6.1 *Let $\lambda < d$, and $|\Sigma|$ be arbitrary constants. There exists a constant $\beta_2 = \beta_2(\lambda, d, |\Sigma|) > 0$, such that for every $t \in \mathbb{N}$ and for every d-regular constraint graph $G = \langle (V,E), \Sigma, \mathcal{C}\rangle$ with self-loops and $\lambda(G) \leq \lambda$ the following holds. For every $\vec{\sigma} : V \to \Sigma^{d^{\lceil t/2\rceil}}$ let $\sigma : V \to \Sigma$ be defined according to "popular opinion" by setting, for each $v \in V$,*

$$\sigma(v) \stackrel{\triangle}{=} \max\arg_{a\in\Sigma}\left\{\Pr[\textit{A random } \lceil t/2\rceil\textit{-step walk from } v \textit{ reaches a vertex } w \textit{ for which } \vec{\sigma}(w)_v = a]\right\}. \quad (4)$$

where $\vec{\sigma}(w)_v \in \Sigma$ denotes the restriction of $\vec{\sigma}(w)$ to v. Then,

$$\mathrm{UNSAT}_{\vec{\sigma}}(G^t) \geq \beta_2\sqrt{t}\cdot\min\left(\mathrm{UNSAT}_{\sigma}(G)\,,\,\frac{1}{t}\right).$$

Throughout this section all constants, including those implicitly referred to by $O(\cdot)$ and $\Omega(\cdot)$ notation, are independent of t but may depend on d, λ and $|\Sigma|$. Also, let us assume for notational clarity that t is even.

Before we move to the proof of Lemma 6.1 let us see how it yields Lemma 1.6.

Proof of Lemma 1.6: Let $\vec{\sigma}$ an assignment for G^t with minimum unsat value. Then, for σ defined according to (4),

$$\mathrm{UNSAT}(G^t) = \mathrm{UNSAT}_{\vec{\sigma}}(G^t) \geq \beta_2\sqrt{t}\cdot\min\left(\mathrm{UNSAT}_{\sigma}(G)\,,\,\frac{1}{t}\right) \geq \beta_2\sqrt{t}\cdot\min\left(\mathrm{UNSAT}(G)\,,\,\frac{1}{t}\right)$$

where the first inequality is due to Lemma 6.1. ■

Let us provide some intuition for why Lemma 6.1 holds. Let us begin by a simple mental experiment. Fix an assignment $\sigma : V \to \Sigma$ for G, and consider the probability of choosing t edges in G independently at random and checking whether σ falsifies at least one of these edges. This probability is roughly t times larger than $\text{UNSAT}_\sigma(G)$. Moreover, since G is an expander graph, the probability remains (roughly) the same even if the t edges are chosen by taking a random length-t walk in G.

The graph G^t is constructed to simulate this behavior. It is not hard to see that if $\vec{\sigma} : V \to \Sigma^{d^{t/2}}$ were "faithful" to some underlying assignment $\sigma : V \to \Sigma$ (i.e. $\vec{\sigma}(v)_w = \sigma(w)$ for each w reachable from v by $t/2$ steps) then $\text{UNSAT}_{\vec{\sigma}}(G^t)$ is lower-bounded by the result of the mental experiment. The proof of Lemma 6.1 is more tricky since we must consider assignments $\vec{\sigma}$ that are not "faithful" to any underlying assignment.

The idea of the proof is as follows. Let us refer to the edges of G^t as *walks*, since they come from t-step walks in G, and let us refer to the edges of G as edges. Given an assignment for G^t, $\vec{\sigma} : V \to \Sigma^{d^{t/2}}$, we extract from it a new assignment $\sigma : V \to \Sigma$ by assigning each vertex v the most popular value among the "opinions" (under $\vec{\sigma}$) of v's neighbors. We then relate the fraction of edges falsified by this "popular-opinion" assignment σ to the fraction of walks falsified by $\vec{\sigma}$. The probability that a random edge rejects this new assignment is, by definition, at least $\text{UNSAT}(G)$. The idea is that a random walk passes through *at least one* rejecting edge with even higher probability. Moreover, we will show that if a walk does pass through a rejecting edge, it itself rejects with constant probability.

Proof of Lemma 6.1: Let $\vec{\sigma} : V \to \Sigma^{d^{t/2}}$ be any assignment for G^t. For each v, $\vec{\sigma}(v)$ assigns a vector of $d^{t/2}$ values in Σ, interpreted as values for every vertex w within distance $t/2$ of v. This can be thought of as the opinion of v about w. Define the assignment $\sigma : V \to \Sigma$ according to (4). Let X_v be a random variable that assumes a value a with probability that a $t/2$-step random walk from v ends at a vertex w for which $\vec{\sigma}(w)_v = a$. Then $\sigma(v) = a$ for a value a which maximizes $\Pr[X_v = a]$, and in particular

$$\Pr[X_v = \sigma(v)] \geq \frac{1}{|\Sigma|}. \tag{5}$$

As mentioned above, the assignment σ can be interpreted as being the "popular opinion" about v among v's neighbors.

Let F be a subset of edges that reject σ so that if $\text{UNSAT}_\sigma(G) < 1/t$ then $\frac{|F|}{|E|} = \text{UNSAT}_\sigma(G)$, and otherwise we take F to be an arbitrary subset of these edges, such that $|F| = \lfloor \frac{|E|}{t} \rfloor$. We have

$$\frac{|F|}{|E|} \leq \min(\text{UNSAT}_\sigma(G), 1/t) \tag{6}$$

where equality holds if we ignore the rounding error. From now on $\vec{\sigma}, \sigma, F$ will be fixed for the rest of the proof.

Let $\mathbf{E} = E(G^t)$ be the edge set of G^t. There is a one-to-one correspondence between edges $\mathbf{e} \in \mathbf{E}$ and walks of length t in G. With some abuse of notation we write $\mathbf{e} = (v_0, v_1, \ldots, v_t)$ where $(v_{i-1}, v_i) \in E$ for all $1 \leq i \leq t$.

Definition 6.1 *A walk* $\mathbf{e} = (v_0, \ldots, v_t) \in \mathbf{E}$ *is* hit *by its* i*-th edge if*

1. $(v_{i-1}, v_i) \in F$, *and*
2. *Both* $\vec{\sigma}(v_0)_{v_{i-1}} = \sigma(v_{i-1})$ *and* $\vec{\sigma}(v_t)_{v_i} = \sigma(v_i)$.

Let $I = \left\{ \frac{t}{2} - \sqrt{\frac{t}{2}} < i \le \frac{t}{2} + \sqrt{\frac{t}{2}} \right\} \subset \mathbb{N}$ be the set of "middle" indices. For each walk $\mathbf{e}$, we define $N(\mathbf{e})$ to be the number of times $\mathbf{e}$ is hit in its middle portion:

$$N(\mathbf{e}) = |\{ i \in I \mid \mathbf{e} \text{ is hit by its } i\text{-th edge}\}| \ .$$

$N(\mathbf{e})$ is an integer between 0 and $\sqrt{2t}$. Clearly, $N(\mathbf{e}) > 0$ implies that $\mathbf{e}$ rejects under $\vec{\sigma}$ (because having $\mathbf{e}$ hit by the i-th edge means $(v_{i-1}, v_i) \in F$ and so $\sigma(v_{i-1})$ is inconsistent with $\sigma(v_i)$ which carries over to the constraint on $\vec{\sigma}(v_0)$ and $\vec{\sigma}(v_t)$). Thus,

$$\Pr_{\mathbf{e}}[N(\mathbf{e}) > 0] \le \Pr_{\mathbf{e}}[\mathbf{e} \text{ rejects } \vec{\sigma}] = \text{UNSAT}_{\vec{\sigma}}(G^t) \, .$$

We will prove

$$\Omega(\sqrt{t}) \cdot \frac{|F|}{|E|} \le \Pr_{\mathbf{e}}[N(\mathbf{e}) > 0] \, . \tag{7}$$

Combining the above with (6) we get

$$\Omega(\sqrt{t}) \cdot \min(\text{UNSAT}_{\sigma}(G), \frac{1}{t}) \le \Omega(\sqrt{t}) \cdot \frac{|F|}{|E|} \le \Pr_{\mathbf{e}}[N(\mathbf{e}) > 0] \le \text{UNSAT}_{\vec{\sigma}}(G^t)$$

which gives the lemma.

We will prove (7) by estimating the first and second moments of the random variable N,

Lemma 6.2

$$\mathbb{E}_{\mathbf{e}}[N(\mathbf{e})] \ge \Omega(\sqrt{t}) \cdot \frac{|F|}{|E|}$$

Lemma 6.3

$$\mathbb{E}_{\mathbf{e}}[(N(\mathbf{e}))^2] \le O(\sqrt{t}) \cdot \frac{|F|}{|E|}$$

Equation (7) follows by Fact 2.6,

$$\Pr[N(\mathbf{e}) > 0] \ge \mathbb{E}^2[N(\mathbf{e})]/\mathbb{E}[(N(\mathbf{e}))^2] = \Omega(\sqrt{t}) \cdot \frac{|F|}{|E|}.$$

■

6.1 Proof of Lemma 6.2

Define an indicator variable N_i by setting $N_i(\mathbf{e}) = 1$ iff the walk $\mathbf{e}$ is hit by its i-th edge, as in definition 6.1. Recall $I = \left\{ \frac{t}{2} - \sqrt{\frac{t}{2}} < j \le \frac{t}{2} + \sqrt{\frac{t}{2}} \right\}$. Clearly, $N = \sum_{i \in I} N_i$. In order to estimate $\mathbb{E}[N]$ we will estimate $\mathbb{E}[N_i]$, and use linearity of expectation.

Fix $i \in I$. In order to estimate $\mathbb{E}[N_i]$ we choose $\mathbf{e} \in \mathbf{E}$ according to the following distribution:

1. Choose $e = (u, v) \in E$ uniformly at random.
2. Choose a random walk of length $i-1$ starting from u, denote it by $(u = v_{i-1}, v_{i-2}, \ldots, v_1, v_0)$.
3. Choose a random walk of length $t-i$ starting from v, denote it by $(v = v_i, v_{i+1}, \ldots, v_t)$.
4. Output the walk $\mathbf{e} = (v_0, \ldots, v_t)$

Since G is d-regular this is no other than the uniform distribution on $\mathbf{E}$. According to Definition 6.1, $\mathbf{e}$ is hit by its i-th edge iff $(u,v) \in F$ and $\vec{\sigma}(v_0)_u = \sigma(u)$ and $\vec{\sigma}(v_t)_v = \sigma(v)$.

Clearly, the probability that step 1 results in an edge $(u,v) \in F$ equals exactly $\frac{|F|}{|E|}$. Observe also that the choice of v_0 in step 2 *only depends on* u, and the choice of v_t in step 3 *only depends on* v. Therefore

$$\Pr[N_i > 0] = \frac{|F|}{|E|} \cdot p_u \cdot p_v \tag{8}$$

where $p_u = \Pr_{v_0}\left[\vec{\sigma}(v_0)_u = \sigma(u)\right]$ and $p_v = \Pr_{v_t}[\vec{\sigma}(v_t)_v = \sigma(v)]$. It remains to analyze p_u and p_v. Let us focus on p_u as the case of p_v is symmetric.

Define a random variable $X_{u,\ell}$ as follows. $X_{u,\ell}$ takes a value $a \in \Sigma$ with probability that a random ℓ-step walk from u ends in a vertex w for which $\vec{\sigma}(w)_u = a$. In these terms $p_u = \Pr[X_{u,i-1} = \sigma(u)]$, (and $p_v = \Pr[X_{v,t-i} = \sigma(v)]$). Recall that by definition $\sigma(u)$ equals a value $a \in \Sigma$ that maximizes $\Pr[X_{u,t/2} = a]$. In particular, $\Pr[X_{u,t/2} = \sigma(u)] \geq \frac{1}{|\Sigma|}$. For $i - 1 = t/2$ it follows immediately that $p_u \geq 1/|\Sigma|$.

We will prove that for all ℓ

$$\text{If} \quad |\ell - t/2| \leq \sqrt{t/2} \quad \text{then} \quad \Pr[X_{u,\ell} = a] > \frac{\tau}{2} \cdot \Pr[X_{u,t/2} = a] \tag{9}$$

for some absolute constant $\tau > 0$ to be determined. The intuition for (9) is that the self-loops of G make the distribution of vertices reached by a random $t/2$-step walk from u roughly the same as distribution on vertices reached by an ℓ-step walk from u, for any $\ell \in I$.

Fix $\ell \in I$. Mark one self-loop on each vertex, and observe that any length-ℓ walk from u in G can be equivalently described by (i) specifying in which steps the marked edges were traversed, and then (ii) specifying the remaining steps conditioned on choosing only non-marked edges. Let $X'_{u,k}$ be a random variable that assumes a value a with probability that a k-step random walk *conditioned on walking only on non-marked edges* reaches a vertex w for which $\vec{\sigma}(w)_u = a$. In other words, for a binomial variable $B_{\ell,p}$ with $\Pr[B_{\ell,p} = k] = \binom{\ell}{k} p^k (1-p)^{\ell-k}$ and $p = 1 - 1/d$,

$$\Pr[X_{u,\ell} = a] = \sum_{k=0}^{\ell} \Pr[B_{\ell,p} = k] \Pr[X'_{u,k} = a]\,. \tag{10}$$

The point is that if $|\ell_1 - \ell_2|$ is small, then the distributions $B_{\ell_1,p}$ and $B_{\ell_2,p}$ are similar, as formalized in the following lemma:

Lemma 6.4 *For every $p \in (0,1)$ and $c > 0$ there exists some $\ell_0 > 0$ and $0 < \tau < 1$ such that if $\ell_0 < \ell_1 - \sqrt{\ell_1} \leq \ell_2 < \ell_1 + \sqrt{\ell_1}$, then*

$$\forall k, |k - p\ell_1| \leq c\sqrt{\ell_1}, \qquad \tau \leq \frac{\Pr[B_{\ell_1,p} = k]}{\Pr[B_{\ell_2,p} = k]} \leq \frac{1}{\tau}$$

The proof is a straightforward computation that follows from concentration properties of the binomial distribution, and can be found in Appendix A. We choose $c > 0$ so that

$$K = \left\{ k \in \mathbb{N} \;\middle|\; |k - pt/2| \leq c\sqrt{t/2} \right\}$$

is large: $\Pr_{k \sim B_{t/2,p}}[k \notin K] < \frac{1}{2|\Sigma|}$. Then we apply Lemma 6.4 with the constant $c > 0$, $p = 1 - \frac{1}{d}$, $\ell_1 = t/2$, and $\ell_2 = \ell$ and deduce for all $k \in K$,

$$\Pr[B_{\ell,p} = k] \geq \tau \cdot \Pr[B_{\frac{t}{2},p} = k]$$

where $0 < \tau < 1$ is the appropriate constant from the lemma.

We now have for any $\ell \in I$,

$$\begin{aligned}
\Pr[X_{u,\ell} = a] &\geq \sum_{k \in K} \Pr[B_{\ell,p} = k] \Pr[X'_{u,k} = a] \\
&\geq \tau \cdot \sum_{k \in K} \Pr[B_{t/2,p} = k] \Pr[X'_{u,k} = a] \\
&\geq \tau \cdot \left(\Pr[X_{u,t/2} = a] - \frac{1}{2\,|\Sigma|} \right) \geq \frac{\tau}{2} \cdot \Pr[X_{u,t/2} = a]
\end{aligned}$$

where the last inequality holds because of (5). This establishes (9), and so $p_u, p_v > \frac{\tau}{2|\Sigma|}$ because both $i-1, t-i$ are at most $\sqrt{t/2}$ away from $t/2$. Plugging this into Equation (8), we get $\mathbb{E}[N_i] \geq \frac{|F|}{|E|} \cdot \Omega(1)$, and this completes the proof of Lemma 6.2. ■

6.2 Proof of Lemma 6.3

For a walk $\mathbf{e}$, let $\mathbf{e}_i$ denote its i-th edge. In order to upper bound $\mathbb{E}_{\mathbf{e}}[N(\mathbf{e})^2]$ (all expectations are taken over uniform choice of $\mathbf{e}$) we define a random variable $Z(\mathbf{e}) = |\{ i \in I \mid \mathbf{e}_i \in F\}|$ that counts how many times $\mathbf{e}$ intersects F in the middle portion (recall $I = \left\{ \frac{t}{2} - \sqrt{\frac{t}{2}} < j \leq \frac{t}{2} + \sqrt{\frac{t}{2}} \right\}$). Clearly, $0 \leq N(\mathbf{e}) \leq Z(\mathbf{e})$ for all $\mathbf{e}$, so we will bound $\mathbb{E}[N(\mathbf{e})^2]$ using $\mathbb{E}[N(\mathbf{e})^2] \leq \mathbb{E}[Z(\mathbf{e})^2]$.

Let $Z_i = Z_i(\mathbf{e})$ be an indicator random variable that is 1 iff $\mathbf{e}_i \in F$. So $Z(\mathbf{e}) = \sum_{i \in I} Z_i(\mathbf{e})$, and by linearity of expectation,

$$\mathbb{E}_{\mathbf{e}}[Z(\mathbf{e})^2] = \sum_{i,j \in I} \mathbb{E}_{\mathbf{e}}[Z_i(\mathbf{e}) Z_j(\mathbf{e})] = \sum_{i \in I} \mathbb{E}[Z_i] + 2 \sum_{\substack{i < j \\ i,j \in I}} \mathbb{E}[Z_i Z_j] = |I| \frac{|F|}{|E|} + 2 \sum_{\substack{i < j \\ i,j \in I}} \mathbb{E}[Z_i Z_j] \tag{11}$$

As it turns out, $\mathbb{E}[Z^2]$ is not much larger than $\frac{|I||F|}{|E|} \approx \sqrt{t}\frac{|F|}{|E|}$. The intuitive reason is that since the graph G is an expander, correlations between the i-th and the j-th steps of a random walk cannot last long, so $\sum \mathbb{E}[Z_i Z_j]$ is small.

Proposition 6.5 *Fix $i, j \in I,\ i < j$, and $F \subseteq E$. Then,*

$$\mathbb{E}[Z_i Z_j] \leq \frac{|F|}{|E|} \left(\frac{|F|}{|E|} + |\lambda|^{j-i-1} \right).$$

Let us first see that combining the proposition with (11) completes the lemma. Indeed, since $|I| = \sqrt{2t}$ and since $\frac{|F|}{|E|} \leq \frac{1}{t}$,

$$\sum_{\substack{i < j \\ i,j \in I}} \mathbb{E}[Z_i Z_j] \leq \frac{|F|}{|E|} \sum_{\substack{i < j \\ i,j \in I}} \left(\frac{|F|}{|E|} + |\lambda|^{j-i} \right) < |I|^2 \left(\frac{|F|}{|E|} \right)^2 + |I| \frac{|F|}{|E|} \sum_{i=1}^{\sqrt{2t}} |\lambda|^i = O(\sqrt{t}) \cdot \frac{|F|}{|E|}$$

where the 'O' notation is hiding a constant that depends only on $|\lambda|$. Let us now prove the proposition.

Proof: Observe that $Z_i Z_j \in \{0, 1\}$, and $\Pr[Z_j = 1] = \frac{|F|}{|E|}$. Thus,

$$\mathbb{E}[Z_i Z_j] = \Pr[Z_i Z_j = 1] = \Pr[Z_i = 1] \Pr[Z_j = 1 \mid Z_i = 1] = \frac{|F|}{|E|} \cdot \Pr[Z_j = 1 \mid Z_i = 1].$$

Assume first $i = 1$ and $j > i$. By Proposition 2.5,

$$\Pr_{\mathbf{e}}[Z_j(\mathbf{e}) = 1 \mid Z_1(\mathbf{e}) = 1] \le \frac{|F|}{|E|} + |\lambda|^{j-2}$$

where $\lambda < 1$ is the normalized second eigenvalue of the graph G. Indeed, let $\mathbf{e} = (v_0, \ldots, v_t)$, conditioned on $Z_1(\mathbf{e}) = 1$ the distribution of v_1 is exactly the distribution K defined in Proposition 2.5, so the probability that the j-th edge in this walk is in F is at most $\frac{|F|}{|E|} + \left(\frac{|\lambda|}{d}\right)^{j-2}$.

If $j > i > 1$, we don't care where the random walk $\mathbf{e}$ visited during its first $i - 1$ steps, so we can ignore those steps. In other words the last $t - i + 1$ steps of a random walk of length t are a random walk of length $t - i + 1$. This is formalized by writing

$$\Pr_{|\mathbf{e}|=t}[Z_j(\mathbf{e}) = 1 \mid Z_i(\mathbf{e}) = 1] = \Pr_{|\mathbf{e}'|=t-i+1}[Z_{j-i+1}(\mathbf{e}') = 1 \mid Z_1(\mathbf{e}') = 1]\,.$$

Now by applying Proposition 2.5 on walks of length $t - i + 1$, the right hand side cannot exceed $\frac{|F|}{|E|} + |\lambda|^{j-i-1}$. ■

We conclude this section by commenting that there is a modification of our construction, due to Jaikumar Radhakrishnan, that allows one to replace $\sqrt{t}$ in Lemma 1.6 by t. This is obviously tight (up to the constant hidden in the O notation). Amplification by factor $\Theta(t)$ is achieved by the following modified definition of G^t: the vertices stay the same, and the alphabet is $\Sigma^{d^{t/2}}$ as before. The edges of G^t are weighted, and described by the following random process: choose a vertex v at random and choose a second vertex by taking a random walk form v that stops after each step with probability $1/t$. The constraints are defined as before. For details on how to analyze this construction the reader is referred to [28].

7 An Explicit Assignment Tester

In this section we prove Theorem 5.1, i.e., we outline a construction of an assignment tester $\mathcal{P}$. Let ψ be a Boolean circuit over Boolean variables $x_1, \ldots, x_s$. We describe an algorithm $\mathcal{P}$ whose input is ψ and whose output will be a constraint graph satisfying the requirements of Definition 2.2. We begin by introducing the Long-Code. Let

$$L = \{f : \{0,1\}^s \to \{0,1\}\}$$

be the set of all Boolean functions on s bits. Given a string $a = (a_1, \ldots, a_s) \in \{0,1\}^s$, define $A_a : L \to \{0,1\}$ by

$$\forall f \in L \quad A_a(f) = f(a)\,. \tag{12}$$

Each A_a itself can be viewed as a string of $|L|$ bits, and the set of strings $\{\, A_a \mid a \in \{0,1\}^s \}$ is an error-correcting code called *the Long-Code*. Recall that two ℓ-bit strings s_1, s_2 are said to be δ-far (resp. δ-close) from one another if $\mathrm{dist}(s_1, s_2) \ge \delta\ell$ (resp. if $\mathrm{dist}(s_1, s_2) \le \delta\ell$). It is not hard to see that if $a \ne a'$ then A_a and $A_{a'}$ are $\frac{1}{2}$-far from one another.

In fact we consider only so-called "folded" strings. A string A is said to be folded over true if for every f, $A(-f) = -A(f)$. A string A is said to be folded over $\psi : \{0,1\}^s \to \{0,1\}$ if for every f, $A(f) = A(f \wedge \psi)$. When ψ is clear from the context we say that a string A is "folded" if it is folded both over true and over ψ. Clearly, in order to specify a folded string $A : L \to \{0,1\}$ it is enough to specify it on the coordinates $L'_\psi \subset L$, defined as follows. For every pair $f, 1 - f \in L$, let L' contain exactly one of them, and set

$$L'_\psi = \left\{\, f \in L' \;\middle|\; f = f \wedge \psi \right\}.$$

We are now ready to state the Long-Code test theorem.

Theorem 7.1 *There exists a* Long-Code Test T *which is a randomized algorithm that has input a function* $\psi : \{0,1\}^s \to \{0,1\}$*, and also oracle access to a folded string* $A : L \to \{0,1\}$*.* T *reads the input* ψ *and tosses some random coins. Based on these it computes a three-bit predicate* $w : \{0,1\}^3 \to \{\mathsf{true}, \mathsf{false}\}$ *and three locations* $f_1, f_2, f_3 \in L$ *in which it queries the string* A*. It then outputs* $w(A(f_1), A(f_2), A(f_3))$*. Denote an execution of* T *with access to input* ψ *and string* A *by* $T^A(\psi)$*. Then the following hold,*

- *(Perfect completeness:) If* $a \in \{0,1\}^s$ *such that* $\psi(a) = 1$*, then* $\Pr[T^{A_a}(\psi) = \mathsf{true}] = 1$*.*
- *(Strong*[4] *soundness:) For every* $\delta \in [0,1]$*, if* $A : L \to \{0,1\}$ *is folded and at least* δ*-far from* A_a *for all* a *for which* $\psi(a) = 1$*, then* $\Pr[T^A(\psi) = \mathsf{false}] \geq \Omega(\delta)$*.*

For the sake of self-containedness, we include a proof of this theorem in Appendix B. We now proceed to construct a system of constraints based on the test T. This is done in two rather standard steps,

1. (Modified Test:) Let $X = \{x_1, \ldots, x_s\}$ be a set of s Boolean variables. Also, let there be a Boolean variable for each $f \in L'_\psi$. Since an assignment for these variables can be expanded into a *folded* assignment for L, we pretend from now on that we have a Boolean variable for every $f \in L$. We allow the test to access any variable indexed by L. When it accesses some variable $f \in L \setminus L'_\psi$ the value of the variable is determined by accessing the appropriate variable in L'_ψ. For example, if $1 - f \in L'_\psi$ then to read the value of f we access the value of $1 - f$ and negate the assignment. To summarize, from now on we ignore this issue and simply pretend that we have a variable for each $f \in L$ and that the assignment for these variables is guaranteed to be folded.

   Define a modified test T' as follows. Given input ψ and oracle access to a folded assignment $A : L \to \{0,1\}$ and an assignment $\sigma : X \to \{0,1\}$, run T on ψ and A with probability $1/2$, and otherwise choose a random $x_i \in X$ and a random $f \in L$, and test that $\sigma(x_i) = A(f) \oplus A(f + e_i)$.

2. (Creating the Constraints:) Introduce a new variable z_r per outcome r of the coin tosses of T'. These variables will take values in $\{0,1\}^3$, supposedly specifying the correct values of all three variables queried by T' on coin tosses r.

   We construct the following system of constraints: There will be a constraint for every possible choice of $z_r \in Z$ and a variable y of the three accessed by T' on coin toss r (so $y \in X \cup L$). This constraint will check that the assignment for z_r would have satisfied T', and that it is consistent with the assignment for y.

The algorithm $\mathcal{P}$ will output the constraint graph G whose vertices are $X \cup L \cup Z$, and whose constraints (and edges) are as specified above. The alphabet is $\Sigma_0 = \{0,1\}^3$, where the Boolean variables $X \cup L$ take values only in $\{000, 111\} \subset \Sigma_0$, identified with $\{0,1\}$ (i.e., a constraint involving $y \in X \cup L$ immediately rejects if the value of y is not in $\{000, 111\}$).

Lemma 7.2 *The reduction taking* $\psi : \{0,1\}^s \to \{0,1\}$ *to* G *is an assignment tester, with* $\Sigma_0 = \{0,1\}^3$ *and constant rejection probability* $\varepsilon > 0$*.*

Proof: Let us identify the Boolean variables of ψ with X, so the constraint graph G has the correct form according to Definition 2.2. We need to prove

- (Completeness) If $a \in \mathrm{SAT}(\psi)$, there exists $b : L \cup Z \to \Sigma_0$ such that $\textsc{unsat}_{a \cup b}(G) = 0$.

[4]We refer to 'strong' soundness as opposed to regular soundness, due to the stronger property of having the rejection probability proportional to the distance from a "good" string.

- (Soundness) If $a \notin \mathrm{SAT}(\psi)$ then for all $b : L \cup Z \to \Sigma_0$, $\textsc{unsat}_{a\cup b}(G) \geq \epsilon \cdot \mathrm{rdist}(a, \mathrm{SAT}(\psi))$.

The completeness part is easy. Let the assignment for the variables in L be A_a (defined in Equation (12)). It is then easy to assign the variables Z in a consistent manner.

For soundness, assume that $\sigma : X \to \{0,1\}$ is an assignment such that $\mathrm{rdist}(\sigma, \mathrm{SAT}(\psi)) = \delta$, for some $\delta > 0$. Fix any $b : L \cup Z \to \Sigma_0$, and denote $A = b|_L$. We claim

Proposition 7.3 $\Pr[T'^{A,\sigma}(\psi) = \mathsf{false}] = \Omega(\delta)$.

Proof: We observe that A is folded (by the discussion in item 1 above). Assume first that $A : L \to \{0,1\}$ is $\delta/2$ far from A_a for all $a \in SAT(\psi) \subset \{0,1\}^s$. Then by Theorem 7.1 T rejects with probability at least $\Omega(\delta)$, so T' rejects with probability at least half of that, which is also $\Omega(\delta)$. Otherwise, A is $\delta/2$-close to the long-code encoding of some $a' \in SAT(\psi)$. We now compare a' and σ which are both assignments for the variables of ψ. Since $a' \in SAT(\psi)$,

$$\Pr_i[\sigma(x_i) \neq a'(x_i)] = \mathrm{rdist}(\sigma, a') \geq \mathrm{rdist}(\sigma, SAT(\psi)] = \delta.$$

Now recall that with probability $1/2$, T' chooses a random i and a random f and checks that $A(f) \oplus A(f + e_i) = \sigma(x_i)$. Since A is $\frac{\delta}{2}$-close to $A_{a'}$, we have for all i:

$$\begin{aligned} \Pr_{f\in L}\left[A(f) \oplus A(f+e_i) = a'(x_i)\right] &\geq \Pr_{f\in L}\left[A(f) = f(a') \quad and \quad A(f+e_i) = (f \oplus e_i)(a')\right] \\ &\geq 1 - 2\cdot\delta/2 = 1 - \delta \end{aligned}$$

The check fails whenever i, f are such that $a'(x_i) \neq \sigma(x_i)$ and yet $A(f) \oplus A(f + e_i) = a'(x_i)$. Altogether this occurs with probability at least $(1-\delta)\delta \geq \delta/2$, and T' runs this test with probability $1/2$, so it rejects again with probability $\Omega(\delta)$ as claimed. ■

Consider the assignment $b|_Z$. For every random string that causes T' to reject (on input σ, A), the associated variable z_r is either assigned consistently with A, σ which means that its value immediately causes the associated constraint to reject; or it is inconsistent with A, σ. Each inconsistency will be detected with probability at least $1/3$. Thus at least $\frac{\Omega(\delta)}{3} = \Omega(\delta)$ fraction of the constraints reject. Hence $\textsc{unsat}_{a\cup b}(G) = \Omega(\delta) = \Omega(\mathrm{rdist}(\sigma, SAT(\psi)))$. ■

8 Short PCPs and Locally Testable Codes

In this section we describe how to construct extremely-short Probabilistically Checkable Proofs and Locally-Testable Codes (LTCs). Our starting point is the construction of Ben-Sasson and Sudan [8]. The case of short PCPs follows rather directly from our main theorem (Theorem 1.5) and is described first, in Subsection 8.2. The case of short LTCs is analogous, and is obtained similarly from a variant of the main theorem. This variant is an adaptation of our reduction between constraint graphs into a special kind of reduction called an assignment tester or a PCP of Proximity. We feel that this adaptation may be of independent interest, and it is described fully in Section 9. Assuming this adaptation, we describe our short LTCs in Subsection 8.3. Let us first begin with some definitions and notations.

8.1 Definitions and Notation

Given a system of constraints Φ, we denote its *unsat-value* by $\textsc{unsat}(\Phi)$: the minimum over all possible assignments for Φ's variables, of the fraction of unsatisfied constraints. This is a natural extension of the unsat-value of a constraint graph.

Definition 8.1 ($PCP_{s,c}[\log \ell, q]$) *We define the class of languages $PCP_{s,c}[\log_2(\ell(n)), q(n)]$, with parameters $s(n), c(n)$ and $\ell(n)$ and $q(n)$ as follows. A language L is in this class iff there is a reduction taking an instance x to a system of constraints $\Phi(x)$ such that, for $n = |x|$,*

- *$|\Phi(x)| \leq \ell(n)$; and each constraint $\varphi \in \Phi(x)$ accesses at most $q(n)$ variables.*
- *If $x \in L$ then $1 - \text{UNSAT}(\Phi(x)) \geq c(n)$*
- *If $x \notin L$ then $1 - \text{UNSAT}(\Phi(x)) \leq s(n)$*

Definition 8.2 (Locally Testable Codes) *A code $C \subset \Sigma^n$ is (q, δ, ε)-locally testable if there is a randomized algorithm A that is given oracle access to a string x, then (non-adaptively) reads at most q symbols from x, and decides whether to accept or reject such that*

- *For every $x \in C$, $Pr[A^x \ accepts] = 1$.*
- *For every string $y \in \Sigma^n$ such that $\text{rdist}(y, C) \geq \delta$, $\Pr[A^y \ rejects] \geq \varepsilon$.*

8.2 Short PCPs

Our main theorem in this section is,

Theorem 8.1 $SAT \in PCP_{\frac{1}{2},1}[\log_2(n \cdot \text{poly} \log n), O(1)]$.

We prove this theorem by relying on a recent result of Ben-Sasson and Sudan,

Theorem 8.2 ([8, Theorem 2.2]) *For any proper complexity function $t : \mathbb{N} \to \mathbb{N}$,*

$$NTIME(t(n)) \subseteq PCP_{\frac{1}{2},1}[\log(t(n)\text{poly} \log t(n)), \text{poly} \log t(n)].$$

From this result, we derive $SAT \in PCP_{1-\frac{1}{\text{poly}\log n},1}[\log_2(n \cdot \text{poly} \log n), O(1)]$. More precisely,

Lemma 8.3 *There exist constants $c_1, c_2 > 0$ and a polynomial-time reduction that transforms any SAT instance φ of size n into a constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ such that*

- *$size(G) \leq n(\log n)^{c_1}$ and $|\Sigma| = O(1)$.*
- *If φ is satisfiable, then $\text{UNSAT}(G) = 0$.*
- *If φ is not satisfiable, then $\text{UNSAT}(G) \geq \frac{1}{(\log n)^{c_2}}$.*

Before proving the lemma, let us see how it implies Theorem 8.1,

Proof of Theorem 8.1: Given a SAT instance of size n, we rely on Lemma 8.3 to reduce it to a constraint graph G whose size we denote by $m = n \cdot (\log n)^{c_1}$. Then, we apply the main theorem (Theorem 1.5) iteratively $k = c_2 \cdot \log \log m < 2c_2 \log \log n$ times. This results in a constraint-graph G' for which $\text{UNSAT}(G') \geq \min(2^k \cdot \text{UNSAT}(G) \, , \, \alpha) = \alpha$, and such that $size(G') = C^{c_2 \log \log m} \cdot m \leq n \cdot (\log n)^{c_1 + 2c_2 \log C} = n \cdot \text{poly} \log n$.

To get an error-probability of $\frac{1}{2}$ one can apply the expander-neighborhood sampler of [19] (see also [16, Section C.4]) for efficient amplification. ■

Proof of Lemma 8.3: Since $SAT \in NTIME(O(n))$, Theorem 8.2 yields some constants $a_1, a_2 > 0$ and a reduction from SAT to a system Ψ_0 of at most $m = n \cdot (\log n)^{a_1}$ constraints, each over at most $(\log n)^{a_2}$ Boolean variables such that satisfiable inputs go to satisfiable systems, and unsatisfiable inputs result in systems for which any assignment satisfies at most $\frac{1}{2}$ of the constraints. Our goal is to reduce the number of queries per constraint. Basically, this is done by introducing new variables over a large alphabet, which enables few queries in a naive way (which causes the rejection probability to deteriorate). Then, the alphabet size is reduced through composition.

Two-variable Constraints For each constraint in Ψ_0, let us introduce one new (big) variable. This variable will take values over alphabet $\Sigma = \{0,1\}^{(\log n)^{a_2}}$ that supposedly represent values to all of the original (small) variables queried in that constraint. The number of big variables is $m = n \cdot (\log n)^{a_1}$. Introduce $(\log n)^{a_2}$ new constraints per big variable: Each constraint will query the big variable and exactly one of the small variables queried by the corresponding constraint. The constraint will check that the value for the big variable satisfies the original constraint, and that it is consistent with the second (small) variable. Call this system Ψ and observe that $|\Psi| = n \cdot (\log n)^{a_1+a_2}$.

What is $\text{UNSAT}(\Psi)$? Given an assignment for the original variables it must cause at least $m/2$ (original) constraints to reject. Each big variable that corresponds to a rejecting constraint must now participate in at least one new rejecting constraint. Indeed, even if it is assigned a value that is accepting, it must differ from this assignment, so it will be inconsistent with at least one original (small) variable. Altogether, at least $\frac{m/2}{m \cdot (\log n)^{a_2}} \geq (\log n)^{-(a_2+1)}$ fraction of the constraints in Ψ must reject.

Composition We next apply composition to reduce the alphabet size from $\log|\Sigma| = \text{poly}\log n$ to $O(1)$. This is exactly as done in Lemma 1.8 except that we are somewhat more restricted in our choice of the assignment tester algorithm $\mathcal{P}$ (or equivalently: a PCP of Proximity), in that the output size of $\mathcal{P}$ must be polynomial in the input size. Observe that we only require that the size of the output is *polynomial* (and not quasi-linear) in the input size, so there is no circularity in our argument. Existence of such an algorithm $\mathcal{P}$ is an implicit consequence of the proof of the PCP Theorem of [3, 2], and was explicitly described in [7, 11].

Here is a brief summary of the construction of Lemma 1.8: We encode each variable via a linear dimension, linear distance error-correcting-code, treating the 'small' variable in each constraint as if its value lies in the large alphabet. We then run $\mathcal{P}$ on each constraint and let the new system Ψ' be the union of the output constraint systems.

Assuming that the rejection probability of $\mathcal{P}$ is $\varepsilon = \Omega(1)$, the soundness analysis shows that

$$\text{UNSAT}(\Psi') \geq \text{UNSAT}(\Psi) \cdot \varepsilon = \Omega((\log n)^{-(a_2+1)}) = \frac{1}{\text{poly}\log n}$$

where the middle equality holds since ε is a constant. Since the input size for $\mathcal{P}$ was the size of one constraint in Ψ, i.e., $\text{poly}\log n$, it follows that the size of the constraint system output by $\mathcal{P}$ is also $\text{poly}\log n$. This means that $|\Psi'| = |\Psi| \cdot \text{poly}\log n = n \cdot \text{poly}\log n$ ■

8.3 Short Locally Testable Codes

A similar construction to that of Theorem 8.1 can be used to obtain locally-testable codes with inverse poly-logarithmic rate (i.e., mapping k bits to $k \cdot \text{poly}\log k$ bits), that are testable with a constant number of queries.

The way we go about it is by relying on a variant of the main theorem (Theorem 1.5). Recall that the main theorem is a reduction from G to $G' = (\text{prep}(G)^t) \circ \mathcal{P}$. We will need a stronger kind of reduction, that is an assignment tester (also called a PCP of Proximity), as defined in Definition 2.2.

In the next section we will prove that the main amplification step (as in Theorem 1.5) can also work for assignment-testers. Formally,

Theorem 9.1 *There exists $t \in \mathbb{N}$ such that given an assignment-tester with constant-size alphabet Σ and rejection probability ϵ, one can construct an assignment-tester with the same alphabet and rejection* *probability at least* $\min(2\epsilon, 1/t)$*, such that the output size of the new reduction is bounded by at most a constant factor times the output size of the given reduction.*

Just as our main theorem (Theorem 1.5) could be combined with the construction of [8] yielding a short PCP, Theorem 9.1 can be combined with the construction of [8] to yield short PCPs of Proximity / assignment-tester reductions.

Corollary 8.4 *There exists an assignment-tester with constant size alphabet, and constant rejection probability* $\epsilon > 0$*, such that inputs of size* n *are transformed to outputs of size at most* $n \cdot \mathrm{poly}\log n$*.*

Proof: As in the proof of Theorem 8.1, we begin with a lemma that follows from the construction of [8],

Lemma 8.5 *There exist a polynomial-time assignment-tester with constant alphabet size and rejection probability* $\epsilon \geq \frac{1}{(\log n)^{O(1)}}$*, such that inputs of size* n *are transformed to outputs of size at most* $n \cdot \mathrm{poly}\log n$*.*

The difference between this lemma and Lemma 8.3 is that here we require the reduction to be an assignment-tester. This can be derived from the construction of [8], in a similar way to the proof of Lemma 8.3.

Let $\mathcal{A}_0$ be the assignment-tester from Lemma 8.5. Let $\mathcal{A}_i$ be the result of applying the transformation guaranteed in Theorem 9.1 on $\mathcal{A}_{i-1}$. For $i = O(\log\log n)$, the reduction $\mathcal{A}_i$ will have the required parameters. ■

Finally, we claim that Corollary 8.4 directly implies the existence of locally testable codes of rate $1/\mathrm{poly}\log n$.

Corollary 8.6 *For every* $\delta > 0$ *there exists an* $\varepsilon = \Omega(\delta) > 0$*, and an infinite family of codes* $\{C_N\}_N$ *with rate* $1/\mathrm{poly}\log N$*, such that* C_N *is* $(2, \delta, \varepsilon)$*-locally-testable.*

Proof: Assuming we have the assignment tester from Corollary 8.4, we apply the construction of [7, Construction 4.3]. We give a brief sketch of the construction. We construct C_N as follows. Fix $n \in \mathbb{N}$ and let $C'_n \subset \Sigma^n$ be an error correcting code with rate and distance $\Theta(n)$. Let Φ be a circuit over variables $X = \{x_1, \ldots, x_n\}$ that accepts iff the assignment for X is a codeword in C'_n. We can assume that $|\Phi| = O(n)$ (using, e.g., expander codes [33]). Run the reduction of Corollary 8.4 on Φ, and let G be the output constraint graph, $size(G) = n \cdot \mathrm{poly}\log n$. Let $Y = V \setminus X$ be the new variables added by the reduction, and denote $m = |Y|$, $m \leq n \cdot \mathrm{poly}\log n$. Let $\ell = \frac{2m}{\delta n}$, $N = n\ell + m$, and define a new code

$$C_N = \left\{ a^\ell b \in \Sigma^{n\ell+m} \;\middle|\; a \in C'_n, b \in \Sigma^m \text{ and } \mathrm{UNSAT}_\sigma(G) = 0 \text{ where } \sigma|_X = a \text{ and } \sigma|_Y = b \right\} \subset \Sigma^N .$$

where $a^\ell b$ denotes the concatenation of ℓ copies of a with b. Clearly, the rate of C_N is $1/\mathrm{poly}\log N$. We claim that C_N is $(2, \delta, \varepsilon)$-locally-testable. Here is the testing algorithm for a given word $w \in \Sigma^{n\ell+m}$. Denote the i-th bit of w by w_i.

1. Flip a random coin.
2. If heads, choose a random $i \in [n]$ and a random $j \in \{1, 2, \ldots, \ell - 1\}$, and accept iff $w_i = w_{i+j\cdot\ell}$
3. If tails, choose a random constraint in G. View $w[1, \ldots, n]$ as an assignment for X and $w[n\ell + 1, \ldots, n\ell + m]$ as an assignment for Y. Accept iff the constraint is satisfied by this assignment.

Clearly, every $w \in C_N$ passes the test with probability 1. If $\mathrm{rdist}(w', C_N) > \delta$, then for any codeword $\sigma = a^\ell b \in C_N$, since $m \le n\ell \cdot \frac{\delta}{2}$, the strings w' and σ must differ on $\delta n\ell/2$ of their first $n\ell$ bits. The reader may verify that the test rejects with probability at least $\Omega(\delta)$. ■

Remark 8.1 (Constant Relative Distance) *The codes above also have a constant relative distance. This follows almost immediately from the distance of C'_n, except for the following caveat. A problem would arise if for some assignment a for X that satisfies Φ there are two assignments b_1, b_2 for Y such that both $\mathrm{UNSAT}_{a\cup b_1}(G) = 0$ and $\mathrm{UNSAT}_{a\cup b_2}(G) = 0$. This would imply that $a^\ell b_1, a^\ell b_2 \in C_N$, and their distance can be quite small. However, this can be ruled out if every assignment a has only one assignment b such that $\mathrm{UNSAT}_{a\cup b}(G) = 0$. This can be ensured here, and therefore we conclude that the above does yield codes with constant relative distance.*

9 Adapting the Main Theorem for Assignment-Testers

In this section we show how to adapt the main amplification step (Theorem 1.5), that was described as a reduction between constraint graphs, to work within the more demanding framework of an assignment-tester. This gives an extension of our main theorem (and Theorem 1.2), to assignment-testers / PCPs of proximity.

Theorem 9.1 *There exists $t \in \mathbb{N}$ and $|\Sigma_0| > 1$ such that given an assignment-tester with constant-size alphabet Σ and rejection probability ϵ, one can construct an assignment-tester with alphabet Σ_0 and rejection probability at least $\min(2\epsilon, 1/t)$, such that the output size of the new reduction is bounded by at most C times the output size of the given reduction, and C depends only on $|\Sigma|$.*

Suppose we have a reduction taking Φ to G. We construct from G a new graph G' and prove that the reduction taking Φ to G and then to G' has the desired properties.

Let $H = (\mathrm{prep}(G))^t$ be the result of running the preprocessing step (Lemma 1.7) and then raising the resulting constraint graph to the power t. What are the variables of H? Going from G to $\mathrm{prep}(G)$ each variable $v \in V$ is split into many copies, and we denote the set of copies of v by $[v]$. Next, going from $\mathrm{prep}(G)$ to $H = (\mathrm{prep}(G))^t$, the variables of H are identical to those of $\mathrm{prep}(G)$, but take values from a larger alphabet. So denoting the variables of H by V_H, we have $V_H = \cup_{v\in V}[v]$. Syntactically, V_H is disjoint from V, although the values for V_H are supposed to "encode" values for V. Indeed, an assignment $\sigma : V \to \Sigma$ can be mapped to an assignment $\sigma_2 : V_H \to \Sigma^{d^{t/2}}$ that "encodes" it, by the following two steps.

1. First define a mapping $\sigma \mapsto \sigma_1$, where the assignment $\sigma_1 : V_H \to \Sigma$ for $\mathrm{prep}(G)$ is defined by assigning all copies of v the same value as $\sigma(v)$:

$$\forall v \in V\ w \in [v], \quad \sigma_1(w) \overset{\triangle}{=} \sigma(v). \tag{13}$$

   Let us name this mapping m_1. Observe also that given any assignment for $\mathrm{prep}(G)$, $\sigma' : V_H \to \Sigma$, it can be "decoded" into an assignment for G according to "popularity" as follows. Simply set $\sigma = m_1^{-1}(\sigma')$ to be an assignment $\sigma : V \to \Sigma$ for which $m_1(\sigma)$ is closest[5] in Hamming distance to σ'.

[5] Breaking ties arbitrarily.

2. Next, define a mapping $\sigma_1 \mapsto \sigma_2$, where the assignment $\sigma_2 : V_H \to \Sigma^{d^{t/2}}$ for H is defined by assigning each vertex w a vector consisting of the σ_1-values of all vertices reachable from w by a $t/2$-step walk

$$\forall w \in V_H, \quad \sigma_2(w)_v \overset{\triangle}{=} \sigma_1(v) \text{ for all } v \text{ reachable from } w \text{ by a } t/2\text{-step walk in } G\,. \tag{14}$$

Let us name this mapping m_2, and again, given any assignment $\sigma' : V_H \to \Sigma^{d^{t/2}}$ for $(\mathrm{prep}(G))^t$ it can be "decoded" into an assignment for $\mathrm{prep}(G)$ as follows. Simply set $\sigma = m_2^{-1}(\sigma')$ to be the assignment defined by

$$\sigma(v) \overset{\triangle}{=} \max\arg_{a\in\Sigma} \left\{ \Pr[\text{A random } \lceil t/2 \rceil\text{-step walk from } v \text{ reaches a vertex } w \text{ for which } \sigma'(w)_v = a] \right\}.$$

This coincides with the "most popular opinion" assignment as defined in Equation (4) of Section 6.

Going back to our reduction, we recall that in order for our reduction to be an assignment-tester, our output constraint graph must have the variables X of Φ contained in its set of variables. Then, we must also verify that the completeness and soundness conditions (that refer to X) hold.

The Graph H' We next transform H to H' so as to include X among the variables of H'. The vertices of H' will be $V_H \cup X$. The constraints of H' will include all of the constraints of H, and also additional constraints that will check that the assignment for V_H is a correct encoding, according to the mapping $m_2 \circ m_1$ which maps σ to σ_2 (via σ_1), of the assignment for X.

We describe the constraints between X and V_H by the following randomized procedure. Let $A : V_H \to \Sigma^{d^{t/2}}$ and let $a : X \to \{0,1\}$.

1. Select $x \in_R X$.
2. Select $z \in_R [x]$ (recall that $[x]$ is the set of vertices in $\mathrm{prep}(G)$ that are copies of x).
3. Take a $t/2$-step random walk in $\mathrm{prep}(G)$ starting from z, and let w be the endpoint of the walk. Accept if and only if $A(w)_z = a(x)$.

For every possible random choice of the test, we will place (an edge and) a constraint between w and x, that accepts iff the test accepts. We will reweigh the constraints (by duplication) so that the weight of the comparison constraints defined by the random procedure is half of the total weight of the edges. This completes the description of H'. Observe that the size of H' is at most a constant times the size of G, because $\mathrm{prep}(G)$ is d-regular for $d = O(1)$, so every vertex $w \in V_H$ participates in exactly $d^{t/2} = O(1)$ new comparison constraints. The next lemma states that the reduction from Φ to H' is an assignment-tester with large alphabet, and rejection probability $\Theta(\sqrt{t}) \cdot \epsilon$.

Lemma 9.2 *Assume $\varepsilon < 1/t$, and fix $a : X \to \{0,1\}$.*

- *If $a \in \mathrm{SAT}(\Phi)$, there exists $b : V_H \to \Sigma^{d^{t/2}}$ such that $\mathrm{UNSAT}_{a\cup b}(H') = 0$.*
- *If $\delta = \mathrm{rdist}(a, \mathrm{SAT}(\Phi)) > 0$, then for every $b : V_H \to \Sigma^{d^{t/2}}$, $\mathrm{UNSAT}_{a\cup b}(H') > \delta{\cdot}\min(\frac{1}{16}, (\beta_1\beta_2\sqrt{t}/2)\varepsilon)$.*

We prove this lemma shortly below. First, note that the constraint graph H' is almost what we need, except that it is defined over the alphabet $\Sigma^{d^{t/2}}$, rather than over Σ. Let us now proceed to construct the final graph G'.

The Graph G' To reduce the alphabet of H', we use composition. I.e., we assume that we have at our disposal an assignment-tester $\mathcal{P}$ such that its rejection probability is some constant $\varepsilon_0 > 0$, and its alphabet is Σ_0. We make no requirements about the length of the output of $\mathcal{P}$, because we will only run it on inputs of bounded size. For example, we can use the construction given in Section 7.

Now, the Composition Theorem of assignment-testers, [11, Theorem 3.7], states that given any two such reductions, their composition is well defined (it is essentially described in the proof of Lemma 1.8 herein) and is itself an assignment-tester, with the following parameters:

- The *alphabet size* is that of the inner reduction $\mathcal{P}$, thus the constraints in G' are over alphabet Σ_0, as desired.

- The *output size* is the product of the output sizes of the two reductions. In our case, this means that the output size of the reduction $\Phi \Rightarrow H'$ is multiplied by a *constant* factor that is the maximum size of the output of $\mathcal{P}$ when run on a constraint of H'.

- The *rejection probability* is the product of the rejection probabilities of the two reductions. Thus, denoting the rejection probability of $\mathcal{P}$ by ε_0, it is ε_0 times the rejection probability of the reduction $\Phi \Rightarrow H'$. Since this value was $\min(\frac{1}{16}, (\beta_1\beta_2\sqrt{t}/2)\varepsilon)$, by choosing t large enough, even after multiplying by ε_0 it is still larger than 2ε for all small enough ε.

This completes the description of the transformation taking Φ to G'. It remains to prove Lemma 9.2.
Proof: (of Lemma 9.2) In this proof, there are four constraint graphs that we keep in mind

$$G \quad\Rightarrow\quad \text{prep}(G) \quad\Rightarrow\quad H = (\text{prep}(G))^t \quad\Rightarrow\quad H' .$$

Recall that we encode assignments for G via m_1, obtaining assignments for $\text{prep}(G)$. These are encoded via m_2, giving assignments for H. We can also go in the opposite direction where an assignment for H can be decoded into an assignment for $\text{prep}(G)$ via m_2^{-1}, and similarly an assignment for $\text{prep}(G)$ can be decoded via m_1^{-1} into as assignment for G.

- Suppose $a \in \text{SAT}(\Phi)$. Then, by assumption on the reduction from Φ to G, there is an assignment $b : V \to \Sigma$ such that $\sigma = a \cup b$ satisfies all constraints in G. The assignment σ is mapped, via m_1 to an assignment σ_1 for $\text{prep}(G)$, and σ_1 in turn is mapped via m_2 into an assignment for H: $\sigma_2 : V_H \to \Sigma^{d^{t/2}}$. By the completeness of the preprocessing and the powering, σ_2 will satisfy all constraints in H. It is easy to verify that σ_2 will also satisfy (together with a) all of the new comparison constraints, so $\text{UNSAT}_{a\cup\sigma_2}(H') = 0$

- Assume now $\text{rdist}(a, \text{SAT}(\Phi)) = \delta > 0$. Fix some assignment $b : V_H \to \Sigma^{d^{t/2}}$. We will show that the assignment $a \cup b$ violates many of the constraints. The idea is to first "decode" b (through m_2^{-1} and then m_1^{-1}) thereby getting an assignment $b_0 : V \to \Sigma$. Then, we show that either b_0 is close to the assignment a, in which case it is far from $\text{SAT}(\Phi)$, so by amplification b must violate many of the constraints in H. Otherwise, if b_0 is far from a, then many (a constant fraction!) of the comparison constraints will fail.

  So let $b_1 = m_2^{-1}(b)$ be an assignment for the vertices of $\text{prep}(G)$, and let $b_0 = m_1^{-1}(b_1)$ be an assignment for the vertices of G, where notation m_1^{-1}, m_2^{-1} was defined in steps 1 and 2 of the construction. There are two cases.

  – If $\text{rdist}(b_0|_X, a) \le \delta/2$ then $\text{rdist}(b_0|_X, \text{SAT}(\Phi)) \ge \delta/2$ by the triangle inequality. Since the reduction from Φ to G is an assignment-tester with rejection probability ε, this means that no matter what $b_0|_{(V\setminus X)}$ is, $\text{UNSAT}_{b_0}(G) \geq \varepsilon\delta/2$. Now we claim that b_1 must also be violating a similar fraction of the constraints of $\text{prep}(G)$:

$$\text{UNSAT}_{b_1}(\text{prep}(G)) \geq \varepsilon\delta/2 \cdot \beta_1. \tag{15}$$

Indeed, recall Corollary 4.3 that asserts that for every G and for every assignment σ' for $\text{prep}(G)$, the fraction of constraints of $\text{prep}(G)$ violated by σ' is proportional to the fraction of constraints of G violated by $m_1^{-1}(\sigma')$. Plugging in b_1 for σ', and since $b_0 = m_1^{-1}(b_1)$, this implies (15).

Next, we claim that b must be violating an even larger fraction of $H = (\text{prep}(G))^t$ than $\text{UNSAT}_{b_1}(\text{prep}(G))$:

$$\text{UNSAT}_b((\text{prep}(G))^t) \geq \beta_2\sqrt{t} \cdot \min(\frac{1}{t}, \text{UNSAT}_{b_1}(\text{prep}(G)))\,. \tag{16}$$

Indeed, this follows precisely from Lemma 6.1 that states that for every G and every assignment $\vec{\sigma}$ for G^t, the fraction of constraints of G^t violated by $\vec{\sigma}$ is larger than the fraction of constraints of G violated by the "popular opinion" assignment, by factor $\Omega(\sqrt{t})$. Observe that indeed $m_2^{-1}(\vec{\sigma})$ is the "popular opinion" assignment. Plugging in b for $\vec{\sigma}$, and since $b_1 = m_2^{-1}(b)$, this implies (16). Combining (15) and (16), and observing that by assumption $1/t$ is clearly larger than $\varepsilon > \text{UNSAT}_{b_1}(\text{prep}(G))$,

$$\text{UNSAT}_b(H) \geq \varepsilon\delta/2 \cdot \beta_1 \cdot \beta_2\sqrt{t}\,.$$

Since the constraints of H are half of the constraints of H', we have

$$\text{UNSAT}_{a\cup b}(H') \geq \frac{1}{2}\text{UNSAT}_b(H) \geq \varepsilon\delta/4 \cdot \beta_1 \cdot \beta_2\sqrt{t}.$$

– If $\text{rdist}(b_0|_X, a) > \delta/2$, then we will show that $\delta/8$ fraction of the comparison constraints reject. Indeed, with probability at least $\delta/2$ the randomized test selects, in step 1, a variable $x \in X$ for which $b_0(x) \neq a(x)$. Conditioned on that, consider the probability that in step 2 a variable $z \in [x]$ is selected such that $b_1(z) \neq a(x)$. Since $b_0(x)$ is, by definition, a most popular value among values assigned by b_1 to the copies of x, and since by conditioning $a(x) \neq b_0(x)$, this probability is at least $1/2$. Conditioned on both previous events occurring, step 3 selects a vertex w for which $b(w)_z \neq a(x)$, with probability at least $1/2$ (for similar reasoning). Altogether, with probability at least $\frac{\delta}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \delta/8$ the test rejects. This means that at least $\delta/16$ of the total number of tests reject, i.e., $\text{UNSAT}_{a\cup b}(H') \geq \delta/16$.

We have proven that for $\delta = \text{rdist}(a, \text{SAT}(\Phi))$, and for every assignment b, the rejection probability $\text{UNSAT}_{a\cup b}(H')$ is either at least $\delta \cdot \frac{1}{16}$ or at least $\delta \cdot (\beta_1\beta_2\sqrt{t}/2 \cdot \varepsilon)$.

This completes the proof. ■

Theorem 9.1 also gives an immediate combinatorial construction of assignment-testers or PCPPs in the same way that the main theorem (Theorem 1.5) was used to derive the PCP Theorem (Theorem 1.2).

Corollary 9.3 *There is an assignment-tester, with constant alphabet, constant rejection probability, and polynomial output length.*

Proof: Given a circuit Φ it is easy to construct a constraint graph G_0 such that the reduction $\Phi \mapsto G_0$ is an assignment-tester with rejection probability $1/|G_0|$. Let us name this (trivial) assignment tester $\mathcal{P}_0$. Let us denote the rejection probability of an assignment tester $\mathcal{P}$ by $\varepsilon(\mathcal{P})$. We can now construct $\mathcal{P}_i$ inductively for every $i \geq 1$. Indeed for every $i \geq 0$ let us construct $\mathcal{P}_{i+1}$ by applying the transformation guaranteed in Theorem 9.1 to $\mathcal{P}_i$. The theorem asserts that the assignment-tester $\mathcal{P}_{i+1}$ maps Φ to G_{i+1} such that

1. The alphabet of G_{i+1} is Σ_0.
2. $\varepsilon(\mathcal{P}_{i+1}) \geq \min(\frac{1}{t}, 2\varepsilon(\mathcal{P}_i))$, where t is a global constant.
3. The running time of $\mathcal{P}_{i+1}$ is at most a constant C times the running time of $\mathcal{P}_i$.

We now prove that for $k = \log_2 n$, where $n = size(G_0)$, $\mathcal{P}_k$ is the desired assignment tester. Clearly the alphabet is Σ_0. It is also easy to see by induction that the running time of $\mathcal{P}_k$ is at most $C^k = \mathrm{poly}(n)$ times the running time of $\mathcal{P}_0$. Since $\mathcal{P}_0$ runs in polynomial-time, altogether $\mathcal{P}_k$ is a polynomial-time transformation.

It remains to show that $\varepsilon(\mathcal{P}_k) \geq 1/t$. Indeed, if for some $i^* < k$, $\varepsilon(\mathcal{P}_k) \geq 1/t$ then for every $i > i^*$ $\varepsilon(\mathcal{P}_i) \geq 1/t$ and in particular this holds for $\mathcal{P}_k$. Otherwise, it follows by induction from item 2 above that $\varepsilon(\mathcal{P}_k) \geq 2^k \varepsilon(\mathcal{P}_0) > 1/t$. ■

10 Graph Powering and Parallel-Repetition

The celebrated parallel repetition theorem of Raz [29] gives a different method of amplification. Given a system $\mathcal{C}$ of constraints, a new system $\mathcal{C}_{\shortparallel}^{\ell}$ is constructed by taking new variables corresponding to ℓ-tuples of the old variables, and new constraints corresponding to ℓ-tuples of the old constraints (we ignore the issue of bipartiteness in this discussion). The alphabet grows from Σ to Σ^{ℓ}. The theorem asserts that given a system of constraints $\mathcal{C}$ with $\text{UNSAT}(\mathcal{C}) = \alpha$, the ℓ-parallel-repetition system, $\mathcal{C}_{\shortparallel}^{\ell}$, will have $\text{UNSAT}(\mathcal{C}_{\shortparallel}^{\ell}) \geq 1 - (1-\alpha)^{\Theta(\ell)}$.

Our graph powering construction can be viewed as setting $\ell = d^{t/2}$ (where the graph underlying the constraints is d-regular) and taking a small and carefully chosen subset of the ℓ-tuples of variables and of the ℓ-tuples of constraints. Viewed this way, the graph powering construction is a 'derandomization' of the parallel-repetition theorem. We recall that Feige and Kilian proved that no generic derandomization of the parallel-repetition theorem is possible [13]. Their result focuses on a range of parameters that does not apply to our setting. This raises questions about the limits of such constructions in a wider range of parameters.

Let us conclude by mentioning a specific setting of the parameters which is of particular interest. When the unsat value of a constraint system is some fixed constant, then applying the parallel repetition transformation results in a new system whose unsat value approaches 1 as ℓ increases. This feature is very useful in inapproximability reductions. On the other hand, our amplification stops to make any progress for constant $\alpha > 0$, as is demonstrated in the instructive example of Bogdanov [10].

Of course, the main advantage of our 'derandomized' construction is that the new system size is only linear in the old system size. This feature is essential for our inductive proof of the PCP theorem.

Acknowledgements

I am thankful to Omer Reingold and Luca Trevisan for many discussions, especially ones about combinatorial analyses of graph powering, which were the direct trigger for the amplification lemma. I would like to thank Jaikumar Radhakrishnan for very helpful comments on an earlier version of this manuscript, and the anonymous referees whose excellent comments greatly improved the quality of the manuscript. I would also like to thank David Arnon, Miki Ben-Or, Ehud Friedgut, Oded Goldreich, and Alex Samorodnitsky for helpful comments.

References

[1] S. Arora. *Probabilistic checking of proofs and the hardness of approximation problems*. PhD thesis, U.C. Berkeley, 1994. Available via anonymous ftp as Princeton TR94-476.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[4] L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 421–429, 1985.

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

[6] M. Ben-or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi prover interactive proofs: How to remove intractability assumptions. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 113–131, 1988.

[7] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proc. 36th ACM Symp. on Theory of Computing*, 2004.

[8] E. Ben-Sasson and M. Sudan. Robust locally testable codes and products of codes. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*, 2004.

[9] E. Ben-Sasson, M. Sudan, S. P. Vadhan, and A. Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proc. 35th ACM Symp. on Theory of Computing*, pages 612–621, 2003.

[10] A. Bogdanov. Gap amplification fails below 1/2. Comment on ECCC TR05-046, can be found at `http://eccc.uni-trier.de/eccc-reports/2005/TR05-046/commt01.pdf`, 2005.

[11] I. Dinur and O. Reingold. Assignment testers: Towards combinatorial proofs of the PCP theorem. In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS)*, 2004.

[12] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Journal of the ACM*, 43(2):268–292, 1996.

[13] U. Feige and J. Kilian. Impossibility results for recycling random bits in two-prover proof systems. In *Proc. 27th ACM Symp. on Theory of Computing*, pages 457–468, 1995.

[14] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[15] E. Friedgut, G. Kalai, and A. Naor. Boolean functions whose fourier transform is concentrated on the first two levels. *Adv. in Applied Math.*, 29:427–437, 2002.

[16] O. Goldreich. A sample of samplers a computational perspective on sampling. Electronic Colloquium on Computational Complexity TR97-020, 1997.

[17] O. Goldreich and S. Safra. A combinatorial consistency lemma with application to proving the PCP theorem. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*. LNCS, 1997.

[18] O. Goldreich and M. Sudan. Locally testable codes and PCPs of almost-linear length. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science*, pages 13–22, 2002.

[19] O. Goldreich and A. Wigderson. Tiny families of functions with random properties: A qualitysize tradeoff for hashing. *Journal of Random structures and Algorithms*, 11(4):315–343, 1997.

[20] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proofs. *SIAM Journal on Computing*, 18:186–208, 1989.

[21] P. Harsha and M. Sudan. Small PCPs with low query complexity. In *STACS*, pages 327–338, 2001.

[22] J. Håstad. Some optimal inapproximability results. *Journal of ACM*, 48:798–859, 2001.

[23] N. Linial and A. Wigderson. Expander graphs and their applications. Lecture notes of a course: http://www.math.ias.edu/ boaz/ExpanderCourse/, 2003.

[24] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, October 1992.

[25] R. O'Donnell and V. Guruswami. Lecture notes from a course on: the PCP theorem and hardness of approximation. 2005.

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

[27] A. Polishchuk and D. Spielman. Nearly linear size holographic proofs. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 194–203, 1994.

[28] J. Radhakrishnan. Private communication. 2005.

[29] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998.

[30] O. Reingold. Undirected st-connectivity in log-space. In *Proc. 37th ACM Symp. on Theory of Computing*, 2005.

[31] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *Annals of Mathematics*, 155(1):157–187, 2002.

[32] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992. Prelim. version in 1990 FOCS, pages 11–15.

[33] M. Sipser and D. A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1710–1722, 1996. Codes and complexity.

A A Lemma about similar binomial distributions

For $n \in \mathbb{N}$ and $p \in (0,1)$ let $B_{n,p}$ denote a binomially distributed random variable, i.e., $\Pr[B_{n,p} = k] = \binom{n}{k} p^k (1-p)^{n-k}$. The following lemma asserts that if n, m are close, then the distributions of $B_{n,p}$ and $B_{m,p}$ are close.

Lemma 6.4 *For every $p \in (0,1)$ and $c > 0$ there exists some $0 < \tau < 1$ and n_0 such that if $n_0 < n - \sqrt{n} \le m < n + \sqrt{n}$, then*

$$\forall k \in \mathbb{N}, |k - pn| \le c\sqrt{n}, \qquad \tau \le \frac{\Pr[B_{n,p} = k]}{\Pr[B_{m,p} = k]} \le \frac{1}{\tau}.$$

Proof: Assume first that $m \le n$. We write $n = m + r$ for some $0 \le r \le \sqrt{n}$ and use the identity $\binom{m+1}{k} = \frac{m+1}{m+1-k}\binom{m}{k}$,

$$\begin{aligned}
\Pr[B_{n,p} = k] &= \binom{m+r}{k} p^k (1-p)^{m+r-k} \\
&= \frac{m+1}{m+1-k} \cdot \frac{m+2}{m+2-k} \cdots \frac{m+r}{m+r-k} \binom{m}{k} \cdot p^k (1-p)^{m-k} (1-p)^r \\
&= X \cdot p^k (1-p)^{m-k} \binom{m}{k} = X \cdot \Pr[B_{m,p} = k]
\end{aligned}$$

where $X = (1-p)^r \frac{m+1}{m+1-k} \cdot \frac{m+2}{m+2-k} \cdots \frac{m+r}{m+r-k}$ is bounded as follows. Let $X_a = \frac{m+a}{m+a-k}$. For all $a \le r \le \sqrt{n}$ we have $X_a = \frac{1}{1 - \frac{k}{m+a}}$ and clearly

$$\frac{1}{1 - \frac{k}{m}} \ge \frac{1}{1 - \frac{k}{m+a}} \ge \frac{1}{1 - \frac{k}{n}}.$$

We will choose n_0 large enough to make all of the expressions below strictly positive. Since $k \ge pn - c\sqrt{n}$,

$$X_a \ge \frac{1}{1 - \frac{k}{n}} \ge \frac{1}{1 - p + \frac{c}{\sqrt{n}}} = \frac{1}{1-p} \cdot \frac{1}{1 + \frac{c}{1-p}\frac{1}{\sqrt{n}}}.$$

Now, $X = (1-p)^r \cdot \prod_{a=1}^{r} X_a \ge (1 + \frac{c}{1-p}\frac{1}{\sqrt{n}})^{-r} =: \tau_1$.

Similarly, since $n - \sqrt{n} < m$ and $pn - c\sqrt{n} \le k$ it follows that $1 - \frac{k}{m} \ge 1 - p - \frac{c+1}{\sqrt{n}}$. So

$$X_a \le \frac{1}{1 - \frac{k}{m}} \le \frac{1}{1 - p - \frac{c+1}{\sqrt{n}}} = \frac{1}{1-p} \cdot \frac{1}{1 - \frac{c+1}{(1-p)(\sqrt{n})}},$$

and we have $X = (1-p)^r \cdot \prod_{a=1}^{r} X_a \le (1 - \frac{c+1}{(1-p)(\sqrt{n}-1)})^{-r} =: \tau_2$. Clearly since $r \le \sqrt{n}$ both τ_1 and τ_2 can be bounded by constants (independent of n), and we take $\tau = \min(\tau_1, \tau_2^{-1})$.

Finally, if $n < m$, then since clearly $m - \sqrt{m} < n < m$ we can deduce the result from applying the lemma with the roles of m and n reversed, the same p, and the constant $c' = c + 1$. ■

B The Long Code Test

In this section we prove Theorem 7.1. Let us identify $\{0,1\}^s$ with $[n]$ (where $n = 2^s$) in an arbitrary way. We consider Boolean functions $\psi : [n] \to \{1, -1\}$ by identifying -1 with true (so $a \in [n]$ is said to satisfy ψ iff $\psi(a) = -1$). Recall that $L = \{f : [n] \to \{-1,1\}\}$ is the set of Boolean functions on $[n]$. We restate Theorem 7.1 with this modified notation.

Theorem 7.1 *There exists a* Long-Code Test T *which is a randomized algorithm that has input a function* $\psi : [n] \to \{1,-1\}$, *and also oracle access to a folded string* $A : L \to \{1,-1\}$. T *reads the input* ψ *and tosses some random coins. Based on these it computes a three-bit predicate* $w : \{0,1\}^3 \to \{\mathsf{true}, \mathsf{false}\}$ *and three locations* $f_1, f_2, f_3 \in L$ *in which it queries the string* A. *It then outputs* $w(A(f_1), A(f_2), A(f_3))$. *Denote an execution of* T *with access to input* ψ *and folded string* A *by* $T^A(\psi)$. *Then the following hold,*

- *(Perfect completeness:) If* $a \in [n]$ *such that* $\psi(a) = -1$, *then* $\Pr[T^{A_a}(\psi) = \mathsf{true}] = 1$.
- *(Strong[6] soundness:) For every* $\delta \in [0,1]$, *if* $A : L \to \{1,-1\}$ *is folded and at least* δ*-far from* A_a *for all* a *for which* $\psi(a) = -1$, *then* $\Pr[T^A(\psi) = \mathsf{false}] \geq \Omega(\delta)$.

Our proof is basically a reworking of a test of Håstad [22], into our easier setting:

Standard Definitions. We identify $L = \{f : [n] \to \{-1,1\}\}$ with the Boolean hypercube $\{1,-1\}^n$, and use letters f, g for points in the hypercube. We use letters A, B or χ to denote functions whose domain is the hypercube[7]. For $\alpha \subset [n]$, define

$$\chi_\alpha : \{-1,1\}^n \to \{-1,1\}, \qquad \chi_\alpha(f) \stackrel{\triangle}{=} \prod_{i \in \alpha} f(i)\,.$$

The characters $\{\chi_\alpha\}_{\alpha \subseteq [n]}$ form an orthonormal basis for the space of functions $\{A : \{-1,1\}^n \to \mathbb{R}\}$, where inner product is defined by $\langle A, B\rangle = \mathbb{E}_f\left[A(f)B(f)\right] = 2^{-n}\sum_f A(f)B(f)$. It follows that any function $A : \{-1,1\}^n \to \{-1,1\}$ can be written as $A = \sum_\alpha \hat{A}_\alpha \chi_\alpha$, where $\hat{A}_\alpha = \langle A, \chi_\alpha\rangle$. We also have Parseval's identity, $\sum_\alpha |\hat{A}_\alpha|^2 = \langle A, A\rangle = 1$.

The Test. Let $\psi : [n] \to \{-1,1\}$ be some predicate and fix $\tau = \frac{1}{100}$. Let $A : \{-1,1\}^n \to \{-1,1\}$ be a folded string, i.e., for all $f \in \{-1,1\}^n$ $A(-f) = -A(f)$ and also $A(f) = A(f \wedge \psi)$ where $f \wedge \psi$ is defined by

$$\forall a \in [n], \qquad (f \wedge \psi)(a) = \begin{cases} -1 & f(a) = -1 \text{ and } \psi(a) = -1 \\ \\ 1 & \text{otherwise} \end{cases}.$$

A function $A : \{1,-1\}^n \to \{1,-1\}$ is the legal encoding of the value $a \in [n]$ iff $A(f) = f(a)$ for all $f \in L$. The following procedure tests whether A is close to a legal encoding of some value $a \in [n]$ that satisfies ψ.

1. Select $f, g \in L$ uniformly at random.
2. Set $h = g\mu$ where $\mu \in L$ is selected by doing the following independently for every $y \in [n]$. If $f(y) = 1$ set $\mu(y) = -1$. If $f(y) = -1$ set

$$\mu(y) = \begin{cases} 1 & \text{w. prob. } 1-\tau \\ \\ -1 & \text{w. prob. } \tau \end{cases}.$$

[6]We refer to 'strong' soundness as opposed to regular soundness, since due to the stronger property of having the rejection probability proportional to the distance from a "good" string.

[7]We consider here functions whose domain is an arbitrary set of size n, and wlog we take the set $[n]$. In the application this set is usually some $\{0,1\}^s$ but we can safely ignore this structure, and forget that $n = 2^s$.

3. Accept unless $A(g) = A(f) = A(h) = 1$.

It is clear that the test behaves according to the description in Theorem 7.1. It remains to prove completeness and soundness.

Proposition B.1 (Completeness) *If $a \in [n]$ such that $\psi(a) = -1$, then $\Pr[T^{A_a}(\psi) = \mathsf{true}] = 1$.*

Proof: It is easy to check completeness: We fix some $a \in [n]$ for which $\psi(a) = -1$ and assign for all f, $A(f) = f(a)$. Clearly A is folded. Also, if $A(f) = f(a) = -1$ then the test accepts. Finally, if $A(f) = f(a) = 1$ then $A(h) = h(a) = -g(a) = -A(g) \neq A(g)$, and again the test accepts. ■

Proposition B.2 (Soundness) *There exists a constant $c > 0$ such that for every $\delta \in [0, 1]$, if $A : L \to \{1, -1\}$ is folded and at least δ-far from A_a for all a for which $\psi(a) = -1$, then $\Pr[T^A(\psi) = \mathsf{false}] \geq c \cdot \delta$.*

Proof: Let us fix $\delta \in (0, 1]$ and assume that A is δ-far from every A_a for $a \in [n]$ that satisfies ψ. Denote the rejection probability of the test by $\Pr[T^A(\psi) = \mathsf{false}] = \varepsilon$. The proof of soundness will be based on the following proposition.

Proposition B.3 *There exists a constant $C > 0$ such that if T rejects with probability ε then*

$$\sum_{|\alpha|>1} \left|\hat{A}_\alpha\right|^2 \leq C\varepsilon.$$

We defer the proof of the proposition to later. We will need the following result,

Theorem B.4 ([15]) *There is a global constant $C' > 0$ (independent of n) such that the following holds. Let $\rho > 0$ and let $A : \{1, -1\}^n \to \{1, -1\}$ be a Boolean function for which $\sum_{\alpha, |\alpha|>1} |\hat{A}_\alpha|^2 < \rho$. Then either $|\hat{A}_\phi|^2 \geq 1 - C'\rho$, or for some $i \in [n]$, $|\hat{A}_{\{i\}}|^2 \geq 1 - C'\rho$.*

It is well-known that since A is folded $\hat{A}_\alpha = 0$ whenever (i) $|\alpha|$ is even, or (ii) $\exists i \in \alpha$ for which $\psi(i) = 1$ (recall that 1 corresponds to false). The reason is that we can partition $\{1, -1\}^n$ into pairs f, f' such that

$$\hat{A}_\alpha = 2^{-n} \sum_f A(f)\chi_\alpha(f) = 2^{-n} \cdot \frac{1}{2} \sum_f (A(f)\chi_\alpha(f) + A(f')\chi_\alpha(f')) = 2^{-n-1} \sum_f 0 = 0\,.$$

In (i) let $f' = -f$, so $\chi_\alpha(f) = \chi_\alpha(f')$ but $A(f) = -A(f')$. In (ii) let $f' = f + e_i$ where i is an index for which $\psi(i) = 1$; so $\chi_\alpha(f) = -\chi_\alpha(f')$ but $A(f) = A(f')$. We can thus deduce from Theorem B.4 that there is some $i \in [n]$ for which $\psi(i) = -1$ and $|\hat{A}_{\{i\}}|^2 \geq 1 - C'C\varepsilon$.

This means that one of the following holds,

1. $\hat{A}_{\{i\}} \geq \sqrt{1 - C'C\varepsilon} \geq 1 - C'C\varepsilon$, or
2. $-\hat{A}_{\{i\}} \geq \sqrt{1 - C'C\varepsilon} \geq 1 - C'C\varepsilon$.

Since by definition $\hat{A}_\alpha = \mathbb{E}_f [A(f)\chi_\alpha(f)] = 1 - 2\mathrm{rdist}(A, \chi_\alpha)$, it follows that $\mathrm{rdist}(A, \chi_{\{i\}}) = \frac{1-\hat{A}_{\{i\}}}{2}$. If the first item holds, then clearly $\mathrm{rdist}(A, \chi_{\{i\}}) \leq CC'\varepsilon/2$ and we are done by choosing $c \leq 2/CC'$. Suppose the second item holds, we will show that ε is larger than some absolute constant, and by choosing c smaller than that constant we will be done. Imagine first that $-\hat{A}_{\{i\}} = 1$, i.e., $A = -\chi_{\{i\}}$. Then the probability (over the choice of f, g, and h) that $f(i) = -1$ and $g(i) = -1$ and

$h(i) = -1$ is at least $\frac{1}{4} \cdot (1-\tau)$, and in this case we have $A(f) = A(g) = A(h) = 1$ and the test rejects (so $\varepsilon \geq (1-\tau)/4 > 1/8$). This probability can go down by at most $3\mathrm{rdist}(A, -\chi_{\{i\}})$ (which is an upper bound on the probability that at least one of f, g, h is a point of disagreement between A and $-\chi_{\{i\}}$). We get

$$\varepsilon \geq \frac{1-\tau}{4} - 3\mathrm{rdist}(A, -\chi_{\{i\}}) > \frac{1}{8} - 3CC'\varepsilon$$

rearranging we get $\varepsilon > \frac{1}{8(1+3CC')}$.

Choosing $c = \min(\frac{2}{CC'}, \frac{1}{8(1+3CC')})$, we have proven that A is δ-far from every $\chi_{\{i\}}$ for a value of i that satisfies ψ, then the test rejects with probability at least $c\delta$. ■

It remains to prove the proposition.

Proof of Proposition B.3: Let us arithmetize the acceptance probability as follows

$$1 - \varepsilon = \Pr[\text{Test accepts}] = \mathbb{E}_{f,g,h}\left[1 - \frac{(1+A(f))(1+A(g))(1+A(h))}{8}\right] =$$

and note that since the pairs (f, g) and (f, h) are pairs of random independent functions, and since $\mathbb{E}[A] = \hat{A}_\phi = 0$ due to A being folded, this equals,

$$= \frac{7}{8} - \frac{1}{8}\mathbb{E}_{g,h}\left[A(g)A(h)\right] - \frac{1}{8}\mathbb{E}_{f,g,h}\left[A(f)A(g)A(h)\right].$$

Using the Fourier expansion $A(g) = \sum_\alpha \hat{A}_\alpha \chi_\alpha(g)$ the first expectation can be written as

$$\mathbb{E}_{g,h}\left[\sum_{\alpha,\beta\subseteq[n]} \hat{A}_\alpha \hat{A}_\beta \chi_\alpha(g)\chi_\beta(h)\right] = \sum_{\alpha\subseteq[n]} \hat{A}_\alpha^2 (-\tau)^{|\alpha|}$$

which is bounded by τ in absolute value, since $\hat{A}_\phi = 0$. Recall that the entire expression is equal $1-\varepsilon$ by assumption. This implies that the second expectation (whose value let us name W) must be at most $-1+\tau+8\varepsilon$. We write it as

$$\begin{aligned}
-1+\tau+8\varepsilon \geq W &= \mathbb{E}_{g,f,\mu}\left[\sum_{\alpha,\beta,\gamma\subseteq[n]} \hat{A}_\alpha \hat{A}_\beta \hat{A}_\gamma \chi_\alpha(g)\chi_\beta(g\mu)\chi_\gamma(f)\right] = \\
&= \sum_{\alpha,\gamma\subseteq[n]} \hat{A}_\gamma \hat{A}_\alpha^2 \mathbb{E}_{f,\mu}\left[\chi_\alpha(\mu)\chi_\gamma(f)\right] \\
&= \sum_{\gamma\subseteq\alpha\subseteq[n]} \hat{A}_\gamma \hat{A}_\alpha^2 (-1+\tau)^{|\gamma|}(-\tau)^{|\alpha\setminus\gamma|}.
\end{aligned}$$

where the last equality holds because of the correlation of μ and f. In particular, (i) if $\gamma \not\subseteq \alpha$ then $\mathbb{E}_{f,\mu}\left[\chi_\alpha(\mu)\chi_\gamma(f)\right] = 0$, and (ii) for every $i \in [n]$, $\mathbb{E}[\mu(i)] = \tau$ and $\mathbb{E}[f(i)\mu(i)] = -1+\tau$.

We now bound the absolute value of this sum, following [22]. First we claim that

$$\sum_{\gamma\subseteq\alpha}((1-\tau)^{|\gamma|}(\tau)^{|\alpha\setminus\gamma|})^2 \leq (1-\tau)^{|\alpha|}.$$

The left hand side is the probability that tossing $2|\alpha|$ independent τ-biased coins results in a pattern $\gamma\gamma$ where $\gamma \in \{0,1\}^{|\alpha|}$. This probability is $(\tau^2 + (1-\tau)^2)^{|\alpha|} \leq (1-\tau)^{|\alpha|}$ since $\tau < 1-\tau$. By the Cauchy-Schwarz inequality,

$$\sum_{\gamma\subseteq\alpha} |\hat{A}_\gamma|(1-\tau)^{|\gamma|}(\tau)^{|\alpha\setminus\gamma|} \leq \sqrt{\sum_{\gamma\subseteq\alpha}|\hat{A}_\gamma|^2} \cdot \sqrt{\sum_{\gamma\subseteq\alpha}((1-\tau)^{|\gamma|}(\tau)^{|\alpha\setminus\gamma|})^2} \leq (1-\tau)^{|\alpha|/2}$$

so, splitting the sum into $|\alpha| = 1$ and $|\alpha| > 1$,

$$|W| \leq \sum_{|\alpha|=1} |\hat{A}_\alpha^2|(1-\tau) + \sum_{|\alpha|>1} |\hat{A}_\alpha|^2(1-\tau)^{|\alpha|/2}\,.$$

Let $\rho = \sum_{|\alpha|>1} |\hat{A}_\alpha|^2$. We have $|W| \leq (1-\rho)(1-\tau) + \rho(1-\tau)^{3/2}$, since $\hat{A}_\alpha = 0$ for $|\alpha|$ even. Thus

$$1 - \tau - 8\varepsilon \leq |W| \leq (1-\tau)((1-\rho) + \rho\sqrt{1-\tau}) \quad \Rightarrow \quad \rho \leq \frac{8\varepsilon}{(1-\tau)(1-\sqrt{1-\tau})}\,.$$

Since $\tau = \frac{1}{100}$ is fixed, we have $\rho = O(\varepsilon)$. ■