Assignment Testers: Towards a Combinatorial Proof of the PCP-Theorem*

Irit Dinur† Omer Reingold ‡

January 29, 2006

Abstract

In this work we look back into the proof of the PCP Theorem, with the goal of finding new proofs that are "more combinatorial" and arguably simpler. For that we introduce the notion of an assignment tester, which is a strengthening of the standard PCP verifier, in the following sense. Given a statement and an alleged proof for it, while the PCP verifier checks correctness of the *statement*, the assignment-tester checks correctness of the statement *and the proof*. This notion enables composition that is truly modular, i.e., one can compose two assignment-testers without any assumptions on how they are constructed. A related notion called *PCPs of Proximity* was independently introduced in [BSGH+04].

We provide a toolkit of (non-trivial) generic transformations on assignment testers. These transformations may be interesting in their own right, and allow us to present the following two main results:

1. The first is a new proof of the PCP Theorem. This proof relies on a rather weak assignment tester given as a "black box". From this, we construct combinatorially the full PCP. An important component of this proof is a new combinatorial aggregation technique (i.e., a new transformation that allows the verifier to read fewer, though possibly longer, "pieces" of the proof). An implementation of the black-box tester can be obtained from the algebraic proof techniques that already appear in [BFLS91, FGL+91].
2. Our second construction is a "standalone" combinatorial construction showing $NP \subseteq PCP[polylog, 1]$. This implies, for example, that approximating max-SAT is quasi-NP-hard. This construction relies on a transformation that makes an assignment tester "oblivious": so that the proof locations read are independent of the statement that is being proven. This eliminates, in a rather surprising manner, the need for aggregation in a crucial point in the proof.

1 Introduction

The PCP Theorem is a characterization of the class NP which was discovered in the early 90's [AS98, ALM+98] following an exhilarating sequence of results, including [GMR89, Bab85,

*Preliminary version of this work appeared in FOCS '04.

†Hebrew University. Part of this research was performed while at NEC Research, Princeton NJ. Email: dinuri@cs.huji.ac.il.

‡Incumbent of the Walter and Elise Haas Career Development Chair, Department of Computer Science and Applied Mathematics, Weizmann Institute, Rehovot, 76100 Israel. Email: omer.reingold@weizmann.ac.il. Part of this research was performed while at AT&T Labs - Research. Florham Park, NJ, and while visiting the Institute for Advanced Study, Princeton, NJ. Research was supported in part by US-Israel Binational Science Foundation Grant 2002246.

BGKW88, FRS94, LFKN92, Sha92, BFL91, BFLS91, FGL+91] to list just a few. It has had tremendous impact; most notably it lies at the heart of virtually all inapproximability results, starting with the seminal work of [FGL+91].

Recall that a language L is in NP if there is a polynomial-time algorithm (verifier) that can verify whether an input is in the language, with the assistance of a proof, called the NP witness. The PCP Theorem says that every NP witness can be rewritten in a "PCP" format that allows ultra-efficient (probabilistic) checking. Hence the name, **P**robabilistically **C**heckable **P**roofs.

More concretely, the PCP verifier is an algorithm that is given direct access to a proof, and also a logarithmic number of random coins. The verifier reads the input, tosses the random coins, and then decides which (constant number of) bits to read from the proof. Based on the content of these bits, the verifier decides whether to accept or reject. The PCP Theorem asserts the existence of a polynomial-time verifier for any $L \in NP$ such that:

- (Completeness) For every $x \in L$ there is a proof that causes the verifier to always accept.
- (Soundness) For every $x \notin L$, every alleged proof causes the verifier to reject with probability 99% over its coin tosses.

Let us fix the NP-language L in our discussion to be (circuit) satisfiability (SAT). For every fixed outcome of the random coin tosses of the verifier, the verifier's action can be described by specifying which (constant number of) bits are read from the proof, along with the acceptance predicate over these bits. Enumerating over all possible random coin tosses, one gets a list of $\mathsf{R} = 2^{O(\log n)} = n^{O(1)}$ constant-size predicates (described, say, via circuits) over Boolean variables representing the bits in the proof. Thus, the verifier can be thought of as a deterministic polynomial-time reduction whose input is a Boolean circuit φ of size n (denoted $|\varphi| = n$), and whose output is a list of $\mathsf{R} = n^{O(1)}$ circuits $\psi_1, \ldots, \psi_{\mathsf{R}}$ over a new set of variables, such that the following holds:

- (Completeness) If φ is satisfiable, then there is an assignment that simultaneously satisfies all of $\psi_1, \ldots, \psi_{\mathsf{R}}$.
- (Soundness) If φ is unsatisfiable, then every assignment simultaneously satisfies at most 1% of $\psi_1, \ldots, \psi_{\mathsf{R}}$.

This result is already non-trivial for $|\psi_i| = o(n)$, but in fact holds for $|\psi_i| = O(1)$. Indeed, the discovery of this theorem was extremely surprising. The proof is not an easy one, and combines many beautiful and influential ideas.

1.1 Overall Goals

Acknowledging the importance of the PCP Theorem, we look back into its proof, with the goal of finding proofs that are substantially different and desirably also simpler. We note that while the statement of the PCP Theorem is purely "combinatorial", the original proof of [AS98, ALM+98] is heavily based on algebra: low degree polynomials play a crucial role. Therefore, of particular interest is coming up with proofs of "combinatorial" nature, see also [GS97]. In that we were also influenced by the view that such combinatorial proofs, albeit more messy, are sometimes more intuitive (or at least may shed new intuition). We also note that such new proofs and constructions have the potential of implying new results, unknown with previous techniques. A recent example is the combinatorial construction of expander graphs and the subsequent construction of the so called lossless expanders [RVW00, CRVW02].

Composition and Recursion. We first tackle a major ingredient in the original proof, namely the use of composition. Composition is indeed natural in this setting. A given verifier reduction may possibly be improved by applying another (inner) verifier reduction to each one of $\psi_1, \ldots, \psi_{\mathsf{R}}$, replacing each ψ_i with a system Υ_i of even smaller circuits.

Unfortunately, this simplistic composition has a "consistency flaw". It is likely that the resulting system of circuits $\bigcup_i \Upsilon_i$ be completely satisfiable, even if the original circuit φ is unsatisfiable. Indeed each one of the ψ_i's could be individually satisfiable, as it is only impossible to satisfy more than 1% of them *by the same assignment*. Since $\Upsilon_1, \ldots, \Upsilon_{\mathsf{R}}$ are outcomes of independent runs of the second verifier reduction, they are defined over syntactically disjoint sets of variables. This makes it easy to combine inconsistent assignments (each satisfying one ψ_i) into an assignment that satisfies all of $\bigcup \Upsilon_i$.

In the proof of [AS98, ALM+98], this difficulty was overcome by having the second verifier utilize the concrete structural details of the circuits $\psi_1, \ldots, \psi_{\mathsf{R}}$ output by the first verifier, and relying on ingenious consistency mechanisms of algebraic nature.

In search of a simpler and more modular approach to composition, we introduce a natural strengthening of the PCP verifier that we call *assignment tester*. Intuitively, an assignment tester does not only verify satisfiability, but rather satisfiability *by a specified assignment* (given as oracle). This will provide an alternative and simple way of ensuring consistency in the context of PCP composition.

Assignment testers. An assignment tester is a polynomial-time reduction, whose input is a circuit φ over a set of variables X, and whose output is a list of polynomially many significantly-smaller circuits $\Psi = \{\psi_1, \ldots, \psi_{\mathsf{R}}\}$ over *both* X and auxiliary variables Y. The guarantee of the reduction (for a complete definition see Definition 3.1) is that for every possible assignment $a : X \to \{0,1\}$,

- (Completeness) If a satisfies φ, then there is an extension of a, namely an assignment b for Y, such that all of $\psi_1, \ldots, \psi_{\mathsf{R}}$ are satisfied by $a \cup b$.
- (Soundness) If a is "far" from any satisfying assignment for φ then every extension of a to Y can satisfy at most 1% of $\psi_1, \ldots, \psi_{\mathsf{R}}$.

Thus, even if φ is satisfiable, but not by anything close to a, then 99% of $\psi_1, \ldots, \psi_{\mathsf{R}}$ must reject any extension of a. An intuitive way to understand the difference between a standard PCP verifier and an assignment-tester is the following. Given a statement (a predicate φ claimed to be satisfiable) and an alleged proof for it (an assignment for X), the verifier checks that the statement is correct. In contrast, the assignment-tester checks that *the proof* is correct. A related notion was independently introduced by Ben-Sasson et. al. [BSGH+04], see further discussion below.

With this notion, we proceed to prove a composition theorem that is truly modular. The main idea is the following: Previously, relations between ψ_i and ψ_j (for $i \neq j$) were lost upon reduction to Υ_i and Υ_j. Now, all of the R reductions (each reducing ψ_i to the system Υ_i) are correlated through having to refer to the same assignment for X. To carry out this intuition, we give a generic transformation of any assignment-tester into a 'robust' one (we discuss this notion below). Once the first (outer) assignment-tester is robust, the naive composition is trivially sound.

1.2 Our Contributions

To be able to discuss our results, let us first briefly state the parameters of assignment testers. Let R be the number of circuits output by the assignment-tester, and let s upper bound their size. There are three additional parameters: the query complexity q (how many variables are read by each output circuit), the error-probability ε (what fraction of the output circuits may erroneously accept on a "no" input; $\varepsilon = 0.01$ in the above discussion), and the distance parameter δ (what distance of the input assignment from a satisfying assignment should make the tester reject). For simplicity, we will ignore the three last parameters in most of the following discussion, with the implicit understanding that any mention of an assignment-tester means constant q, ε and δ.

New Proofs of Versions of the PCP Theorem

As we discuss below, this paper provides a variety of transformations on assignment testers, complementing the generic composition theorem already mentioned above. Armed with this "toolkit" of transformations, we consider various ways of composing assignment testers with the goal of reproving the PCP Theorem. We now elaborate on two results we obtain in this manner.

An assignment tester with $\mathsf{s} = n$ (recall that s is the size of the circuits produced by the assignment tester reduction) is completely trivial (letting the output equal the input). Nevertheless, we prove that given an assignment-tester with $\mathsf{s} = n^{0.99}$, we can get s all the way down to a constant.

Theorem 1.1 (Informal Statement) *Given an assignment-tester with* $\mathsf{R} = n^{O(1)}$ *and* $\mathsf{s} = n^{0.99}$*, we construct an assignment tester with* $\mathsf{R} = n^{O(1)}$ *and* $\mathsf{s} = O(1)$*.*

The idea of the construction is to use the given assignment-tester as a building-block, and to compose it with itself. The output circuits will have size $n^{0.99}, n^{(0.99)^2}, n^{(0.99)^3}$, and so on, $n^{(0.99)^t}$ after t compositions. So taking $t = \log\log n$ results in a polynomially long list of constant size circuits. However, since the composition step roughly sums up the error probabilities of the two components, going beyond a constant number of composition steps requires an additional ingredient. This is where we incorporate several of the assignment-tester transformations mentioned above, to achieve error reduction. Particularly, we employ a new combinatorial aggregation technique (namely, introducing new 'aggregate' variables which represent ℓ-tuples of old variables).

Our building-block assignment tester (i.e., the one with $\mathsf{s} = n^{0.99}$) can be constructed using *algebraic* techniques (for example, such an assignment tester can be constructed via techniques already present in [BFLS91, FGL+91][1]). Coming up with a combinatorial construction for such an assignment tester would yield a completely combinatorial proof of the PCP Theorem. Subsequently to our work Dinur [Din05] *directly* (i.e., not going through this building block) gave a completely combinatorial proof of the PCP Theorem.

Next, we present a combinatorial construction of an assignment tester, which is quasi-polynomial. It gives a combinatorial proof for $NP \subseteq PCP[polylog, 1]$ and implies, for example, that approximating Max-3-SAT is quasi-NP-hard.

Theorem 1.2 (Informal Statement) *There exists an explicit combinatorial construction of an assignment tester with* $\mathsf{R} = n^{poly\log n}$ *and* $\mathsf{s} = O(1)$*.*

[1] Taking a low degree extension with a constant dimension as in [PS94].

This construction does not rely on any algebraic techniques, rather it is based on recursive applications of our combinatorial transformations. In particular the construction relies on a transformation that makes an assignment tester "oblivious": so that the proof locations read are independent of the statement that is being proven. This eliminates, in a rather surprising manner, the need for aggregation in a crucial point in the proof.

The starting point is the previous construction, again relying on $\log \log n$ steps of composition and recursion. However, to compensate for the lack of a powerful building-block, the main idea is to construct it ourselves, recursively. Thus the main step involves constructing an assignment tester with $\mathsf{s} = \mathsf{n}^{\alpha}$ (for some constant $\alpha < 1$) relying on recursion.

The resulting construction of assignment tester is analogous to the following recursive construction of error correcting codes based on tensor products: First, put your n-bit input in a $\sqrt{n} \times \sqrt{n}$ matrix. Now, recursively apply an error correcting code to each row. Then, making additional recursive calls, apply an error correcting code to each column (of the new matrix). Finally, as the relative distance has slightly deteriorated by this process, one can use simple transformations to amplify it back. Our related construction, in the context of assignment testers, is naturally more delicate and requires new ideas. However, the overall structure is very similar.

We note that this construction makes use of a *constant-size* assignment tester, to facilitate the composition. That is, the construction relies on an assignment tester that is only required to work on inputs of size $\leq n_0$ for some large enough constant n_0. (The only requirement from this assignment tester is that it produces small enough circuits. Say circuits smaller than $s_0 = (n_0)^{0.99}$.) As in a similar situation in [RVW00], such an object can be obtained via an exhaustive search over a constant range. However, the only proofs for its existence that we know of, rely on previous constructions of PCP verifiers. Nevertheless, we can take the constant-size assignment tester needed by the construction to be an instantiation (for constant size inputs) of extremely inefficient constructions, e.g., a Long-code based assignment-tester [BGS98].

Combinatorial Transformations on Assignment Testers

As the above description of our constructions indicates, our main technique is perhaps the most basic technique in computer-science, namely recursion. This is implemented through the basic composition of a relatively weak assignment-tester with a (strong) assignment tester of smaller size that is inductively constructed. Particularly, the composition theorem mentioned above gives a way for reducing the circuit size (s) of an assignment tester (from s_1 or s_2 to $\mathsf{s}_1 \circ \mathsf{s}_2$).

We study several generic transformations on assignment testers that serve to improve various other parameters. We describe combinatorial methods to reduce the error probability, the query complexity, and the distance parameter. While these transformations are required for our constructions to work, we believe they are interesting in their own right. In particular, they provide tradeoffs that allow us to focus our attention on the important parameters of an assignment tester (e.g., it allows us to focus on constructing assignment testers with constant error and constant distance parameter).

Robustness. We mentioned above that for our generic composition theorem to work, the first (outer) assignment-tester needs to be converted into a 'robust' one. This means that in the 'no' case, not only do $1 - \varepsilon$ fraction of the circuits $\psi_1, \ldots, \psi_{\mathsf{R}}$ reject, but moreover they 'strongly' reject in that their input is at least δ-far from any satisfying input. More formally, recall that each circuit reads some q variables. These are both X-variables (the original input variables of

the input circuit), which are bits, and also auxiliary Y-variables that may come from a larger alphabet. We interpret here the Y-variables as bit strings (i.e., chunks of bits that are always read as a whole) and therefore the input of each ψ_i can be viewed as a longer bit string which is the concatenation of all the variables it reads. It is now well defined to say that an assignment to the input variables of ψ_i is δ-far (in Hamming distance) from any satisfying input.

We show a simple generic transformation taking every assignment tester into a robust one, where the robustness parameter is inversely related to the number of variables read, q. Loosely, the transformation goes as follows: given the output $\psi_1, \ldots, \psi_{\mathsf{R}}$ of the tester, construct a revised output $\psi'_1, \ldots, \psi'_{\mathsf{R}}$ over new variables that are supposed to be the error corrected version of the old variables. (Here too the Y-variables are interpreted as bit strings and each X-variable is simply encoded by repetition.) Each ψ'_i now decodes its input variables and applies ψ_i on the decoded variables. Our transformation uses an off-the-shelf error-correcting code and its efficiency depends on standard parameters of the code (essentially on its distance, its rate and the complexity of decoding codewords).

With this generic transformation, we can completely ignore the notion of robustness in all other transformations and only refer to it in the composition theorem. We note in passing that [BSGH+04] have the same notion of robustness. However, in their context, even the modest cost of our generic transformation is impermissible. Therefore, [BSGH+04] works with assignment-testers that are already robust (and indeed they name these objects '*Robust* PCP of Proximity'). This difference between our definitions is further discussed in Section 7.

Distance reduction. The only new parameter of assignment testers, not already used by PCP verifiers is their distance parameter δ (what distance of the input assignment from a satisfying assignment should make the tester reject). We provide a generic method for strengthening assignment-testers so that they identify smaller deviations from satisfying assignments. This transformation comes at a fair cost in other parameters. The main idea is the following: On input circuit φ, over Boolean variables X, encode X with "amplified" distance (so that two assignments for X that are δ' far from each other will be encoded by assignments that are $\delta > \delta'$ apart), and let φ' be the corresponding circuit (that first decodes its input and then applies φ). Now, apply the original assignment tester on φ' (instead of φ).

Error reduction with aggregation. It is easy to reduce the error probability of an assignment tester by repetition: replace $\Psi = \{\psi_1, \ldots, \psi_{\mathsf{R}}\}$ by ANDs of all possible ℓ-tuples of circuits in Ψ. This reduces the error-probability from ε to ε^ℓ but causes an ℓ-fold increase in the number of queried variables. We cannot afford such an increase, as it hurts the effective 'robustness' of the assignment tester, a crucial property for composition. We avoid this increase through aggregation (or alternatively, parallelization). That is, the introduction of new variables to represent ℓ-tuples of previous ones. Our new method of aggregation is purely combinatorial.

The original proof of the PCP theorem also contains an aggregation method, based on sophisticated algebraic techniques. One disadvantage of that method is that it causes a blowup in the size of the domain of the variables, by a factor that is roughly $\log |X|$ (more precisely, it is the degree of the low degree representation of the assignment for X), which cannot be afforded in our context. In contrast, our combinatorial aggregation increases the variables by a factor independent of $|X|$. On the other hand, our combinatorial aggregation introduces $|X|^\ell$ new variables, even if we are only interested in the values of some t possible ℓ-tuples. This is much worse than the $poly(\ell, |X|, t)$ new variables and tests, introduced by the algebraic aggregation.

Nevertheless, this blow-up is tolerable in our context as we only care about $\ell = O(1)$ (since we only plan on reducing error from one constant to another).

The main idea of our aggregation is the following: when reading ℓ-tuples of variables, to simulate the AND of ℓ circuits we are faced with the difficulty that the assignment to the ℓ-tuples may not be consistent with any assignment to the original variables. This problem is exactly what makes the proof of Raz's parallel repetition theorem [Raz98] so challenging. Nevertheless, in our context, we can tolerate a more modest drop in the error probability than in [Raz98], and can afford adding a constant (independent of ℓ) number of queries. So our main idea is the following: simply add a few queries directly aimed at testing the consistency of the ℓ-tuples of variables. This simple idea results in a significantly simpler analysis. In particular, a sufficiently good consistency test was already provided by Goldreich and Safra [GS97]. We give direct and simple analysis of essentially the same test.

Related Work

We have already mentioned that Ben-Sasson et. al. [BSGH+04] independently introduced an object called "PCP of Proximity" which is essentially the same as our assignment tester, presented in a somewhat different language. The work of [BSGH+04] shows connections of these objects to locally testable codes and therefore the results of this paper seem relevant in this context as well. We further discuss this in Section 7.

The notion of assignment testers is very related to the area of property testing. In fact, both assignment testers and PCPs of Proximity can be viewed as a special case of more general definitions given by Ergun, Kumar and Rubinfeld [EKR99], in the context of proof-assisted testing. To the best of our knowledge, the connection to the construction of PCPs has not been explored in the past. The connection of assignment testers to property testing is elaborated upon in Section 7.

As we discussed above, the motivation for defining assignment testers lies in the desire to obtain simple and modular composition of PCPs. This goal was already explored in the past. In particular, Szegedy [Sze99] derived a syntactic composition theorem for abstractions of PCPs based on many valued logics.

Subsequent to our work, Dinur [Din05] described a combinatorial proof of the PCP Theorem which also uses composition of Assignment-Testers, (and in particular, composition with an Assignment-Tester of constant size just as in our second construction).

Organization

We formally define assignment testers in Section 3, and then proceed to prove the composition theorem. In Section 4 we provide generic transformations on assignment testers. Our two main constructions (Theorems 1.1 and 1.2) are proven in Sections 5 and 6 respectively. The proof of the consistency test (required for the aggregation) can be found in Appendix A.

2 Preliminaries

In this section we define some standard combinatorial objects that our constructions rely upon. These are error correcting codes and hitting sets. Both have a trivial random construction and can also be constructed explicitly.

2.1 Error Correcting Codes

As in the original proof of the PCP theorem [AS98, ALM+98], error correcting codes are a very useful tool for our proof. However, unlike the original proof, we do not rely on algebraic properties of the codes nor do we require the codes to be locally testable. The relevant parameters of the code in our case are rather generic: these are its rate, its distance, and the circuit complexity of verifying and decoding legitimate codewords. We call the latter task, "codeword decoding". In the next lemma we give the parameters of the codes that will imply the most elegant version of our results. As we discuss below, such codes are easy to come by. We also note that much weaker codes still imply our main results.

Lemma 2.1 *There exists a polynomial time computable family of codes* $e = \{e_{\mathsf{w}} : \{0,1\}^{\mathsf{w}} \to \{0,1\}^{O(\mathsf{w})}\}_{\mathsf{w}\in\mathbb{N}}$*, such that* $e_{\mathsf{w}}(\cdot)$ *is a code with minimum distance* w *that satisfies*

Linear circuit size for codeword decoding: *For every* w *there exists a circuit* C_{w} *of size* $O(\mathsf{w})$ *that takes as input a string* $z \in \{0,1\}^{O(\mathsf{w})}$ *and outputs* y *such that* $z = e_{\mathsf{w}}(y)$ *if such a string* y *exists and* $\perp$ *otherwise. Furthermore, the circuit* C_{w} *can be uniformly constructed in time poly*(w)*.*

Lemma 2.1 asks for codes with constant rate and constant relative distance, which is quite standard. In addition, it requires linear circuit size for "codeword decoding", that is, for the task of verifying that a word is a legitimate codeword and then decoding it. This second part of decoding a legitimate codeword is trivial in case $e_{\mathsf{w}}(y)$ contains y as a substring (such codes are sometimes called *systematic*).[2] Linear error correcting codes can be assumed without loss of generality to be systematic (by changing the basis of the generating matrix of the code, one can obtain a generating matrix for the *same code* that contains the identity matrix as a submatrix. Note that changing the basis doesn't change the code, so the parity check matrix is the same.) In addition, whenever a linear code is defined by a sparse parity check matrix (that contains only linear number of non-zero entries), verifying that a word is a legitimate codeword can be performed by a circuit of linear size. In particular, Lemma 2.1 holds for the linear codes that are obtained by selecting a sparse parity check matrix, uniformly at random. In addition, it holds for the *explicit* codes best known under the name LDPC (Low Density Parity Check) codes. Note that the expander LDPC codes of [Gal63, Tan81, SS96, Spi96] have explicit combinatorial constructions based on the combinatorial construction of Expander Graphs in [RVW00].

Notation 2.2 *We denote by* e_{w}^{-1} *the "maximum likelihood" decoding transformation that corresponds to the code* e_{w}*. That is,* $e_{\mathsf{w}}^{-1}(z') = y$ *if* $z = e_{\mathsf{w}}(y)$ *is the codeword of minimal Hamming distance to* z' *(where ties can be broken arbitrarily). In particular,* $e_{\mathsf{w}}^{-1}(e_{\mathsf{w}}(y)) = y$ *(assuming* $\mathsf{w} > 0$*). We do not assume anything on the computability of this mapping (in particular we do not assume that it is polynomial-time computable). It is important to note that this maximum likelihood notion of decoding has little to do with "codeword decoding" as defined in Lemma 2.1.*

[2] A natural approach for obtaining efficient codeword decoding is the following. First, slightly revise the error correcting code such that it will be systematic (simply appended y to $e_{\mathsf{w}}(y)$). Now codeword decoding is as easy as encoding: First, we can extract y from the alleged codeword, then re-encode y and finally we can compare the result with the original alleged codeword. Unfortunately, error correcting codes with linear size circuits for encoding are not known (and may very well not exist). Instead, one may use error correcting codes with quasi-linear size circuits for encoding (these are not hard to come by). This natural approach can therefore give codes that are only slightly less efficient than Lemma 2.1 promises, and are still good enough for obtaining the main results of this paper.

2.2 Hitting Sets

We specify parameters of two families of sets with standard hitting properties. We then spell out in Corollary 2.5 the precise (standard) use of them for "error reduction". Both of these hitters can be constructed in an elementary way based on expander graphs. The hitters we give here rely on optimal expanders known as Ramanujan graphs [LPS88]. We note that our main results can be proven using significantly weaker hitters (such as those implied by the expander graphs of [RVW00]).

Lemma 2.3 *Let N be an integer and $0 \leq \mu \leq \alpha \leq 1/2$ two real values. Then there exists a family $\mathcal{F} = \{F_1, \ldots, F_N\}$ such that*

1. *Each F_i is a k-tuple of integers in $[N] = \{1, \ldots N\}$, with $k = O(\alpha/\mu)$.*
2. *Every subset $T \subset [N]$ of size at least μN intersects at least αN of the F_i's (i.e., $|\{i : F_i \cap T \neq \phi\}| \geq \alpha N$).*
3. *Given N, i, α and μ, each F_i can be constructed uniformly in polynomial time (in the input and output lengths).*

The set $\mathcal{F}$ in Lemma 2.3, can be constructed from a k-regular expander graph G_N on the set of vertices $[N]$. Each F_i is simply the neighbor list of vertex i. Requirement (2) asks for sets of size μN to expand by a factor α/μ which for the expanders of [LPS88] can be obtained with degree $k = O(\alpha/\mu)$ (for the expanders of [RVW00] it requires $k = poly(\alpha/\mu)$).

Next, we consider a more traditional setting of the hitting problem. For that we use the so called combined hitter from Corollary C.5 in [Gol97],

Lemma 2.4 *Let N be an integer and β and μ be two positive real values. Then there exists a set $\mathcal{F} = \{F_1, \ldots, F_M\}$ of size $M = N \cdot poly(1/\beta)$ such that*

1. *Each F_i is a k-tuple of integers in $[N]$, with $k = O(\log(1/\beta)/\mu)$.*
2. *Every subset $T \subset [N]$ of size at least μN intersects at least $(1-\beta)M$ of the F_i's (i.e., $|\{i : F_i \cap T \neq \phi\}| \geq (1-\beta)N$).*
3. *Given N, i, β and μ, each F_i can be constructed uniformly in polynomial time (in the input and output lengths).*

The usefulness of Lemma 2.4 for this paper is in the standard application of hitters to error reduction. Particularly, we will use the following immediate corollary.

Corollary 2.5 *Let $\psi_1, \ldots, \psi_N$ be a sequence of N circuits over a set of variables Y. Let β and μ be two positive real values. Then there exists a sequence of $M = N \cdot poly(1/\beta)$ new circuits $\psi'_1, \ldots, \psi'_M$ such that*

1. *Each new circuit ψ'_i is the AND of k old circuits ψ_i with $k = O(\log(1/\beta)/\mu)$. In particular, every assignment to the variables Y that satisfies all of the old circuits also satisfies all of the new circuits.*
2. *Every assignment to the variables Y that causes μN of the old circuits to reject also causes $(1-\beta)M$ of the new circuit to reject.*
3. *On input $\psi_1, \ldots, \psi_N$, β and μ, the new sequence can be constructed uniformly in polynomial time (in the input and output lengths).*

3 Assignment Testers and their Composition

In this section we formally introduce the notion of an *assignment tester*, which is an enhancement of the PCP verifier. As discussed in the introduction, the motivation for assignment testers is in the rather simple and natural way two assignment testers compose. This property is very appealing as composition is a major ingredient in the proof of the PCP Theorem.

Like the PCP verifier, an assignment tester reduces an input circuit φ over variables X into a list of output circuits $\psi_1, \ldots, \psi_{\mathsf{R}}$ over the variables (X and) Y. The main difference is that the output circuits of the PCP verifier might not depend on X at all, while the output circuits of the assignment tester certainly do. Moreover, the completeness and soundness conditions of an assignment tester are with respect to a specific assignment for X, rather than with respect to the general satisfiability of φ. Loosely, an assignment tester doesn't just check that the input is *satisfiable*, but rather that the input is *satisfied by a specified assignment*.

This simplifies composition by eliminating consistency issues altogether. Recall that the main idea of composition is to improve a given verifier reduction by applying another (inner) verifier reduction to each one of $\psi_1, \ldots, \psi_{\mathsf{R}}$, replacing each ψ_i with a system Υ_i of even smaller circuits. By feeding the same assignment to each of the parallel runs of the inner reduction, all of the systems Υ_i directly refer to satisfiability by the same single assignment.

An important parameter, inherent to an assignment tester, is its *distance* parameter. In the soundness condition, it is unreasonable to require that in case the assignment for X is not satisfying, a sizeable fraction of $\psi_1, \ldots, \psi_{\mathsf{R}}$ reject. If we wish each ψ_i to read only a constant number of bits, most ψ_i's won't be sensitive to a single bit flip in X turning a satisfying assignment into an unsatisfying one. Thus, we only require that if the assignment is δ-far (i.e., is at relative Hamming distance δ) from *every* satisfying assignment, then an $1 - \varepsilon$ fraction of $\psi_1, \ldots, \psi_{\mathsf{R}}$ must reject. The parameter $\delta > 0$ is the distance parameter of the assignment tester, and it should be at least inversely proportional to the number of variables (unless we are willing to compromise on the detection probability $1 - \varepsilon$ being subconstant).

In subsection 3.1 we give the formal definition of an assignment tester. In subsection 3.3 we define 'robust' assignment testers and prove an immediate composition theorem for this object. We show a generic way to transform every assignment tester into a robust one ("robustization") in section 3.4. Finally, in subsection 3.5 we combine the above and deduce a composition theorem of assignment testers. We consider additional transformations on assignment testers in Section 4. In Section 7, we shortly discuss the relation between PCP testers and property testers.

3.1 Defining Assignment-Testers

We denote Boolean circuits by φ, ψ, etc., and refer to the predicate computed by the circuit by the same name. We say that an assignment is δ-far from satisfying a circuit φ, if its relative Hamming distance from *every* satisfying assignment for φ is at least δ.

Definition 3.1 (Assignment-Tester) *An Assignment-Tester with parameters* $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$ *is a reduction whose input is a Boolean circuit* φ *of size* n *over Boolean variables* X*. The reduction outputs a system of* $\mathsf{R}(\mathsf{n})$ *Boolean circuits* $\Psi = \{\psi_1, \ldots, \psi_{\mathsf{R}}\}$ *each of size at most* $\mathsf{s}(\mathsf{n})$ *over* X *and auxiliary variables* Y *such that the following conditions hold:*

- *The running time of the algorithm is polynomial in* n *and* $\mathsf{R}(\mathsf{n})$*.*

- $\mathsf{R}(n)$ - Number of output circuits. Reminiscent of the amount of Randomness of the verifier.
- $\mathsf{s}(\mathsf{n})$ - Size of output circuits.
- $\mathsf{q}(\mathsf{n})$ - Maximal number of variables read (or queried) in one circuit.
- $\delta(\mathsf{n})$ - Distance to a satisfying assignment.
- $\varepsilon(\mathsf{n})$ - Error probability: the fraction of circuits that erroneously accept a far-from-satisfying assignment. (Sometimes we consider the detection probability $\gamma = 1 - \varepsilon$, i.e. the remaining fraction of circuits that reject).
- $\mathsf{w}(\mathsf{n})$ - 'Width' of Y variables, i.e. log of alphabet size. This parameter plays a minor role in our discussion and so we usually omit it. We note that w is smaller than s.

Figure 1: The parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$ of an assignment tester.

- *Each ψ_i depends on $\mathsf{q}(\mathsf{n})$ variables from $X \cup Y$. The variables in Y take values in an alphabet Σ, and are accessible to ψ_i as a tuple of $\mathsf{w}(\mathsf{n}) = \lceil \log |\Sigma| \rceil$ bits*[3].
- *For every assignment $a : X \to \{0,1\}$,*
  1. *[Completeness:] If a satisfies φ then there exists an assignment $b : Y \to \Sigma$ such that $a \cup b$ satisfies all of $\psi_1, \ldots, \psi_{\mathsf{R}}$.*
  2. *[Soundness:] If a is δ-far from every satisfying assignment for φ, then for every assignment $b : Y \to \Sigma$, at least $1 - \varepsilon$ of $\psi_1, \ldots, \psi_{\mathsf{R}}$ reject $a \cup b$.*

This definition should be compared to the standard notion of a PCP verifier. The PCP verifier can also be defined as a reduction[4] precisely as above, but there are two differences. One is superficial in that the PCP verifier produces circuits that only depend on the (new) Y variables. The main difference is in the completeness and soundness conditions which in the case of the PCP verifier reduction are defined as follows:

1. *[Completeness:] If φ is satisfiable, then there exists an assignment $b : Y \to \Sigma$ that satisfies all of $\psi_1, \ldots, \psi_{\mathsf{R}}$.*
2. *[Soundness:] If φ is unsatisfiable, then for every assignment $b : Y \to \Sigma$, at least $1 - \varepsilon$ of $\psi_1, \ldots, \psi_{\mathsf{R}}$ reject.*

Every assignment tester is also a PCP verifier reduction. To see for example that the soundness condition carries over, observe that if the input φ is unsatisfiable, then any $a : X \to \{0,1\}$

[3] So accessing all $\mathsf{w}(\mathsf{n})$ bits of a single variable in Y, counts as a single "query".

[4] The PCP verifier is usually described as a probabilistic polynomial-time algorithm that verifies a (PCP) proof by tossing r random coins and then probing the proof in some q locations. By considering the action of the verifier in parallel over all possible outcomes of the random coins, the verifier corresponds to a list of 2^r circuits (each over q input variables). Thus, the verifier can also be viewed as a (deterministic) reduction that outputs a list of circuits. We call this a PCP verifier reduction.

is ($\delta = 1$)-far from all (non-existent) satisfying assignments. Thus it cannot be extended with b so as to satisfy more than ε of $\psi_1, \ldots, \psi_{\mathsf{R}}$.

The converse is not necessarily true since in an arbitrary PCP verifier reduction we have no control over the dependence of $\psi_1, \ldots, \psi_{\mathsf{R}}$ on X. In particular, the output circuits may not even depend on the variables in X. Interestingly, the original proof of the PCP theorem implicitly constructs assignment-testers rather than just PCP verifiers.

Theorem 3.2 (The PCP Theorem [AS98, ALM+98]) *There is a polynomial-time PCP verifier algorithm (alternatively, assignment-tester) with* $\mathsf{R}(\mathsf{n}) = \mathsf{n}^{O(1)}$ *and* $\mathsf{q}(\mathsf{n}) \cdot \mathsf{w}(\mathsf{n}) \leq \mathsf{s}(\mathsf{n}) = O(1)$*, and constant* $0 < \varepsilon, \delta < 1$*.*

3.2 On the Width of Variables

An assignment tester produces circuits ψ_i that are defined over two different kinds of variables. The X variables are Boolean, whereas the auxiliary Y variables take value from a possibly larger alphabet Σ. We would like to think of the assignment to a Y variable as a w-long bit string. (Recall, that $\mathsf{w}(\mathsf{n}) = \lceil \log |\Sigma| \rceil$ is the width of the Y-variables.) This allows us to view each ψ_i as a Boolean circuit (just as φ is), which is particularly important for the composition theorem (where we apply the inner assignment tester to the circuits ψ_i produced by the outer assignment tester). In particular, if ψ_i reads the assignment to q_x variables from X and to q_y variables from Y then we view its input as a $(\mathsf{q}_x + \mathsf{q}_y \cdot \mathsf{w})$-long bit string which is the concatenation of the assignment to all of the variables it reads. Note that, since the size of each ψ_i is larger than the length of its input, we have that $\mathsf{s} > \mathsf{q}_x + \mathsf{q}_y \cdot \mathsf{w}$. We can now define the restriction of a global assignment to the input variables of a particular ψ_i. This definition will allow repetitions of variables.

Definition 3.3 *Each output circuit in* $\{\psi_1, \ldots, \psi_{\mathsf{R}}\}$ *is defined to be a pair* $\langle C, \tau \rangle$ *where* C *is a circuit over local inputs* $v_1, v_2, \ldots, v_q$ *and* $\tau = (x_{i_1}, \ldots, x_{i_{q_x}}, y_{i_{q_x+1}}, \ldots, y_{i_q})$ *is a tuple of variables from* $X \cup Y$ *specifying which variables are mapped to the inputs of the circuit. We emphasize that* τ *is allowed to have repetition of variables.*[5]

Given an assignment σ *for* $X \cup Y$*, its restriction to* ψ_i *is denoted* $\sigma|_{\psi_i}$ *and is defined as the appropriate string of* $q_x + (q - q_x) \cdot w$ *bits (that possibly reflects the repetitions in* τ*).*

It turns out that almost everywhere, the width parameter w only plays a very minor role. The two related parameters that will be much more crucial to our discussion are the size of the circuits s and the query complexity q. We will therefore almost always omit reference the w and be satisfied with the bound on w implied by s (as discussed above).

3.3 Robust Assignment Testers and their Composition

A *robust* assignment tester is an assignment tester such that in the soundness case, not only do $1 - \varepsilon$ of the output circuits reject, but in fact they see an assignment that is ρ-*far* from a satisfying one (i.e., at least a ρ-fraction of the bits read by each of these circuits need to be changed in order for the circuits to be satisfied). This variant is natural in the context of composition, as will be seen below.

[5] Such repetition will be quite useful below: It implicitly affects the *distance* between the restrictions of assignments to ψ_i, turning it into a weighted Hamming distance (more weight to the repeated variable).

Notation: For a circuit φ, denote by $SAT(\varphi)$ the set of all satisfying assignments for φ. Let $SAT_\delta(\varphi)$ be the set of assignments that are δ-close to some assignment in $SAT(\varphi)$ (namely, assignments that are at relative hamming distance at most δ from some assignment in $SAT(\varphi)$).

Definition 3.4 *An assignment-tester is called ρ-robust if in the soundness case in Definition 3.1 above, for every assignment $b : Y \to \Sigma$, the assignment $(a \cup b)|_{\psi_i}$ is ρ-far from $SAT(\psi_i)$ for at least $1 - \varepsilon$ fraction of $\psi_1, \ldots, \psi_{\mathsf{R}}$.*

It is very easy to compose robust assignment testers.

Lemma 3.5 *let $\mathcal{A}_1, \mathcal{A}_2$ be two assignment-testers with parameters $(\mathsf{R}_1, \mathsf{s}_1, \mathsf{q}_1, \delta_1, \varepsilon_1)$ and $(\mathsf{R}_2, \mathsf{s}_2, \mathsf{q}_2, \delta_2, \varepsilon_2)$ respectively. If $\mathcal{A}_1$ is ρ-robust with $\rho = \delta_2$ then one can construct an assignment tester $\mathcal{A}_3$ with parameters $(\mathsf{R}_3, \mathsf{s}_3, \mathsf{q}_3, \delta_3, \varepsilon_3)$ such that:*

$$\mathsf{R}_3(\mathsf{n}) = \mathsf{R}_1(\mathsf{n}) \cdot \mathsf{R}_2(\mathsf{s}_1(\mathsf{n})), \quad \mathsf{s}_3(\mathsf{n}) = \mathsf{s}_2(\mathsf{s}_1(\mathsf{n})), \quad \mathsf{q}_3(\mathsf{n}) = \mathsf{q}_2(\mathsf{s}_1(\mathsf{n})),$$

and

$$\varepsilon_3(\mathsf{n}) = \varepsilon_1(\mathsf{n}) + \varepsilon_2(\mathsf{s}_1(\mathsf{n})) - \varepsilon_1(\mathsf{n})\varepsilon_2(\mathsf{s}_1(\mathsf{n})), \quad \delta_3(\mathsf{n}) = \delta_1(\mathsf{n})\,.$$

Moreover, if $\mathcal{A}_2$ is ρ_2-robust then so is $\mathcal{A}_3$.

Proof: Given an input φ, the tester $\mathcal{A}_3$ will simply run $\mathcal{A}_1$ on it, outputting $\psi_1, \ldots, \psi_{\mathsf{R}}$, and then run $\mathcal{A}_2$ on each ψ_i. The completeness and the parameters of $\mathcal{A}_3$ follow from the definition. The soundness of $\mathcal{A}_3$ draws on the robustness of $\mathcal{A}_1$ in the following way. Let $\{\psi_{i,j}\}$ be the list of circuits output by $\mathcal{A}_2$ on input ψ_i. The soundness of $\mathcal{A}_2$ asserts that a $1 - \varepsilon_2$ of the $\{\psi_{i,j}\}$ reject if the assignment for ψ_i's variables *is far from* a satisfying one. The robustness of $\mathcal{A}_1$ guarantees that this is indeed the case for $1 - \varepsilon_1$ of the ψ_i's, provided that the assignment for φ's variables is far from a satisfying one. ■

3.4 Robustization

We next show a generic way to transform an arbitrary assignment-tester into a robust one. The idea is to replace each variable in Y with a collection of bits that are supposed to be an encoding via some error correcting code e of the value of the variable. In addition, we repeat the X variables for balance, and modify the output circuits accordingly.

Lemma 3.6 *There exists some $c_1 > 0$ such that given an assignment tester $\mathcal{A}$ with parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$, we can construct a ρ-robust assignment tester $\mathcal{A}'$ with parameters $\mathsf{R}' = \mathsf{R}$, $s' = c_1 \cdot \mathsf{s}$, $\rho = \Omega(\frac{1}{\mathsf{q}})$, $\varepsilon' = \varepsilon$, $\delta' = \delta$.*

This transformation allows us to replace the condition about $\mathcal{A}_1$ being ρ-robust in Lemma 3.5, with a condition about its query complexity, see Theorem 3.7 below. Throughout the rest of the paper the lemma is used only in the proof of Theorem 3.7. In fact there is no further mention of robustness, as the lemma allows us to restrict our attention to the query complexity parameter.

We also mention that this transformation is useless for the setting of [BSGH+04], as they cannot afford the (super-linear) increase in the number of variables that is incurred here.

Proof: $\mathcal{A}'$ will run $\mathcal{A}$ on input φ, and obtain output circuits $\psi_1, \ldots, \psi_{\mathsf{R}}$ over variables X and Y. Each ψ_i will be replaced by a "robust" circuit ψ'_i, whose inputs are encodings (via some error-correcting code e) of the inputs to ψ_i.

Let Σ be the alphabet of the Y variables, and let $\mathsf{w} = \lceil \log |\Sigma| \rceil$ be the number of bits needed to represent a value in Σ. Let $e_{\mathsf{w}} : \Sigma \to \{0,1\}^{\ell}$ be an error correcting code as in Lemma 2.1 (with $\ell = c \cdot \mathsf{w}$ and c a small absolute constant). For each $y \in Y$ introduce new Boolean variables $\bar{v}(y) = v_1(y), \ldots, v_\ell(y)$ supposedly representing y's encoding $e_{\mathsf{w}}(y)$. Denote these new sets of bits by

$$Y' = \bigcup_{y \in Y} \bar{v}(y) \,.$$

Suppose the tuple of variables accessed by ψ_i is $(x_1, \ldots, x_{\mathsf{q}_x}, y_1, \ldots, y_{\mathsf{q}_y})$, with $\mathsf{q} = \mathsf{q}_x + \mathsf{q}_y$. Define a new circuit ψ'_i whose input consists of $\ell \cdot \mathsf{q}$ variables: the first $\ell \mathsf{q}_x$ variables are ℓ copies of each variable of $x_1, \ldots, x_{\mathsf{q}_x}$. The next $\ell \mathsf{q}_y$ variables are $\bar{v}(y_1), \ldots, \bar{v}(y_{\mathsf{q}_y})$. The circuit ψ'_i accepts an input if and only if it is a correct encoding of an input that would have satisfied ψ_i. More explicitly, ψ' accepts input $z_1, \ldots, z_{\ell \mathsf{q}} \in \{0,1\}^{\ell \mathsf{q}}$ if and only if (1) $z^i = (z_{i\mathsf{q}+1}, \ldots, z_{(i+1)\mathsf{q}})$ is the all-0 or all-1 string for $1 \leq i \leq \mathsf{q}_x$, (2) z^i is a legal codeword of e for $i > \mathsf{q}_x$, (3) the values encoded by the z^i's do satisfy ψ_i.

This completes the description of $\mathcal{A}'$ and we now prove its properties. Completeness is clear: a satisfying assignment $a : X \to \{0,1\}$ for φ can easily be extended by $b : Y' \to \{0,1\}$ so that $a \cup b$ satisfies all of $\{\psi'_i\}$.

What is the size of ψ'_i? Since for the code e_{w} there exists a linear size circuit for codeword decoding (see Lemma 2.1), then there exists some c_1 such that the size of each ψ'_i is bounded by $\mathsf{s}(\mathsf{n}) + O(\ell \mathsf{q}) \leq c_1 \cdot \mathsf{s}(\mathsf{n})$.

Next, the soundness of $\mathcal{A}'$. Assume an assignment $a : X \to \{0,1\}$ that is δ-far from satisfying φ. The soundness of $\mathcal{A}$ guarantees that every $b : Y \to \Sigma$ extending a will be rejected by at least $1 - \varepsilon$ of the ψ_i's. What does this mean for the robust version ψ'_i? Consider an arbitrary assignment $b' : Y' \to \{0,1\}$. Such an assignment defines a 'decoded' assignment $b'' : Y \to \Sigma$ by relying on the "maximum likelihood" decoding mapping e_{w}^{-1} of the code e_{w} (see Notation 2.2) as follows:

$$b''(y) : Y \to \Sigma, \quad b''(y) \stackrel{def}{=} e_{\mathsf{w}}^{-1}(b'(v_1(y)), \ldots, b'(v_\ell(y))) \,.$$

The soundness of $\mathcal{A}$ implies that at least $1 - \varepsilon$ of $\psi_1, \ldots, \psi_{\mathsf{R}}$ will reject $a \cup b''$, regardless of the assignment b'' to Y. For each rejecting ψ_i, at least one of the q variables it queries must be re-assigned in order for it to accept. Consider now the corresponding ψ'_i and its assignment $\sigma = (a \cup b')|_{\psi'_i}$. Recall that by definition (see also Definition 3.3) σ is an $\ell\mathsf{q}$-bit-string. Clearly, σ will not satisfy ψ'_i. More importantly, there must be at least one of the q (ℓ-bit)-blocks that need to be changed *into a different legal ℓ-bit-block* in order to turn σ into a satisfying input: If this ℓ-bit block consists of Y'-variables, then it must be changed in more than half the code distance (i.e. $\frac{\mathsf{w}}{2}$) locations. If it consists of (repetitions of) an X-variable, then it must be changed in all ℓ bits. In any case, this means that $\mathcal{A}'$ is ρ-robust with $\rho = \frac{\min(\mathsf{w}/2, \ell)}{\mathsf{q} \cdot c\mathsf{w}} = \frac{1}{2c\mathsf{q}}$.

■

It is important to note that the error correcting code e we use to obtain the robustness property is not part of the definition of an assignment-tester but rather part of the robustization transformation. Furthermore, a feature of this transformation is that e, defined in Lemma 2.1, is quite a generic error correcting code. We do not rely on algebraic properties of e nor require it to be locally testable.

3.5 Generic Composition of Assignment Testers

Combining Lemmas 3.6 and 3.5 we get a convenient composition theorem:

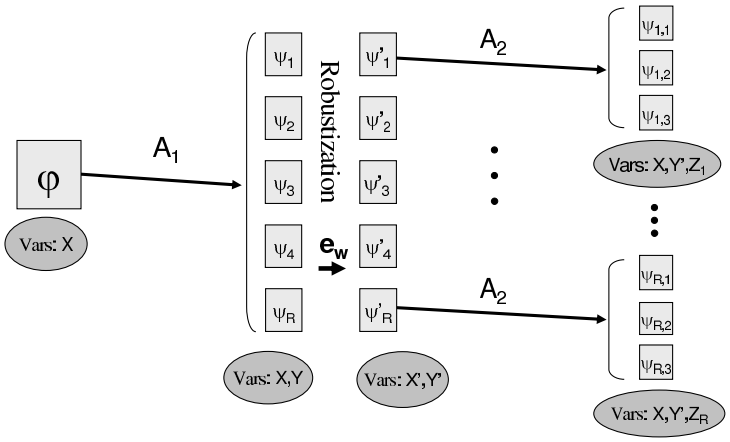

Figure 2: Composing $\mathcal{A}_1$ and $\mathcal{A}_2$

Theorem 3.7 (Composition) *There exists some constants c_1, c_2 such that, let $\mathcal{A}_1, \mathcal{A}_2$ be two assignment-testers with parameters $(\mathsf{R}_1, \mathsf{s}_1, \mathsf{q}_1, \delta_1, \varepsilon_1)$ and $(\mathsf{R}_2, \mathsf{s}_2, \mathsf{q}_2, \delta_2, \varepsilon_2)$ respectively. If $\delta_2 \leq \frac{1}{c_2 \cdot \mathsf{q}_1}$ then one can construct an assignment tester $\mathcal{A}_3$ with parameters $(\mathsf{R}_3, \mathsf{s}_3, \mathsf{q}_3, \delta_3, \varepsilon_3)$ such that, for $\mathsf{n}' \stackrel{def}{=} c_1 \cdot \mathsf{s}_1(\mathsf{n})$:*

$$\mathsf{R}_3(\mathsf{n}) = \mathsf{R}_1(\mathsf{n}) \cdot \mathsf{R}_2(\mathsf{n}'), \quad \mathsf{s}_3(\mathsf{n}) = \mathsf{s}_2(\mathsf{n}'), \quad \mathsf{q}_3(\mathsf{n}) = \mathsf{q}_2(\mathsf{n}'),$$

and

$$\varepsilon_3(\mathsf{n}) = \varepsilon_1(\mathsf{n}) + \varepsilon_2(\mathsf{n}') - \varepsilon_1(\mathsf{n})\varepsilon_2(\mathsf{n}'), \quad \delta_3(\mathsf{n}) = \delta_1(\mathsf{n})\,.$$

Comparing this composition theorem with the robust composition of Lemma 3.5, we see that the condition about $\mathcal{A}_1$ being robust has been removed, and the condition $\delta_2 \leq \rho$ has been replaced by the condition $\delta_2 \leq \frac{1}{c_2 \cdot \mathsf{q}_2}$. The parameters are almost the same, except here $n' = c_1 \cdot \mathsf{s}_1(\mathsf{n})$ rather than $n' = \mathsf{s}_1(\mathsf{n})$ (and this slightly affects $\mathsf{R}_3, \mathsf{s}_3, \mathsf{q}_3$ and ε_3).

We mention that the parameters of $\mathcal{A}_3$ in Theorem 3.7 are very similar to those that would follow from a naïve composition of two PCP verifiers, when in the soundness argument one ignores consistency issues altogether (imagining a prover that is "honest" with respect to consistency). In that sense, these parameters are essentially the best one could hope for in this type of composition.

Proof: We first turn $\mathcal{A}_1$ into a robust assignment tester $\mathcal{A}'_1$, using the transformation of Lemma 3.6. Thus, $\mathsf{n}' = c_1 \cdot \mathsf{s}_1(\mathsf{n})$ is the size of the output circuits of $\mathcal{A}'_1$. Next, we compose $\mathcal{A}'_1$ with $\mathcal{A}_2$ according to Lemma 3.5, obtaining $\mathcal{A}_3$. See also Figure 2 for an illustration of the two transformations combined. ■

4 Transformations on Assignment Testers

The composition of assignment testers is mainly used as a tool for reducing the size, s, of the circuits an assignment tester outputs. In this section we give general transformations for reducing (and thus improving) three additional parameters: (i) The tested *distance*, δ, from a satisfying assignment; (ii) The *error probability*, ε, in case of a far-from-satisfying assignment; and (iii) The number, q, of variables read by each output circuit.

Our motivation is threefold. First, these transformations will come in handy in our constructions of assignment testers. Second, given these transformations it is fair to concentrate on

the construction of testers for some constant values of δ and ε. These parameters can be then reduced to the desired values using the general transformations. Finally, as we believe that the concept of assignment tester is interesting in its own right, it is natural to study the behavior of its various parameters.

To illustrate the need, in our context, for transformations that improve these $(\varepsilon, \mathsf{q}, \delta)$ parameters, note that composition may indeed reduce s, but it incurs costs in other parameters. In particular, it causes the error-probability, ε, to increase, as it is the sum of the error-probabilities of the two composed component. This is easily fixable since it is easy to reduce ε by increasing q. However, if we increase q then we will require a smaller value of δ during composition in the next phase. The transformations of this section will help us essentially enhance the basic composition theorem such that it reduces s without harming other parameters.

4.1 Reducing the Distance: $\delta \longrightarrow \delta'$

In this subsection we describe how the distance parameter δ of a generic assignment-tester can be improved (with fair cost in terms of the other parameters). The goal here is to improve the "sensitivity" of the assignment tester, so that the behavior of its output-circuits on mildly bad assignments (i.e., whose distance from satisfying is at least δ') imitates their behavior on very bad assignments (i.e., whose distance from satisfying is at least $\delta > \delta'$).

Our transformation is of 'black-box' nature: we are given an assignment tester $\mathcal{A}$, whose inner workings we do not wish to manipulate, yet we want to reduce its distance parameter. A natural approach would be to manipulate the input variables X, creating a new set of variables X' (which will be part of the auxiliary variables) that encode X with amplified distance. This encoding would guarantee that two assignments for X that are δ'-apart are encoded by two assignments for X' that are δ-apart. Naturally, manipulating the input variables is not enough and we also need to manipulate φ such that it "recognizes" the X' variables. Specifically, instead of applying $\mathcal{A}$ to φ, we apply it to φ' that is defined over X' (rather than over X), where φ' is defined to decode the assignment to X' and to apply φ to the decoded assignment (viewed as an assignment to X). But we are not done, as the circuits that $\mathcal{A}$ produces on φ' *do not even depend on the X variables* (as φ' does not depend on these variables either). We therefore augment these circuits by a verification that the assignment to X' correctly encodes the assignment to X. We will need to carefully define the encoding of the X variables so that verifying consistency between the X and X' variables would be sufficiently efficient.

Lemma 4.1 (Distance Reduction) *There exists a positive constant $\bar{\delta}$ such that for every $0 < \delta' \leq \delta \leq \bar{\delta}$, given an assignment-tester $\mathcal{A}$ with parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$ we can construct an assignment-tester $\mathcal{A}'$ with parameters $(\mathsf{R}', \mathsf{s}', \mathsf{q}', \delta', \varepsilon)$ where for $Q = O(\frac{1}{\delta'} \log \frac{1}{\varepsilon})$ and $M = O(\frac{\delta}{\delta'} \mathsf{n})$,*

$$\mathsf{R}'(\mathsf{n}) = \mathit{poly}(1/\varepsilon) \cdot \mathsf{R}(M),\ \mathsf{s}' = \mathsf{s}(M) + Q,\ \mathsf{q}'(\mathsf{n}) = \mathsf{q}(M) + Q\,.$$

Remark 4.2 *Note that both the size of the new circuits s' and the number of queries q' contain an additive term of $O(\frac{1}{\delta'} \log \frac{1}{\varepsilon})$. This seems acceptable as it is not hard to show that both s' and q' must be $\Omega(\frac{1}{\delta'} \log \frac{1}{\varepsilon})$ by the definition of assignment testers and lower bounds on the query complexity of hitters [Gol97]. Consider for example an input circuit φ that is only satisfied by the all-zero assignment. For each of the circuits produced by the assignment tester consider the set of variables in X that it reads. For every subset T of variable of density δ', at least $\gamma = 1 - \varepsilon$ fraction of the sets must hit T.*

Proof: We follow the basic sketch outlined above. Let the input for our tester be a circuit φ over Boolean variables X. We fix an encoding E whose properties will be formally defined below. We let X' be new variables whose assignment supposedly represents the encoding via E of some assignment for X. We define the circuit φ' over X' to be a circuit that accepts only assignments for X' that encode (via E) an assignment for X that would have caused φ to accept. In other words, $SAT(\varphi') = E(SAT(\varphi))$. We choose E so that if an assignment $a : X \to \{0,1\}$ is δ'-far from the set $SAT(\varphi)$, then its encoding $b : X' \to \{0,1\}$ is δ-far (recall $\delta > \delta'$) from the set $SAT(\varphi')$.

Now, we run $\mathcal{A}$ on φ' and obtain a list of output circuits $\psi'_1, \ldots, \psi'_{\mathsf{R}(|\varphi'|)}$ over the variables X' and new variables Y. By the soundness of $\mathcal{A}$, starting with an assignment $b \notin SAT_\delta(\varphi')$ for X', no matter how one assigns the remaining Y variables, at most ε fraction of the ψ'_is accept.

It remains to add tests comparing between the assignment b for X' to the assignment a for X. In order to be able to do this via circuits that make only few queries, X' must encode X in a *"locally checkable"* manner. Thus, the heart of our proof is the encoding E, whose properties are formalized next.

Lemma 4.3 *Let $\bar{\delta}$ be a universal constant. For every $\delta_1 < \delta_2 < \bar{\delta}$, there exists a constant $c = O(\frac{\delta_2}{\delta_1}) \geq 1$ and an encoding $E : \{0,1\}^n \to \{0,1\}^{cn}$ that is polynomial-time computable, such that*

1. *If $a_1, a_2 \in \{0,1\}^n$, $\mathsf{dist}(a_1, a_2) > \delta_1$ then $\mathsf{dist}(E(a_1), E(a_2)) > \delta_2$.*
2. *There is a linear time circuit that computes $E^{-1}(b)$, if b is in the image of E, and otherwise rejects.*
3. *There is a polynomial-time constructible collection of n circuits each of size $O(c) = O(\frac{\delta_2}{\delta_1})$ such that given $a \in \{0,1\}^n$ and $b \in \{0,1\}^{cn}$ the following hold:*
   - *If $b = E(a)$ all of the circuits accept.*
   - *At least a $\mathsf{dist}(b, E(a))$ fraction of the circuits reject.*

Before proving Lemma 4.3, let us complete the description of $\mathcal{A}'$, and prove its properties. Let E be as in the lemma, choosing $\delta_1 = \delta'$ and $\delta_2 = 2\delta$.

As a first step, $\mathcal{A}'$ will compute φ' from φ, and generate $\psi'_1, \ldots, \psi'_{\mathsf{R}(|\varphi'|)}$ which are the outcome of running $\mathcal{A}$ on φ'. Recall that φ' is a circuit over variables X' that satisfies $SAT(\varphi') = E(SAT(\varphi))$. Due to the second item in Lemma 4.3, the size of φ', denoted M, is larger than the size of φ by a multiplicative factor $O(\frac{\delta}{\delta'})$. The number of circuits output by $\mathcal{A}$ is $\mathsf{R}(M)$, their size is $\mathsf{s}(M)$ and they read $\mathsf{q}(M)$ variables.

In addition, let $\{\mathsf{compare}_r\}_r$ be the collection of at most n circuits guaranteed by the third item of Lemma 4.3. We amplify the rejection probability of $\{\mathsf{compare}_r\}_r$ by derandomized serial error-reduction. We define a set of tests by applying Corollary 2.5 on the sequence $\{\mathsf{compare}_r\}$ with parameters $\mu = \delta$ and $\beta = \varepsilon$. Denote the new tests by $\{\mathsf{compare}'_1, \ldots, \mathsf{compare}'_{M_1}\}$. By Corollary 2.5, $M_1 = poly(1/\varepsilon)n$ and each $\mathsf{compare}'_i$ is the AND of $d = O(\frac{1}{\delta}\log\frac{1}{\varepsilon})$ $\mathsf{compare}$ tests. Therefore the size, denoted by Q, of each $\mathsf{compare}'_i$ satisfies $Q = O(c \cdot d) = O(\frac{1}{\delta'}\log\frac{1}{\varepsilon})$ (which also upper-bounds the number of X and X' variables these circuits read).

The final output circuits of $\mathcal{A}'$ will be $\{(\psi_i \wedge \mathsf{compare}'_i)\}_i$, where by repetition we may assume an equal number of at most $poly(1/\varepsilon) \cdot \mathsf{R}(M)$ circuits of each type.

It is easy to check that $\mathcal{A}'$ has the claimed parameters, and it remains to prove the completeness and soundness of $\mathcal{A}'$.

Completeness is immediate: Given some $a \in SAT(\varphi)$, extend it to X'-variables by letting $b : X' \to \{0,1\}$ be defined by $b \stackrel{def}{=} E(a)$. The rest follows from the completeness of $\mathcal{A}$.

For soundness, one needs to show that given any assignment for X that is δ'-far from a satisfying assignment, no assignment for the remaining variables $(X' \cup Y)$ can cause more than ε of the output circuits to accept. So let $a \notin SAT_{\delta'}(\varphi)$. Let $b : X' \to \{0,1\}$ and $c : Y \to \Sigma$ be arbitrary. There are two cases:

- If $\mathsf{dist}(E(a), b) \geq \delta$ then by Lemma 4.3 at least δ fraction of the circuits $\{\mathsf{compare}_r\}_r$ reject, so by construction and according to Corollary 2.5, at most ε of the circuits in $\{\mathsf{compare}'_i\}$ accept.
- Otherwise, $\mathsf{dist}(E(a), b) < \delta$. Then, using $\mathsf{dist}(a, SAT(\varphi)) > \delta'$, the first item in Lemma 4.3 implies $\mathsf{dist}(E(a), SAT(\varphi')) > 2\delta$. So by the triangle inequality, $\mathsf{dist}(b, SAT(\varphi')) > \delta$. The soundness of $\mathcal{A}$ implies that no matter what the assignment for Y, at most ε of the circuits ψ'_i accept.

This completes the proof of soundness, since in both cases, $a \notin SAT_{\delta'}(\varphi)$ allows at most ε fraction of the final circuits to accept. Thus, assuming Lemma 4.3 we have proved Lemma 4.1.

■ **Proof: (of Lemma 4.3)** Clearly, item 1 can be obtained using any error-correcting code (though the length of the code will be somewhat larger than in the lemma) . However, this would fail to give item 3, as given strings a and b, it is not clear how to check that $\mathsf{dist}(E(a), b)$ is small with few queries. Instead, we will use an encoding that is not an error correcting code in the standard sense, but does have the desired amplification property.

Let $k = O(\frac{\delta_2}{\delta_1})$. We first describe an encoding over non-binary alphabet $\Sigma = \{0,1\}^k$, denoted $E_0 : \{0,1\}^n \to \Sigma^N$. To encode a string $a \in \{0,1\}^n$, simply write its restriction on all possible k-bit substrings (so there are $N = n^k$ possible such restrictions). It is easy to see that this encoding achieves distance amplification. Indeed, two strings $a_1, a_2 \in \{0,1\}^n$ with $\mathsf{dist}(a_1, a_2) > \delta_1$, will differ on $1 - (1 - \delta_1)^k \approx k\delta_1 > \delta_2$ fraction of the symbols.

Moreover, it is unnecessary to take all $N = n^k$ restrictions. With judicious choice of k-tuples, distance amplification will hold even with $N = n$ restrictions. Let X_k be an efficiently constructible hitting set of k-tuples of $[n]$, such that every subset of X of size $\geq \delta_1 n$ intersects at least a $c_1\delta_2$ fraction of the k-tuples in X_k (the constant $c_1 > 1$ will be chosen below). Lemma 2.3 guarantees such a set X_k with $|X_k| = n$. This gives an encoding $E_1 : \{0,1\}^n \to \Sigma^{|X_k|}$. The choice of X_k guarantees that if $\mathsf{dist}(a, a') > \delta_1$ then $\mathsf{dist}(E_1(a), E_1(a')) > c_1\delta_2$, because at least $c_1\delta_2$ fraction of the tuples in X_k 'hit' the set $\{\, i \mid a_i \neq a'_i\}$. Note that we are assuming (when applying Lemma 2.3) $c_1\delta_2 < 1/2$, which can be ensured by setting $\bar{\delta}$ to be a small enough constant.

The encoding E_1 is almost what we want, except it is not binary. Thus, we concatenate E_1 (in the coding-theoretical sense) with the error-correcting code $e_k : \{0,1\}^k \to \{0,1\}^{c_2 \cdot k}$, given by Lemma 2.1. We now fix $c_1 > 0$ to be a constant such that e_k has relative distance $\geq 1/c_1$. Recall that e_k also has constant rate (i.e., c_2 is a constant), and linear size "codeword decoding" as defined in Lemma 2.1. Let $E \stackrel{def}{=} E_1 \circ e_k$ encode a string $a \in \{0,1\}^n$ by first computing $E_1(a)$ and then encoding each symbol of $E_1(a)$ using e_k. The distance of e_k being at least $1/c_1$ guarantees that

$$\mathsf{dist}(a, a') > \delta_1 \quad \Rightarrow \quad \mathsf{dist}(E_1(a), E_1(a')) > c_1\delta_2 \quad \Rightarrow \quad \mathsf{dist}(E(a), E(a')) > \delta_2,$$

which establishes item 1 of the lemma. For item 2, we use the fact that codeword decoding e_k

needs circuits of size $O(k)$, and the fact that verifying that a string is an output of E_1 and then inverting E_1 on it is easy to do by a linear size circuit.

For item 3, we consider the following randomized test (which translates in the natural way to the required n circuits). The input is $a \in \{0,1\}^n$ and $b \in \{0,1\}^{c_2 kn}$, and we wish to verify that $b = E(a)$.

- Select a random k-tuple in X_k, and denote it by $(i_1, \ldots, i_k)$. Read $a_{i_1}, \ldots, a_{i_k}$.
- Let $j_1, \ldots j_{c_2 k}$ be the indices such that $b|_{j_1,\ldots,j_{c_1 k}}$ supposedly equals $e_k(a|_{i_1,\ldots,i_k})$. Read $b_{j_1}, \ldots, b_{j_{c_1 k}}$.
- Accept iff $e_k(a_{i_1}, \ldots, a_{i_k}) = b_{j_1}, \ldots, b_{j_{c_1 k}}$ (this is performed by first applying the circuit for codeword decoding of e_k on $b_{j_1}, \ldots, b_{j_{c_1 k}}$ and then comparing the result with $a_{i_1}, \ldots, a_{i_k}$).

The test requires $\log |X_k| = \log n$ random bits, and reads $k + c_1 k = O(\frac{\delta_1}{\delta_2})$ bits. If $b = E(a)$, the test clearly accepts. Otherwise, denote $\beta = \mathsf{dist}(b, E(a))$. By construction of E_1, for at least β fraction of the tuples $(i_1, \ldots, i_k) \in X_k$: $e_k(a_{i_1}, \ldots, a_{i_k}) \neq (b_{j_1}, \ldots, b_{j_{c_1 k}})$, so the test rejects with probability at least β. The test for any particular fixing of the $\log n$ random bits can be implemented by a circuit of size $O(k) = O(\frac{\delta_1}{\delta_2})$. ■

Remark 4.4 *In the construction provided in Lemma 4.1 of the new distance-reduced assignment tester $\mathcal{A}'$ for inputs of size n, we only require $\mathcal{A}$ to be well-defined on inputs of size at most $M = O(\frac{\delta}{\delta'} n)$. This will be important for our inductive constructions, where $\mathcal{A}$ has only been defined for inputs of up to a certain size.*

Remark 4.5 *Lemma 4.1 will only be used in this work, for constant δ. However, we will ignore for the simplicity of presentation, the requirement that δ is smaller than some fixed constant $\bar{\delta}$. It is not hard to ensure that the lemma is only applied with sufficiently small δ.*

4.2 Serial Error-Reduction: $\varepsilon \longrightarrow \varepsilon'$

Let ε be the error probability of an assignment tester $\mathcal{A}$, i.e., the fraction of output circuits that erroneously accept a far-from-satisfying assignment. The naive way of reducing ε is by serial repetition. Namely, taking ANDs of k uniformly selected output circuits of $\mathcal{A}$ (as the output circuits of the new assignment tester $\mathcal{A}'$). The set of variables for $\mathcal{A}'$ is exactly the same as for $\mathcal{A}$ (therefore the width of variables does not change by this transformation). However the number of variables read by each circuit increases by a factor k. This implies the following reduction:

Lemma 4.6 (Serial Error Reduction) *For any integer ℓ, given an assignment-tester $\mathcal{A}$ with parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$, we can construct a new assignment tester $\mathcal{A}'$ whose error probability is ε^ℓ. The other parameters of $\mathcal{A}'$ are: $\mathsf{R}'(\mathsf{n}) = (\mathsf{R}(\mathsf{n}))^\ell$, $\mathsf{s}' = O(\ell \mathsf{s}(\mathsf{n}))$, $\mathsf{q}' = O(\ell \mathsf{q}), \delta' = \delta$.*

The number of circuits R' that $\mathcal{A}'$ outputs can be decreased by standard derandomization techniques. Particularly, based on Corollary 2.5 we obtain the following reduction.

Lemma 4.7 (Derandomized Serial Error Reduction) *Given an assignment-tester $\mathcal{A}$ with parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$, we can construct a new assignment tester $\mathcal{A}'$ whose error probability is ε'. The other parameters of $\mathcal{A}'$ are $\mathsf{R}'(\mathsf{n}) = O(poly(1/\varepsilon')\mathsf{R}(\mathsf{n}))$, $\mathsf{s}' = O(\ell \mathsf{s}(\mathsf{n}))$, $\mathsf{q}' = O(\ell \mathsf{q}), \delta' = \delta$, where $\ell = \log(1/\varepsilon')/(1-\varepsilon)$.*

The main disadvantage of the serial error reduction (in both versions) is that it reduces ε at the expense of increasing the number of variables q that the circuits read. When error reduction will be used in our constructions, increasing q will not be acceptable. Therefore, we next give a new method for decreasing q back, by aggregation (i.e., by "consistently" reading some ℓ variables at once "in parallel").

4.3 Error Reduction via Parallelization/Aggregation

A central ingredient in the proof of the PCP-theorem is the notion of aggregation (alternatively referred to as parallelization). Namely, starting with an assignment-tester that reads q variables we want to construct a new assignment-tester that reads fewer $\mathsf{q}' < \mathsf{q}$ variables and is otherwise comparable to the old assignment-tester (though typically the width of the new variables will be larger to compensate for the fewer number that are being read). One motivation for aggregation in our context is error reduction, as discussed above. (Indeed, Theorem 4.8 below reduces both q and ε.) The original proof of the PCP theorem [AS98, ALM+98] gives a very powerful aggregation method based on low-degree curves. The proofs in this paper will also require an aggregation theorem. However, our setting is significantly simpler as we only need to reduce the number of variables read by the assignment-tester from one *constant* q to a smaller constant q'. A "combinatorial" aggregation method for this setting can possibly be based on parallel repetition theorems. In particular, we could rely on the work of Feige and Kilian [FK94], as we do not require the full exponential decrease provided by [Raz98]. We remark that since we require the result to be an assignment-tester (rather than a verifier), some more details may be involved when applying [FK94, Raz98] in this context.

In this section, we present a simple alternative to both solutions. An advantage of our aggregation compared with the one based on curves, is that it can produce variables of *constant width* (rather than logarithmic width). This feature is vital for our proofs. Compared to parallel repetition theorems, our setting is easier since we can afford more than two queries (but shall make a constant number of queries).

Note that the aggregation method of Theorem 4.8 below, increases the distance parameter δ. This is the reason that the constructions of this paper require a method for reducing δ, as indeed given by the transformation of Lemma 4.1.

Theorem 4.8 (Aggregation) *Let $\mathcal{A}$ be an assignment-tester with parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$. For every $\varepsilon' > 0$, one can construct a new assignment tester with parameters $(poly(1/\varepsilon') \cdot \mathsf{R}^\ell, \ell \cdot \mathsf{s} \cdot Q, Q, 2\delta, \varepsilon')$, for $Q = O(\frac{1}{\delta}\log(\frac{1}{\varepsilon'}))$, and $\ell = poly(\frac{\mathsf{q}}{1-\varepsilon})$.*

Before turning to the actual proof, let us sketch the idea. First, it is easy to reduce q to a constant ($\mathsf{q} = 3$ in our case), at the price of increasing the error. This is done by adding auxiliary variables which encode the entire input of a circuit, and then replacing that circuit by q circuits that compare the new 'big' variable to one of the q original variables (in fact it will be convenient for us to compare to one X-variable and also to one Y-variable, implying query complexity three rather than two). This (standard) transformation will be described in detail later.

Next, the error probability can easily be reduced again by serial repetition, i.e., by taking ANDs of multiple circuits. However, the resulting circuits access more variables than before (i.e., by these two steps, q just got larger). To avoid this, it is natural to use *parallel repetition*. Namely, we introduce new variables $\overline{X} \equiv X^\ell$ that are supposed to be ℓ-tuples of the original variables X. Now, the effect of serial repetition can be emulated with only *few* accesses to the

new (wider) variables. The main obstacle this approach must overcome is that of consistency: Suppose $x \in X$ occurs in two tuples, $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$. Given an assignment $F : \overline{X} \to \Sigma^\ell$, it is quite possible that the value x receives in $F(\bar{\mathbf{x}})$ differs from that in $F(\bar{\mathbf{x}}')$. If we were assured that there are no (or few) inconsistencies of this sort, then the analysis of parallel repetition would be as easy as that of serial repetition. We therefore address this issue by a consistency test (described in Figure 3) such that passing it with high probability guarantees that most tuples are 'mostly' consistent with the plurality function of F, defined shortly (after specifying some notation and conventions).

For the sake of generality, and in order to simplify our analyses, we will define both the plurality function and the consistency test with respect to an arbitrary distribution $\mathcal{D}$ on X. Also for simplicity, we assume that we will always encounter assignments $F : \overline{X} \to \Sigma^\ell$ that are **rotation consistent**. Namely, that if $\bar{\mathbf{x}}'$ is obtained from $\bar{\mathbf{x}}$ using some cyclic shift of its ℓ components, then $F(\bar{\mathbf{x}}')$ can be obtained from $F(\bar{\mathbf{x}})$ in the same way. This assumption slightly simplifies the consistency test and is easy to achieve in our setting using a trivial folding argument (simply let F be specified by the assignment for the various *subsets* of X of at most ℓ elements rather than by assignments to *ordered ℓ-tuples of elements*). Finally, for any ℓ tuple $\bar{\mathbf{x}}$, let $\bar{\mathbf{x}}_i$ denote its ith component.

Definition 4.9 (Plurality) *Let X and Σ be finite sets, let $\ell \geq 1$ be an integer, and set $\overline{X} \equiv X^\ell$. Let $\mathcal{D}$ be an arbitrary probability distribution over X. Let $F : \overline{X} \to \Sigma^\ell$. Define the* plurality *function of F with respect to $\mathcal{D}$, denoted $f_{F,\mathcal{D}} : X \to \Sigma$ as follows. For every $x \in X$, let $f_{F,\mathcal{D}}(x)$ be the value that is assigned to x most frequently by F (with respect to the distribution $\mathcal{D}^\ell$ on $\overline{X}$). Formally,*

$$\forall x \in X, \quad f_{F,\mathcal{D}}(x) \stackrel{def}{=} \max \arg_{a \in \Sigma} \left\{ \Pr_{\bar{\mathbf{x}} \in \mathcal{D}^\ell, i \in [\ell]} [F(\bar{\mathbf{x}})_i = a \mid \bar{\mathbf{x}}_i = x] \right\}.$$

In case $\mathcal{D}$ is the uniform distribution over X, we denote the plurality function as f_F (i.e., we omit $\mathcal{D}$ from the notation).

In Figure 3 we describe the consistency test. Essentially the same test was first studied by Goldreich and Safra [GS97], who proved that (for uniform $\mathcal{D}$) the test establishes consistency with some f. Their proof used a different "two-stage" plurality function f and was established via reduction to another (derandomized) test. In Appendix A we give a direct proof for the same test, summarized by the following theorem,

Theorem 4.10 (Consistency) *Let $F : \overline{X} \to \Sigma^\ell$ and $\mathcal{D}$ an arbitrary probability distribution over X. Let $f = f_{F,\mathcal{D}} : X \to \Sigma$ be the plurality function of F. Let T be the test described in Figure 3.*

- *If for all $\bar{\mathbf{x}} = (x_1, \ldots, x_\ell) \in \overline{X}$, it holds that $F(x_1, \ldots, x_\ell) = (f(x_1), \ldots, f(x_\ell))$, then T always accepts.*
- *Call $\bar{\mathbf{x}} = (x_1, \ldots, x_\ell) \in \overline{X}$* bad *if $F(\bar{\mathbf{x}})$ disagrees with $(f(x_1), \ldots, f(x_\ell))$ on more than $\sqrt[3]{\ell}$ entries. There exists a constant c such that for every $\gamma > 0$,*

$$\Pr_{\bar{\mathbf{x}} \in \mathcal{D}^\ell}[\bar{\mathbf{x}} \text{ bad}] > \gamma \quad \Longrightarrow \quad T \text{ rejects with probability at least } \gamma/c\,.$$

1. Select a random $\bar{\mathbf{x}} = (x_1, \ldots, x_\ell) \in \mathcal{D}^\ell$.
2. Select a random $\bar{\mathbf{x}}' = (x'_1, \ldots, x'_\ell) \in \overline{X}$ as follows: for each $j \in [\ell]$, $x'_j = x_j$ with probability $\alpha \stackrel{def}{=} \ell^{-1/3}$, otherwise x'_j is selected independently according to $\mathcal{D}$.
3. Accept only if $F(\bar{\mathbf{x}})$ agrees with $F(\bar{\mathbf{x}}')$ on *all* common variables, otherwise reject.

Figure 3: Consistency Test for $F : \overline{X} \to \Sigma^\ell$, any $|X|, |\Sigma| < \infty$ and any probability distribution $\mathcal{D}$ over X. In case F is not guaranteed to be rotation consistent then the test is revised slightly: the query $\bar{\mathbf{x}}'$ is also rotated cyclicly by a random shift in $[\ell]$.

Proof: (of Theorem 4.8) Our approach will be as follows. We first transform the circuits generated by $\mathcal{A}$ to circuits that each depend on 3 variables. This is a simple (almost standard) transformation that replaces every circuit reading q variables by several circuits each reading three variables (over a larger domain). This causes the error probability to increase. Next, we perform ℓ-parallel repetition: instead of reading 3ℓ variables as in the serial repetition, we introduce new variables that are ℓ-tuples of the previous ones, and read only three of these. We ensure soundness of this step by adding a separate consistency test. The number of circuits produced by the assignment tester will be bounded by $\mathsf{R}^{O(\ell \log \ell)}$. By redefining ℓ to be a slightly larger polynomial in $\frac{\mathsf{q}}{1-\varepsilon}$, the dependence on ℓ becomes as claimed.

Let $\mathcal{A}$ be an assignment-tester with parameters $(\mathsf{R}, \mathsf{s}, \mathsf{q}, \delta, \varepsilon)$. Let $\gamma = 1 - \varepsilon$ be the detection probability of $\mathcal{A}$. We first fix $\gamma' = \delta/24c$, where c is the absolute constant from Theorem 4.10, and prove that one can construct a new assignment tester with parameters $(\mathsf{R}^{O(\ell \log \ell)}, O(\ell \cdot \mathsf{s}), O(1), 2\delta, \gamma')$, for $\varepsilon' = 1 - \gamma'$ and $\ell = O((\frac{\mathsf{q}}{\gamma})^{3/2})$. The theorem will then follow for an arbitrarily small ε' as a simple corollary of Lemma 4.7 (i.e., by serial error reduction). The proof will follow the two steps mentioned above.

Step 1: Getting to three queries

Let φ be a circuit over variables X. Run $\mathcal{A}$ on input φ, generating output circuits $\psi_1, \ldots, \psi_{\mathsf{R}}$ over X, Y. Define new variables $Z = \{z_1, \ldots, z_{\mathsf{R}}\}$, one z_i per ψ_i. The variable z_i will assume values that are supposedly the complete input to ψ_i. Notation: we denote by $\Sigma_X = \{0,1\}, \Sigma_Y, \Sigma_Z$ the set of values assumed by the variables in X, Y, Z respectively. Thus, $\Sigma_Z \subseteq (\Sigma_X \cup \Sigma_Y)^{\mathsf{q}}$. We also denote by $X(z_i)$ (resp. $Y(z_i)$) the set of X- (resp. Y-) variables accessed by the corresponding circuit, ψ_i. Assume wlog that these sets are non-empty for all i. Clearly, $|X(z) \cup Y(z)| \leq \mathsf{q}$ for all z.

By introducing the Z variables, we can get a system of circuits each reading 3 (rather than q) variables, but with a lower detection probability. This is a variant on a standard "transformation to two-provers" [FRS94], and can be done for example as follows. Replace each ψ_i with circuits $\psi_i^{(1)}, \ldots, \psi_i^{(\mathsf{q})}$ that each read z_i, and also one $x \in X(z_i)$, and one $y \in Y(z_i)$, and then check that the value of z_i would have satisfied ψ_i, and that it is consistent with the values of x and y. (Each one of the $x \in X(z_i)$ and $y \in Y(z_i)$ is read by at least one of these circuits $\psi_i^{(\mathsf{q})}$.) We remark that each new circuit $\psi_i^{(j)}$ could have read two variables (as is more standard) rather than three: Instead of reading both an X-variable *and* a Y-variable (in addition to the Z-variable), it could

have read either an X *or* a Y variable. However, looking ahead, making three queries (each into a different set of variables) will be convenient for repetition. It will allow us, more naturally, to use three separate tables for ℓ-tuples of Z, Y, and X variables. This way we will avoid having to deal with tuples mixing both X and Y variables.

Clearly if $\sigma|_{X\cup Y}$ satisfies all ψ_i, it can be extended to Z such that all of the $\psi_i^{(j)}$ accept. Also,

Proposition 4.11 *Let $\sigma : X \cup Y \cup Z \to \Sigma_X \cup \Sigma_Y \cup \Sigma_Z$. If $\sigma|_X$ is δ-far from satisfying φ, then,*

$$\Pr_{i,j}\left[\psi_i^{(j)} \text{ rejects}\right] \geq \gamma/\mathsf{q}\,.$$

Proof: By the soundness of $\mathcal{A}$, at least γ fraction of the ψ_is reject the assignment $\sigma|_{X\cup Y}$. No matter how one assigns the Z variables, on these ψ_is either $\sigma(z_i) \notin SAT(\psi_i)$ or there is an inconsistency between the value of z_i and at least one one of the q variables accessed by ψ_i. We detect this inconsistency with probability at least $\frac{1}{\mathsf{q}}$. ■

Step 2: Parallelization

We now define new variables that are tuples of the variables X, Y, Z above. Let $\ell = O((\frac{\mathsf{q}}{\gamma})^{3/2})$. For every possible ℓ-tuple of X-variables, we have a new variable $\bar{\mathbf{x}} = (x_1, \ldots, x_\ell) \in \overline{X}$, so that $\overline{X} \equiv X^\ell$. Similarly we define $\overline{Y} \equiv Y^\ell$ and $\overline{Z} \equiv Z^\ell$.[6]

We now define three types of building-block circuits. The first will perform ℓ-parallel repetition of the circuits $\{\psi_i^{(j)}\}$ in order to reduce their error. The second will facilitate the analysis of the parallel repetition by performing a consistency test of the assignment to the ℓ tuples (as in Figure 3). The first two types only test the new variables that supposedly represent ℓ-tuples of original variables. We therefore still need to compare the assignment for the ℓ-tuples to the assignment for the original X-variables.

Parallel-Repetition Circuits. For every choice of an ℓ-tuple of circuits $\psi_{i_1}^{(j_1)}, \ldots, \psi_{i_\ell}^{(j_\ell)}$ in $\{\psi_i^{(j)}\}_{i\in[R],j\in[\mathsf{q}]}$ we will have one circuit simulating their ℓ-wise AND. For every $1 \leq t \leq \ell$ let x_t (resp. y_t, z_t) be the single x- (resp. y-, z-) variable accessed by $\psi_{i_t}^{(j_t)}$. The circuit will access $\bar{\mathbf{z}} = (z_1, \ldots, z_\ell) \in \overline{Z}$, $\bar{\mathbf{x}} = (x_1, \ldots, x_\ell)$ and $\bar{\mathbf{y}} = (y_1, \ldots, y_\ell)$ and accept if on every coordinate $1 \leq t \leq \ell$: z_t satisfies ψ_{i_t} and is consistent with x_t and y_t (recall that the assignment for each z_t is interpreted as an assignment for all of the variables of ψ_{i_t}, and in particular it specifies values for x_t and y_t). Denote these circuits by $C_1^1, \ldots, C_{R_1}^1$ with $R_1 = (\mathsf{q}\mathsf{R})^\ell \leq R^{O(\ell)}$ (since $\mathsf{q} \ll \mathsf{R}$). Let $\mathsf{C}_1 = \{C_1^1, \ldots, C_{R_1}^1\}$.

Consistency Circuits. Let $\mathcal{D}_x$ be the distribution on X defined by having $\Pr_{x\in\mathcal{D}_x}[x]$ equal the probability that a uniformly random $\psi_i^{(j)}$ circuit reads x. Define $\mathcal{D}_y, \mathcal{D}_z$ similarly. For all possible random choices of $\bar{\mathbf{x}}, \bar{\mathbf{x}}'$ as described in the consistency test in Figure 3, with $\mathcal{D} = \mathcal{D}_x$, we have a consistency circuit for the corresponding test. Likewise for all choices of $\bar{\mathbf{y}}, \bar{\mathbf{y}}'$ and of $\bar{\mathbf{z}}, \bar{\mathbf{z}}'$. Let $\mathsf{C}_2 = \{C_1^2, \ldots, C_{R_2}^2\}$ be the sequence of these circuits. The number of random choices is bounded by the number of pairs of circuit $\psi_i^{(j)}$ and variable x (or y) which is $\leq \mathsf{q}\mathsf{R}$, raised to

[6] As discussed above, we ensure that the assignment to these ℓ-tuples is *rotation consistent*, by a simple folding argument. Instead of requiring an assignment to ℓ-tuples we ask for an assignment to subsets of cardinality at most ℓ. This naturally defines a rotation consistent assignment to the ℓ-tuples.

the power ℓ, and then at most squared. So, $R_2 \leq 3\,(\mathsf{qR})^{2\ell} \leq \mathsf{R}^{O(\ell)}$ (note that we have exactly the same number of circuits testing the X-variables, the Y-variables, and the Z-variables).

Comparison Circuits. These ensure consistency between $\overline{X}$ and X. For each $x \in X$ and $\bar{\mathbf{x}} \in \overline{X}$ that contains it, we have a comparison circuit, comparing the value of x, with the value given to it in the tuple $\bar{\mathbf{x}}$. Denote the resulting circuits $\mathsf{C}_3 = \{C_1^3, \ldots, C_{R_3}^3\}$, then $R_3 = \ell\mathsf{R}^\ell = \mathsf{R}^{O(\ell \log \ell)}$ such circuits.

Final Circuits. For simplicity, we assume $R_1 = R_2 = R_3 = R^{O(\ell \log \ell)}$ (otherwise, equality can be obtained by duplication). The final circuits output by our assignment tester will be the AND of the i-th C_1 circuit with the i-th C_2 circuit and with the i-th C_3 circuit. That is the i-th output circuit is $C_i^1 \wedge C_i^2 \wedge C_i^3$. This guarantees that if ε' of the final circuits accept, then ε' of the C_1 circuits accept as do ε' of the C_2 circuits and ε' of the C_3 circuits. Observe that these circuits are over variables $X \cup \overline{X} \cup \overline{Y} \cup \overline{Z}$.

This completes the description of the new assignment tester. Before we prove completeness and soundness, let us examine its parameters. The number of output circuits is indeed $R^{O(\ell \log \ell)}$ Each output circuit queries $O(1)$ variables. The maximum size of an output circuit is $\leq O(\ell \cdot \mathsf{s})$.

Completeness is immediate: Given an assignment to X that satisfies φ, it can be extended (due to the completeness of $\mathcal{A}$) to the Y variables so that all ψ_i are satisfied. This naturally extends to an assignment for Z, and then for $\overline{X}, \overline{Y}, \overline{Z}$.

Lemma 4.12 (Soundness) *Given an assignment σ for $X \cup \overline{X} \cup \overline{Y} \cup \overline{Z}$, if $\sigma|_X$ is 2δ-far from satisfying φ, then at least γ' of the output circuits reject.*

Proof: Let $F_x \stackrel{def}{=} \sigma|_{\overline{X}} : \overline{X} \to \Sigma_X$ be the restriction of σ to $\overline{X}$, and similarly, $F_y \stackrel{def}{=} \sigma|_{\overline{Y}}$ and $F_z \stackrel{def}{=} \sigma|_{\overline{Z}}$. Recall $\mathcal{D}_x$ (respectively, $\mathcal{D}_y, \mathcal{D}_z$) was the distribution according to which the consistency test of $\overline{X}$ (respectively: $\overline{Y}, \overline{Z}$) was performed. Let $f_x = f_{F_x, \mathcal{D}_x}$ and $f_y = f_{F_y, \mathcal{D}_y}$ and $f_z = f_{F_z, \mathcal{D}_z}$ be the plurality functions of F_x, F_y, F_z respectively, as in Definition 4.9. In the following proof the reader may wish to regard $\mathcal{D}_x, \mathcal{D}_y, \mathcal{D}_z$ as being uniform, although in general this need not be the case. We will however require $\mathcal{D}_x$ to be uniform, so let us explain why this can be assumed with no loss of generality. The distribution $\mathcal{D}_x$ would be uniform if the sequence of $\psi_i^{(j)}$ circuits access each X-variable the same number of times. To enforce this condition, we slightly modify the ψ_i circuits. We introduce a new set of variables X' that is supposed to be the exact copy of X, and apply the ψ_i circuits on the assignment to X' (instead of the assignment to X). In addition, we add consistency checks to test that the assignments to X and to X' are δ-close, by checking equality between a random variable in X and its copy in X'. It is easy to argue that the new circuits still define an assignment tester (with a small increase in the distance parameter but otherwise almost identical parameters to the original one, $\mathcal{A}$). In addition, since the consistency checks between X and X' access each X-variable the same number of times, we get the additional desired property.

Recall that $\sigma|_X$ is 2δ-far from satisfying φ. We prove that either γ' of the circuits in C_3 reject, or γ' of the circuits in C_2 reject, or γ' of the circuits in C_1 reject.

Proposition 4.13 *If $\mathsf{dist}_X(f_x, SAT(\varphi)) \leq \delta$, then γ' of the circuits in C_3 reject.*

Proof: If $\mathsf{dist}_X(f_x, SAT(\varphi)) \leq \delta$, then since $\mathsf{dist}_X(\sigma|_X, SAT(\varphi)) > 2\delta$, the triangle inequality implies $\mathsf{dist}_X(f_x, \sigma|_X) > \delta$. Let $x \in X$ be such that $\sigma(x) \neq f_x(x)$. By definition of plurality (and

since $\mathcal{D}_x$ is the uniform distribution!), $f_x = f_{F_x}(x)$ is defined to be the value that is assigned most often to x, among all tuples that contain x. Thus, assuming $f_x \neq \sigma(x)$, the value of $\sigma(x)$ must occur in no more than half of the tuples containing x. Thus, at least half of the comparison circuits (in C_3) that access x reject, altogether $\frac{\delta}{2} > \gamma'$ of all of the comparison circuits reject. ■

Recall that an ℓ-tuple $\bar{\mathbf{x}} \in \overline{X}$ is *bad* with respect to F_x, if $F_x(\bar{\mathbf{x}})$ disagrees with $(f_x(x_1), \ldots, f_x(x_\ell))$ on more than $\sqrt[3]{\ell}$ locations. A similar definition applies to bad tuples in $\overline{Y}$ and in $\overline{Z}$.

Proposition 4.14 *If more than $\frac{1}{8}$ fraction of the tuples (in each of $\overline{X}, \overline{Y}$ or $\overline{Z}$) are bad, then at least γ' of the circuits in C_2 reject.*

Proof: Assume there are more than $\frac{1}{8}$ bad tuples, with respect to F_x, F_y or F_z then Theorem 4.10 guarantees that for some absolute constant c, the consistency test for the particular table will reject with probability $\geq \frac{1}{8c}$. Thus, at least $\frac{1}{24c} \geq \gamma'$ of the circuits in C_2 will reject. ■

Now, assume the conditions in both Propositions 4.13 and 4.14 fail to hold. This will enable us to argue that the circuits in C_1 are emulating the serial repetition of $\left\{\psi_i^{(j)}\right\}$, and to deduce that many of them reject. Failure of the condition of Proposition 4.13 means that $\mathsf{dist}_X(f_x, SAT(\varphi)) > \delta$. Together with Proposition 4.11, this implies that at least γ/q of the $\left\{\psi_i^{(j)}\right\}$ circuits reject the assignment $f_x \cup f_y \cup f_z$. Failure of the condition of Proposition 4.14 means that the variables $\overline{X}, \overline{Y}, \overline{Z}$ are *consistent* with the plurality assignments f_x, f_y, f_z. So for example reading the assignment for a tuple $(x_1, \ldots, x_\ell) \in \overline{X}$ "emulates" reading the assignment f_x in locations $x_1, \ldots, x_\ell$.

We now combine the above to deduce that at least γ' fraction of C_1 reject.

Indeed, choose ℓ random circuits $\psi_{i_1}^{(j_1)}, \ldots, \psi_{i_\ell}^{(j_\ell)} \in \left\{\psi_i^{(j)}\right\}$, and let $\overline{\psi} = (\psi_{i_1}^{(j_1)}, \ldots, \psi_{i_\ell}^{(j_\ell)})$. Under assignment $f_x \cup f_y \cup f_z$, we expect to see at least $\ell \cdot \frac{\gamma}{\mathsf{q}}$ of $\overline{\psi}$'s components reject. Moreover, ℓ is chosen exactly so that this expectation is sufficiently above $3\sqrt[3]{\ell}$, so most tuples $\overline{\psi}$ see more than $3\sqrt[3]{\ell}$ rejecting components. Indeed the following proposition follows from a standard tail inequality,

Proposition 4.15 *Assume the failure of the condition of Proposition 4.13. For some $\ell = O((\frac{\mathsf{q}}{\gamma})^{3/2})$,*

$$\Pr_{\overline{\psi}}\left[\overline{\psi} \text{ contains} \leq 3\sqrt[3]{\ell} \text{ circuits that reject } f_x \cup f_y \cup f_z\right] \leq 1/8\,.$$

We now examine which circuits in C_1 can possibly accept. A circuit $C \in \mathsf{C}_1$ corresponding to some $\overline{\psi} = (\psi_{i_1}^{(j_1)}, \ldots, \psi_{i_\ell}^{(j_\ell)})$ reads three variables $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}}$, and rejects unless:

1. At least one of $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ or $\bar{\mathbf{z}}$ is bad, or
2. $\overline{\psi}$ has at most $3\sqrt[3]{\ell}$ components that reject $f_x \cup f_y \cup f_z$.

Indeed, otherwise there must be at least one coordinate $1 \leq t \leq \ell$ for which $\psi_{i_t}^{(j_t)}$ rejects, and also the t-th coordinate of $F_x(\bar{\mathbf{x}})$ equals the plurality function $f_x(x_t)$, and similarly for $F_y(\bar{\mathbf{y}})$ and $F_z(\bar{\mathbf{z}})$, thus by definition C rejects.

Proposition 4.15 bounds the probability of the second item by $\frac{1}{8}$. The probability of the first item is bounded by the probability of hitting a bad tuple in either one of the tables F_x, F_y or F_z. Each of $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ or $\bar{\mathbf{z}}$ is a random (according to the corresponding distribution $\mathcal{D}_x, \mathcal{D}_y, \mathcal{D}_z$) entry in the corresponding table. Therefore, by the bound on the number of bad tuples (Proposition 4.14), the probability of the first item does not exceed $3 \cdot \frac{1}{8}$. Altogether, the

probability that $\overline{\psi}$ doesn't reject is bounded by $\frac{1}{8} + \frac{3}{8} = \frac{1}{2}$, which means that $\overline{\psi}$ rejects with probability at least $1/2 > \gamma'$.

In conclusion, at least γ' of the final output circuits reject, completing the proof of Lemma 4.12 (soundness). ■

Having established the soundness of the assignment-tester, Theorem 4.8 follows. ■

5 The PCP Theorem - An Alternate Proof

In this section we give a new proof for the PCP theorem [AS98, ALM+98]. Our proof involves composition, and a combination of the combinatorial transformations described in Section 4, applied on a 'weak' assignment-tester, that is assumed to be given:

Theorem 5.1 (Weak Tester) *For some constants $\beta > 0, c_1$ and q_β, there exists an assignment tester $\mathcal{A}_\beta$ with the following parameters*

- *Number of output circuits,* $\mathsf{R}(\mathsf{n}) = \mathsf{n}^{c_1}$,
- *Size of output circuits,* $\mathsf{s}(\mathsf{n}) = O(\mathsf{n}^{1-\beta})$,
- *Number of queries,* $\mathsf{q}(\mathsf{n}) = q_\beta$,
- *Error probability,* $\varepsilon(\mathsf{n}) = 0.1$,
- *Distance,* $\delta(\mathsf{n}) = 0.1$).

Such an assignment tester (and even stronger) can be constructed, for example, by relying on algebraic techniques already present in [FGL+91]. In particular, by taking a low-degree-extension with a *constant* number of dimensions (as opposed to the way it is usually invoked with a logarithmic number of dimensions) and then performing the [LFKN92] "sum-check" procedure (while relying on a *weak* low degree test). It is interesting to note that although an assignment-tester seems stronger than just a PCP verifier, all known constructions give the stronger object.

Our main theorem in this section is the following,

Theorem 1.1 (Formal Statement) *Assuming $\mathcal{A}_\beta$ as in Theorem 5.1, there exists an assignment tester $\mathcal{A}$ with parameters*

- *Number of output circuits,* $\mathsf{R}(\mathsf{n}) = poly(\mathsf{n})$,
- *Size of output circuits,* $\mathsf{s}(\mathsf{n}) = O(1)$,
- *Number of queries,* $\mathsf{q}(\mathsf{n}) = O(1)$,
- *Error probability,* $\varepsilon(\mathsf{n}) = 0.1$,
- *Distance,* $\delta(\mathsf{n}) = 0.1$.

The PCP theorem now follows as an immediate corollary,

Corollary 5.2 (PCP Theorem) *Given a set of Boolean constraints $\psi_1, \ldots, \psi_{\mathsf{R}}$ over Boolean variables Z, such that each read a constant number of variables, it is NP-hard to distinguish between the following two cases:*

1. *[Completeness:] There is an assignment to Z satisfying all of $\psi_1, \ldots, \psi_{\mathsf{R}}$.*

2. *[Soundness:] Every assignment to Z satisfies at most 10% of $\psi_1, \ldots, \psi_{\mathsf{R}}$.*

Proof: We reduce from SAT. On input a SAT formula φ over variables X, feed it to the assignment tester algorithm $\mathcal{A}$ given by Theorem 1.1. The output is a list of $\mathsf{R}(|\varphi|)$ circuits $\psi_1, \ldots \psi_{\mathsf{R}}$ such that each read a constant number of bits from $Z = X \cup Y$.

If φ is satisfiable, say by an assignment $a : X \to \{0,1\}$, then there is some b such that together $a \cup b$ satisfy all of $\psi_1, \ldots, \psi_{\mathsf{R}}$. If φ is not satisfiable, at most $\varepsilon = 0.1$ fraction of $\{\psi_1, \ldots, \psi_{\mathsf{R}}\}$ can accept. Otherwise, the soundness condition of $\mathcal{A}$ implies that a is δ-close to a satisfying assignment for φ, and in particular it means that φ is satisfiable. ■

The naive approach for proving Theorem 1.1, the main theorem of this section, is to compose $\mathcal{A}_\beta$ with itself $\log \log n$ times. Starting with an input circuit of size n, the output circuits have size $\mathsf{n}^{1-\beta}$, $(\mathsf{n}^{1-\beta})^{1-\beta} = \mathsf{n}^{(1-\beta)^2}$, and so on $\mathsf{n}^{(1-\beta)^i}$ at step i. Therefore, the circuit size of the output circuits is constant for $i = O(\log \log n)$. While already achieving non-trivial parameters, this construction does not quite give the PCP Theorem. The reason has to do with the error probability, or its complement, the detection probability ($\gamma = 1 - \varepsilon$), which will be more convenient to work with in this section. The detection probability becomes $\gamma_1 \gamma_2 \cdots \gamma_i = (\gamma_\beta)^{\log \log n} = O(1/\log n)$, rather than constant as we would like. Our proof of Theorem 1.1 will have the same general structure as in the naive approach but with a more elaborate recursive step, as follows.

It may seem at first that there is an additional more inherent problem with the naive approach. After all, to compose an assignment tester with itself we need $\delta < 1/\mathsf{q}$ which seems impossible. However, recall that the number of queries q of an assignment tester can be easily made a small fixed constant (e.g. 3) without harming the distance parameter δ (see Proposition 4.11).

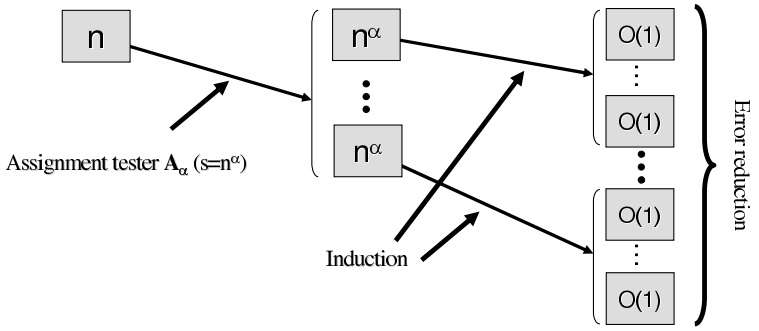

Figure 4: The inductive step. Error reduction encapsulates the additional parameter-fixing steps which compensate for the error-increase of the composition step.

Inductive step - quick overview (see also Figure 4). Our construction is bottom-up. We assume having constructed an assignment tester with the desired parameters that works for inputs of size smaller than n, and show how to extend it for inputs of size n. Therefore, by induction, we obtain the desired assignment-tester for all input-sizes n.

Our inductive step begins with composition of the black-box assignment tester with the one assumed by induction. This composition, as discussed above, increases the error of the assignment tester. To correct this we next perform error-reduction by 'parallelization' (as in

Theorem 4.8) which will take care of the error but will have two undesirable side effects: the distance parameter δ will increase and the size of the output circuits will increase as well. The next step will be to perform a distance reduction (as in Lemma 4.1). Finally, the last step will be to reduce back the size of the output circuits, to achieve the inductive hypothesis. This is done by composition (again) but this time with a constant size assignment tester (the inputs to this assignment tester will always be of some bounded constant size). We note that this second assignment tester (called $\mathcal{A}_0$ below) can be obtained, e.g., by instantiation of the black box tester (of Theorem 5.1) for a fixed input size.

Before proceeding to the formal proof, it will be convenient to strengthen a bit the parameters of $\mathcal{A}_\beta$ given in Theorem 5.1.

Corollary 5.3 *Assuming $\mathcal{A}_\beta$ as in Theorem 5.1, there exists a constant $\bar{q}$ such that for every $\alpha > 0$ there exists c_1 and an assignment tester $\mathcal{A}_\alpha$ with the following parameters $\mathsf{R}(\mathsf{n}) = \mathsf{n}^{c_1}$, $\mathsf{s}(\mathsf{n}) = O(\mathsf{n}^\alpha)$, $\mathsf{q}(\mathsf{n}) = \bar{q}$, $\delta(\mathsf{n}) = 0.1$, $\varepsilon(\mathsf{n}) = 0.1$.*

This strengthening follows by an appropriate sequence of transformations similar to those below. We defer the proof to Appendix B.

The rest of this section is devoted to proving Theorem 1.1 by following the outline above. We first describe the building blocks used to construct the recursive algorithm $\mathcal{A}$, and then formally describe $\mathcal{A}$, see also Figure 5.

5.1 Building-block testers and their parameters

As discussed above, we consider three testers in this construction. $\mathcal{A}$ itself (that is constructed recursively), $\mathcal{A}_\alpha$ which is essentially the assignment tester we assume as a black box, and finally $\mathcal{A}_0$ - the constant size assignment tester. It is important to note that there are various interdependencies between the parameters of these three testers, for example, to allow their composition (as Theorem 3.7 makes some restriction on parameters of the composed testers). The main purpose of the following definitions and discussion is to make explicit the dependencies between the various parameters, so as to make sure that we do not run into circular definitions. The reader may choose to ignore at first reading these subtleties and skip to the construction of $\mathcal{A}$ (provided in Figure 5).

First, $\mathcal{A}_\alpha$ (guaranteed by Corollary 5.3) has the parameters ($\mathsf{R}(\mathsf{n}) = \mathsf{n}^{c_1}$, $\mathsf{s}(\mathsf{n}) = O(\mathsf{n}^\alpha)$, $\mathsf{q}(\mathsf{n}) = \bar{q}$, $\delta(\mathsf{n}) = 0.1$, $\varepsilon(\mathsf{n}) = 0.1$). We will select α later and remember that c_1 is a constant and is the only parameter that depends on α. We stress that $\bar{q}$ is an absolute constant. Since $\mathcal{A}_\alpha$ will only be applied on input circuits of size larger than some n_0 (that can be made large enough), we can assume that $\mathsf{s}(\mathsf{n}) = \mathsf{n}^\alpha$. Denote the error probability and distance parameter by $\varepsilon_\alpha = \delta_\alpha = 0.1$ and the detection probability, by $\gamma_\alpha = 1 - \varepsilon_\alpha = 0.9$.

Regarding the two additional testers $\mathcal{A}$ and $\mathcal{A}_0$, we first set their distance parameters in a way that will allow the composition needed by the construction. Set $\delta = \frac{1}{c_2 \bar{q}}$, where c_2 is the constant from Theorem 3.7. Recall that composition of two assignment testers is allowed (Theorem 3.7) if the distance parameter δ of the second (inner) is related to the query parameter $\bar{q}$ of the first, by $\delta \leq \frac{1}{c_2 \bar{q}}$. In foresight, set $\mathsf{q}_1 = O(\frac{1}{\delta} \log \frac{1}{\varepsilon_\alpha})$ (the constant in the 'O' notation is an absolute constant that will be determined by the analysis below). Set $\delta_0 = \frac{1}{c_2 \mathsf{q}_1}$. As we can set $\mathsf{q}_1 \geq \bar{q}$ we obtain $\delta_0 \leq \delta$. We can now state the parameters of $\mathcal{A}_0$ (and prove its existence).

Let the input φ of $\mathcal{A}$ be a circuit of size n. If $|\varphi| \leq \mathsf{n}_0$, then $\mathcal{A}$ just runs $\mathcal{A}_0$. Otherwise, we describe the algorithm $\mathcal{A}$ via several 'intermediate' transformations $(\mathcal{A}_I \Rightarrow \mathcal{A}_{IIa} \Rightarrow \mathcal{A}_{IIb} \Rightarrow \mathcal{A}_{IIc} = \mathcal{A})$:

I. Initial Composition (major size reduction): Let $\mathcal{A}_I = \mathcal{A}_\alpha \circ \mathcal{A}$ be the result of composing $\mathcal{A}_\alpha$ with a recursive invocation of $\mathcal{A}$ using Theorem 3.7. That is, the recursive invocation refers to circuits of size at most $\mathsf{s}(\mathsf{n}) = O(\mathsf{n}^\alpha)$.

II. Error Reduction:

a) (Error and query reduction) Let $\mathcal{A}_{IIa}$ be the outcome of reducing the error parameter of $\mathcal{A}_I$ to ε_α using Theorem 4.8.

b) (Distance reduction) Let $\mathcal{A}_{IIb}$ be the outcome of reducing the distance parameter of $\mathcal{A}_{IIa}$ to δ using Lemma 4.1.

c) (Auxiliary size reduction) Let $\mathcal{A}_{IIc} = \mathcal{A}_{IIb} \circ \mathcal{A}_0$ be the composition of $\mathcal{A}_{IIb}$ with $\mathcal{A}_0$ again using Theorem 3.7, obtaining circuits of size s_0.

$\mathcal{A}$ will run $\mathcal{A}_{IIc}$ on φ.

Figure 5: The Assignment Tester $\mathcal{A}$

Proposition 5.4 *For any $\alpha_0 > 0$ and any n_0 which is large enough (as a function of α_0 and δ_0), there exist $\mathsf{R}_0, \mathsf{s}_0$ and q_0 with $\mathsf{s}_0 < (n_0)^{\alpha_0}$ and there exists an assignment tester $\mathcal{A}_0$ such that for input circuits of size $\leq \mathsf{n}_0$, $\mathcal{A}_0$ has parameters $(\mathsf{R}_0, \mathsf{s}_0, \mathsf{q}_0, \delta_0, \varepsilon_\alpha)$. (By definition $q_0 \leq s_0$.)*

Proof: Set $\alpha_0' = \alpha_0/3$ and let $\mathcal{A}_{\alpha_0'}$ be the assignment tester guaranteed by Corollary 5.3. $\mathcal{A}_0$ will be the result of applying the distance-reduction transformation (given by Lemma 4.1) to $\mathcal{A}_{\alpha_0'}$, reducing its distance parameter to δ_0. All we need to verify is the condition on s_0. By Corollary 5.3 and Lemma 4.1 we have that the size of the output circuits of $\mathcal{A}_0$ for input circuits of size $\leq \mathsf{n}_0$ is at most $\mathsf{s}_0 = O(\mathsf{n}_0/\delta_0)^{\alpha_0'} + O(\frac{1}{\delta_0} \log \frac{1}{\varepsilon_\alpha})$. By setting n_0 to be large enough we get that indeed $s_0 < (n_0)^{\alpha_0}$. ■

In the proof we will need to set α_0 to be a small enough constant, this in turn will set n_0 and also R_0 to be large enough constants.

5.2 The Recursive Construction

The following lemma restates Theorem 1.1, giving explicit parameters for $\mathcal{A}$ that will be obtained by induction. The constants s_0 and q_0 are as in Proposition 5.4, and $\delta = \frac{1}{c_2 \bar{q}}$ and $\gamma_\alpha = 0.9$.

Lemma 5.5 *There exists $C > 0$ and an assignment-tester $\mathcal{A}$ with the following parameters:* $\mathsf{R}(\mathsf{n}) \leq \mathsf{n}^C$, $\mathsf{s}(\mathsf{n}) = \mathsf{s}_0, \mathsf{q}(\mathsf{n}) = \mathsf{q}_0, \delta(\mathsf{n}) = \delta = \frac{1}{c_2 \bar{q}}, \varepsilon(\mathsf{n}) = 1 - (\gamma_\alpha)^2$.

Proof: We give a full description of $\mathcal{A}$, and prove (using induction) that $\mathcal{A}$ is well defined, and has the desired final parameters. A brief outline of $\mathcal{A}$ is given in Figure 5.

Let the input φ of $\mathcal{A}$ be a circuit of size n. If $|\varphi| \leq \mathsf{n}_0$, then $\mathcal{A}$ just runs $\mathcal{A}_0$. In this case, the base of the induction is established by Proposition 5.4. As long as the constant C is chosen to

be large enough so that $\mathsf{R}(\mathsf{n}_0) = \mathsf{n}_0^C \geq \mathsf{R}_0$), we have that the parameters of $\mathcal{A}_0$ (on input size at most n_0) are only better than required.

Assume by induction that Lemma 5.5 holds for all circuits of size smaller than n (in fact we will only use that it holds for circuits of size $\mathsf{s}(O(\mathsf{n})) = O(\mathsf{n}^\alpha)$). We verify that each of the following steps (I, IIa, IIb, IIc below) is well defined, and that we get the claimed parameters. Figure 6 may assist the reader in following the evolving parameters.

Name	Description	system-size (R)	circuit-size (s)	q	δ	γ
$\mathcal{A}_\alpha$	Black-box	n^{c_1}	n^α	$\bar{q}$	δ_α	γ_α
$\mathcal{A}$	Induction	n^C	s_0	q_0	δ	$(\gamma_\alpha)^2$
$\mathcal{A}_0$	Const. Size	R_0	s_0	q_0	δ_0	γ_α
$\mathcal{A}_I$	reduce s	$\mathsf{n}^{c_1} \cdot O(\mathsf{n}^\alpha)^C$	s_0	q_0	δ_α	$(\gamma_\alpha)^3$
$\mathcal{A}_{IIa}$	& raise γ reduce q	$\left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)}$	$\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}'_1$	q'_1	$2\delta_\alpha$	γ_α
$\mathcal{A}_{IIb}$	reduce δ	$\left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)}$	$\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}_1$	q_1	δ	γ_α
$\mathcal{A}_{IIc}$	reduce s	$\left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)}$	s_0	q_0	δ	$(\gamma_\alpha)^2$

Figure 6: Evolution of parameters during one inductive step. Recall that $\mathsf{q}_1 = O(\frac{1}{\delta}\log\frac{1}{\varepsilon_\alpha})$. In addition, $\mathsf{q}'_1 = O(\frac{1}{\delta_\alpha}\log\frac{1}{\varepsilon_\alpha})$ and $\ell = poly\left(\frac{\mathsf{q}_0}{\gamma_\alpha}\right)$. The main composition step (step I), reduces the circuit size to s_0 at the price of reducing the detection probability from $(\gamma_\alpha)^2$ to $(\gamma_\alpha)^3$. The error-reduction step (step IIa), increases the detection probability back (even above its final intended value). It is also crucial that the number of queries is also reduced, i.e. $\mathsf{q}'_1 < \mathsf{q}_0$. However, both s and δ increase. This is corrected in steps IIb and IIc.

I. $\mathcal{A}_I$ is the composition of $\mathcal{A}_\alpha$ with the recursive invocation of $\mathcal{A}$, using Theorem 3.7. This is well defined because $\mathcal{A}$ uses distance parameter $\delta \leq \frac{1}{c_2\bar{q}}$ and since $\mathcal{A}$ is defined (inductively) for all inputs of size smaller than $\mathsf{s}(\mathsf{n}) < \mathsf{n}$. Based on Theorem 3.7, we have that $\mathcal{A}_I = \mathcal{A}_\alpha \circ \mathcal{A}$ produces $\mathsf{n}^{c_1} \cdot (O(\mathsf{n}^\alpha))^C$ circuits, and has detection probability $\gamma_\alpha \cdot (\gamma_\alpha)^2$. As for the additional parameters, we have circuit size s_0, number of queries q_0 and error parameter δ_α.

IIa. We now apply error-reduction to $\mathcal{A}_I$ to increase the detection probability from $(\gamma_\alpha)^3$ to γ_α, using Theorem 4.8. For $\mathsf{q}'_1 = O(\frac{1}{\delta_\alpha}\log\frac{1}{\varepsilon_\alpha})$ and $\ell = poly\left(\frac{\mathsf{q}_0}{\gamma_\alpha}\right)$, we have that the resulting assignment tester $\mathcal{A}_{IIa}$ produces $poly(1/\varepsilon_\alpha) \cdot \left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)} = \left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)}$ circuits. Its circuit size is $\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}'_1$, the number of queries is q'_1, the distance parameter is $2\delta_\alpha$ and the detection probability is γ_α as desired.

IIb. Next, we get $\mathcal{A}_{IIb}$ by reducing the distance parameter of $\mathcal{A}_{IIa}$ from $2\delta_\alpha$ to δ according to Lemma 4.1. As discussed in Remark 4.4, this will require applying $\mathcal{A}_{IIa}$ on circuits of size $M = O(\frac{2\delta_\alpha}{\delta}\mathsf{n}) = O(\mathsf{n})$ (recall that $\delta = \Omega(1/\bar{q})$ is a constant). We note that as long as α is small enough this will only require invoking $\mathcal{A}$ recursively on circuits of size smaller than $\mathsf{s}(M) < \mathsf{n}$. The number of circuits that $\mathcal{A}_{IIb}$ produces is $poly(1/\varepsilon_\alpha)\left(M^{c_1+\alpha C}\right)^{O(\ell)} = \left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)}$. The number of queries is $\mathsf{q}_1 = \mathsf{q}'_1 + O(\frac{1}{\delta}\log\frac{1}{\varepsilon_\alpha}) = O(\frac{1}{\delta}\log\frac{1}{\varepsilon_\alpha})$. Note that this is consistent with our previous definition of q_1. The circuit size is $\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}'_1 + O(\frac{1}{\delta}\log\frac{1}{\varepsilon_\alpha}) <$

$\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}_1$ (by setting $\mathsf{q}_1 = O(\frac{1}{\delta}\log\frac{1}{\varepsilon_\alpha})$ to be large enough). The detection probability remains γ_α and the distance parameter is δ as desired.

IIc. Finally, for the composition $\mathcal{A}_{IIc} = \mathcal{A}_{IIb} \circ \mathcal{A}_0$ to be well-defined we need to have $\mathsf{n}_0 \geq O(\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}_1)$. Note that $\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}_1 = poly(s_0)$ (there are hidden constants here which depend on $\delta_\alpha, \varepsilon_\alpha$ and $\bar{q}$). Since $s_0 < (n_0)^{\alpha_0}$ the condition that $\mathsf{n}_0 \geq O(\ell \cdot \mathsf{s}_0 \cdot \mathsf{q}_1)$ is satisfied provided that α_0 is small enough (which translates to n_0 and R_0 being large enough). In addition, we have that δ_0 satisfies the condition of Theorem 3.7, and allows the composition of $\mathcal{A}_{IIb}$ with $\mathcal{A}_0$. The parameters obtained by $\mathcal{A}_{IIc}$ are: number of circuits is $\mathsf{R}_0 \cdot \left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)} = \left(\mathsf{n}^{c_1+\alpha C}\right)^{O(\ell)}$; the circuit size is s_0; the number of queries is q_0; the detection probability is $(\gamma_\alpha)^2$ and the distance parameter is δ.

It remains to verify that the parameters of $\mathcal{A}_{IIc}$ are as good as those required of $\mathcal{A}$ (and so $\mathcal{A}$ will just simulate $\mathcal{A}_{IIc}$ on inputs with $|\varphi| > \mathsf{n}_0$, and this will complete the proof by induction). This already holds directly for all parameters apart of the number of circuits which is supposed to be n^C. Therefore, the condition that is imposed is $\left(\mathsf{n}^{c_1+\alpha C}\right)^{c' \cdot \ell} \leq \mathsf{n}^C$, for some constant c' derived from the analysis above. For that we set $\alpha < 1/(2c' \cdot \ell)$ (which causes the constant c_1 that controls the number of circuits output by $\mathcal{A}_\alpha$ to increase, but still remain constant). The required condition now holds as long as $C > 2c_1 \cdot c' \cdot \ell$. ■

6 A Combinatorial Construction

In this section we describe a fully combinatorial construction of assignment testers with parameters $\mathsf{R} = n^{poly\log n}$ and $\mathsf{s} = O(1)$. This implies a combinatorial proof for $NP \subseteq PCP[polylog, 1]$. Although not as strong as the PCP-Theorem, such a result still has quite powerful implications, for example that approximating Max-3-SAT is quasi-NP-hard.

In the construction of the assignment tester $\mathcal{A}$, we follow the recursion style of Section 5. Namely, we compose an assignment tester $\mathcal{A}_\alpha$ that produces circuits of size $\mathsf{n}^{O(\alpha)}$, with a recursive call to $\mathcal{A}$ itself on circuits of this size. We then reduce the error of the resulting tester to maintain the induction hypothesis (as described in step II of Figure 5). The main difference, is that here we can no longer assume the existence of $\mathcal{A}_\alpha$ as a black box that is given to the construction. We will therefore have to construct $\mathcal{A}_\alpha$ ourselves.

The first observation we make is the following: when defining $\mathcal{A}$ on input circuits of size n we only need to apply $\mathcal{A}_\alpha$ on inputs of size $O(\mathsf{n})$. However, when defining $\mathcal{A}$ on input circuits of size n we are allowed to assume that $\mathcal{A}$ is already well defined for inputs of size smaller than n (in fact we are already making this assumption in the construction of Section 5). For this reason, we do not really have to build $\mathcal{A}_\alpha$ from scratch. Instead, in the definition of $\mathcal{A}_\alpha$ on inputs of size $\mathsf{n}' = O(\mathsf{n})$ we are allowed to invoke $\mathcal{A}$ itself as long as we only invoke it on inputs that are smaller than n. In the presentation below we will concentrate on this task of constructing $\mathcal{A}_\alpha$ given $\mathcal{A}$ which is only defined on smaller inputs (formally stated in Lemma 6.3). This will include all of the new ideas not already described in Section 5.

Putting everything together (as we do in Section 6.3), we obtain a construction of $\mathcal{A}$ which on inputs of size n makes two recursive calls, one with inputs of size $\mathsf{n}^{1-\alpha}$ (in the construction of $\mathcal{A}_\alpha$) and the other with inputs of size $\mathsf{n}^{3\alpha}$ (when composing $\mathcal{A}_\alpha$ with $\mathcal{A}$ similarly to Section 5). Taking into account the additional costs of the error-reduction steps we get a recursion formula, essentially of the form, $\mathsf{R}(\mathsf{n}) = (\mathsf{R}(\mathsf{n}^{1-\alpha}) \cdot \mathsf{R}(\mathsf{n}^{3\alpha}))^{O(1)}$ which solves to $2^{poly\log n}$.

We note that, for the base of the induction, we rely on a given constant-size tester, say like the one in Proposition 5.4 of Section 5. For that we can rely even on extremely inefficient constructions of PCPs like the one based on the long-code. (As mentioned earlier, the constant-size assignment tester can also be found via exhaustive search but its existence is only guaranteed through some explicit proof.)

6.1 Overview – Constructing $\mathcal{A}_\alpha$ and Oblivious Testers

Before turning to the formal proof, let us first give a more detailed overview of the construction of $\mathcal{A}_\alpha$. This will also allow us to introduce a new desired property of assignment testers that lies at the heart of our new construction, namely we discuss the notion of *oblivious* assignment testers.

Recall the task at hand: we would like to construct a tester $\mathcal{A}_\alpha$ that on input circuits of size n produces circuits of size $n^{O(\alpha)}$. In addition we need $\mathcal{A}_\alpha$ to have constant query complexity. Namely, each of the circuits produced by $\mathcal{A}_\alpha$ should only read a constant number of variables. On the other hand, we have at our disposal a tester $\mathcal{A}$ that produces circuits of constant size (and in this respect is much better than what we need of $\mathcal{A}_\alpha$), albeit it is only defined on input circuits of size smaller than n by some constant factor.

Our approach is to decompose the input for $\mathcal{A}_\alpha$ into smaller pieces, and apply $\mathcal{A}$ on each piece separately. We then show how to combine the outcomes of the different runs of $\mathcal{A}$, resulting in our $\mathcal{A}_\alpha$.

Ignoring various important issues (which we will soon address) the construction goes as follows: On an input circuit φ of size n, the first step will be to *decompose* φ into the disjunction of $n^{3\alpha}$ smaller circuits $\bigwedge_{\mathbf{i}} \varphi_{\mathbf{i}}$. Now we apply the assignment tester $\mathcal{A}$ on each one of the $\varphi_{\mathbf{i}}$'s to obtain $\psi_{\mathbf{i},1}, \ldots, \psi_{\mathbf{i},\mathsf{R}_1}$, where each $\psi_{\mathbf{i},k}$ has constant size. It may be useful to think of $\psi_{\mathbf{i},1}, \ldots, \psi_{\mathbf{i},\mathsf{R}_1}$ as the $\mathbf{i}$'th row of a very skewed matrix (with longer rows and shorter columns). At this point, we will turn each column of this matrix into a new test. For the kth column we define the test $\psi^k = \bigwedge_{\mathbf{i}} \psi_{\mathbf{i},k}$. These circuits have size $O(n^{3\alpha})$, as they are composed of $n^{3\alpha}$ circuits of size $O(1)$. The transformation $\varphi \Rightarrow \{\psi^k\}$ defines the desired $\mathcal{A}_\alpha$ (in other words, on input φ, the tester $\mathcal{A}_\alpha$ outputs $\{\psi^k\}$).

In the informal description above, we have ignored two major issues, to which we now draw attention.

1. **Lack of robustness.** It comes up when trying to argue soundness for $\mathcal{A}_\alpha$, say as a PCP verifier: in the case that φ is not satisfiable then any assignment will fail to satisfy at least one of the $\varphi_{\mathbf{i}}$'s. We now want to argue that this implies that for this $\mathbf{i}$ many of the $\psi_{\mathbf{i},k}$'s will not be satisfied, which will finally imply that many of the ψ^k's will not be satisfied. Our problem is that while the given assignment will not satisfy one of the $\varphi_{\mathbf{i}}$'s it may be *close* to satisfying each and every one of them. In that case, applying $\mathcal{A}$ does not guarantee much regarding the $\psi_{\mathbf{i},k}$'s. Similarly to the proof for the composition theorem for assignment testers we solve this difficulty by applying $\mathcal{A}$ on a "robust version" of the $\varphi_{\mathbf{i}}$'s rather than on the $\varphi_{\mathbf{i}}$'s themselves. The robust circuits $\varphi'_{\mathbf{i}}$ will be defined such that when φ is not satisfiable then any assignment will be far from satisfying at least one of the $\varphi'_{\mathbf{i}}$'s (in fact we will need a bit more of the $\varphi'_{\mathbf{i}}$'s in order to argue soundness as an assignment tester rather than as a PCP verifier).

   We omit from this overview more specific details of how the $\varphi_{\mathbf{i}}$ and $\varphi'_{\mathbf{i}}$ circuits are defined and instead turn our attention to the second and potentially more devastating difficulty of our construction.

2. **Huge query complexity.** It is indeed true that the tester $\mathcal{A}_\alpha$ described above produces circuits ψ^k, of size $n^{O(\alpha)}$ as desired. However, in the general case, the query complexity of $\mathcal{A}_\alpha$ is also $n^{O(\alpha)}$ rather than $O(1)$ as required (recall that large query complexity means low "effective robustness" which implies that the composition of $\mathcal{A}_\alpha$ with the recursive call to $\mathcal{A}$ as in Section 5 will fail miserably). To overcome this major obstacle we introduce the notion of an "oblivious" assignment-tester. Such a tester produces output circuits that have the following property: the names of the variables queried by each circuit is a function only of $|\varphi|$ and not of φ. Formally,

   Definition 6.1 *Let $\mathcal{A}$ be an assignment tester that on input φ over variables $x_1, \ldots, x_{n_1}$, produces circuits $\psi^1, \ldots, \psi^{\mathsf{R}(\mathsf{n})}$ over variables $x_1, \ldots, x_{n_1}, x_{n_1+1}, \ldots, x_{n_1+n_2}$. $\mathcal{A}$ is called* oblivious *if for every n there exist $\mathsf{q}(\mathsf{n})$ functions*

$$\nu_1, .., \nu_{\mathsf{q}} : \{1, \ldots, \mathsf{R}(\mathsf{n})\} \to \{1, \ldots, n_1 + n_2\}$$

   such that ψ^k ($1 \leq k \leq \mathsf{R}$) always depends on the q variables indexed by $\nu_1(k), \ldots, \nu_{\mathsf{q}}(k)$. In other words, which variables are read by ψ^k depends on $|\varphi|$ but not on φ.

   Note that the *predicate* evaluated by each ψ^k may certainly depend on φ and not only on $|\varphi|$. Also note that we do not make any requirement on the efficiency of evaluating the ν_i's, (we can point out however that their efficiency does follows from the efficiency of $\mathcal{A}$).

   Intuitively, the reason that obliviousness of $\mathcal{A}$ is useful in reducing the query complexity of $\mathcal{A}_\alpha$ is that it implies a very regular structure of the variables read by the $\psi_{\mathbf{i},k}$'s that compose the final circuit ψ^k. This allows us to cluster the variables read by the output circuits ψ^k into larger, $n^{O(\alpha)}$-bit long, variables, such that each ψ^k depends only on a constant number of those larger variables. We would like to point out that this is not a generic method to obtain aggregation of variables but rather a method tuned towards our specific construction. Finally it remains to address the typical case where $\mathcal{A}$ is *not oblivious*. Luckily, we show that it is not hard to turn *every assignment tester* into an oblivious one.

6.2 Constructing $\mathcal{A}_\alpha$

We now formalize the construction of $\mathcal{A}_\alpha$ for inputs of size $N_\alpha > N$ based on $\mathcal{A}$ which is only defined for input circuits whose size is smaller than N.

Lemma 6.2 *For every $\alpha < \frac{1}{3}$, given an assignment tester $\mathcal{A}$ defined for inputs of size n such that $\mathsf{n} < N$, with parameters $(\mathsf{R}(\mathsf{n}), \mathsf{s}(\mathsf{n}) = O(1), \mathsf{q}(\mathsf{n}) = O(1), \delta(\mathsf{n}) = O(1), \varepsilon(\mathsf{n}) = O(1))$, we can construct $\mathcal{A}_\alpha$ defined for inputs of size at most N_α, where N_α is such that $N_\alpha^{(1-\alpha)} poly \log N_\alpha < N$, with circuit size $\mathsf{s}_\alpha(\mathsf{n}) = O(\mathsf{n}^{3\alpha})$, and other parameters:*

$$\mathsf{R}_\alpha(\mathsf{n}) = \mathsf{R}(\mathsf{n}^{1-\alpha} poly \log \mathsf{n}) \cdot \mathsf{n} \cdot poly(1/\varepsilon), \quad \mathsf{q}_\alpha(\mathsf{n}) = O(\frac{1}{\delta} \log \frac{1}{\varepsilon}), \quad \delta_\alpha(\mathsf{n}) = 24\delta, \quad \varepsilon_\alpha(\mathsf{n}) = \varepsilon\,.$$

Remark 6.3 *Note that the main advantage of $\mathcal{A}_\alpha$ over $\mathcal{A}$ is that it is defined for significantly larger inputs. In the application of Lemma 6.3, we will only need it to be defined for inputs of length at most $N_\alpha = c \cdot N$ for some constant $c > 1$.*

Proof: Let φ, $|\varphi| = \mathsf{n} \leq N_\alpha$, be the input circuit over a set of variables X. First, we decompose it as the disjunction of smaller circuits $\varphi_{\mathbf{i}}$.

Decomposing φ. We start by preprocessing φ, transforming it into a $3SAT$ formula $C_1 \wedge \cdots \wedge C_m$, where each variable appears only a constant number of times. This can easily be done by adding new variables Y for each internal gate as well as for multiple copies of X variables with out degree larger than one. Some of the clauses C_i will correspond to gates in φ. These clauses verify that the assignment for the gate variable is consistent with the assignment for the input-variables. In addition, for variables $y_{i_1}, y_{i_2}, y_{i_3}, \ldots$, that are supposed to be copies of a variable x_i, we have clauses verifying that $x_i = y_{i_1}$, and that $y_{i_j} = y_{i_{j+1}}$. Clearly, $|X \cup Y| \leq \mathsf{n} = |\varphi|$. This also guarantees that an assignment to X could be extended to an assignment to $X \cup Y$ that satisfies $C_1 \wedge \cdots \wedge C_m$ if and only if the assignment to X satisfies φ.

Assume that $\mathsf{n}_1 = \mathsf{n}^{1-\alpha}$ is an integer. Arrange the variables in an n^α-by-n_1 matrix V, such that the X variables fill the first $\frac{|X|}{\mathsf{n}_1}$ rows (for simplicity, assume that n_1 divides $|X|$). Each row of V consists of n_1 variables. Denote the rows by V_i, $i = 1, \ldots, n^\alpha$.

For every $\mathbf{i} = (i_1, i_2, i_3) \in [\mathsf{n}^\alpha]^3$, define $\varphi_{\mathbf{i}}$ to be the AND of all of the clauses C_i whose three variables are contained in $V_{i_1} \cup V_{i_2} \cup V_{i_3}$. This is a decomposition of φ in that

$$C_1 \wedge \cdots \wedge C_m = \bigwedge_{\mathbf{i}=(i_1,i_2,i_3)} \varphi_{\mathbf{i}} \,.$$

Moreover, the circuits $\varphi_{\mathbf{i}}$ are significantly smaller than n: Each $\varphi_{\mathbf{i}}$ depends on $\leq |V_{i_1} \cup V_{i_2} \cup V_{i_3}| = 3\mathsf{n}_1$ variables and consists of $O(\mathsf{n}_1)$ clauses (since each variable appears in a bounded number of clauses in the 3SAT formula).

Making the $\varphi_{\mathbf{i}}$'s robust. Next, we prepare the $\varphi_{\mathbf{i}}$'s for composition by making them robust (similarly to the proof of Lemma 3.6). Define new circuits $\varphi'_{\mathbf{i}}$ over new 'encoding' variables W_i in a natural way as follows. Let $e : \{0,1\}^{\mathsf{n}_1} \to \{0,1\}^{c\mathsf{n}_1}$ be an error correcting code, defined below. For every $i = 1, \ldots, n^\alpha$, let W_i be a set of $c\mathsf{n}_1$ new Boolean variables, supposedly representing the encoding via e of V_i. Let

$$\forall \mathbf{i} = (i_1, i_2, i_3) \in [\mathsf{n}^\alpha]^3, \qquad \varphi'_{\mathbf{i}} \text{ be a circuit over variables } W_{i_1} \cup W_{i_2} \cup W_{i_3} \tag{1}$$

that accepts only those assignments that are correct encodings (via e) of assignments that would have made $\varphi_{\mathbf{i}}$ accept.

With foresight, we choose an encoding e that in addition to having good rate and distance, also has the following "local-checkability" property. Given an assignment a for the original X variables, and an assignment b for the encoding variables, it is easy to verify (with two queries) that either b is close to $e(a)$, or it is far from any codeword. This property is needed for $\mathcal{A}_\alpha$ to be an assignment tester, rather than just a PCP verifier. We obtain this property by choosing e to be the string-concatenation of two error-correcting codes e_1 and e_2 where $e_1 : \{0,1\}^{\mathsf{n}_1} \to \{0,1\}^{\frac{c}{2}\mathsf{n}_1}$ is an error-correcting-code as defined in Lemma 2.1, and $e_2 : \{0,1\}^{\mathsf{n}_1} \to \{0,1\}^{\frac{c}{2}\mathsf{n}_1}$ is the trivial 'repetition' encoding, i.e. it outputs $c/2$ repetitions of each bit in the input. The distance of e is at least the distance of e_1, i.e. at least n_1. By Lemma 2.1, the size of $\varphi'_{\mathbf{i}}$ is linear in that of $\varphi_{\mathbf{i}}$, i.e. it is $O(\mathsf{n}_1)$.

Let W be an $n^\alpha \times c\mathsf{n}_1$ matrix whose rows are W_i. The first $\frac{c}{2}\mathsf{n}_1$ columns (the 'left' half) of W correspond to encoding according to e_1, and the last $\frac{c}{2}\mathsf{n}_1$ columns (the 'right' half) of W correspond to $\frac{c}{2}$ copies of V (recall V is the matrix that contains the X). The following proposition is straightforward,

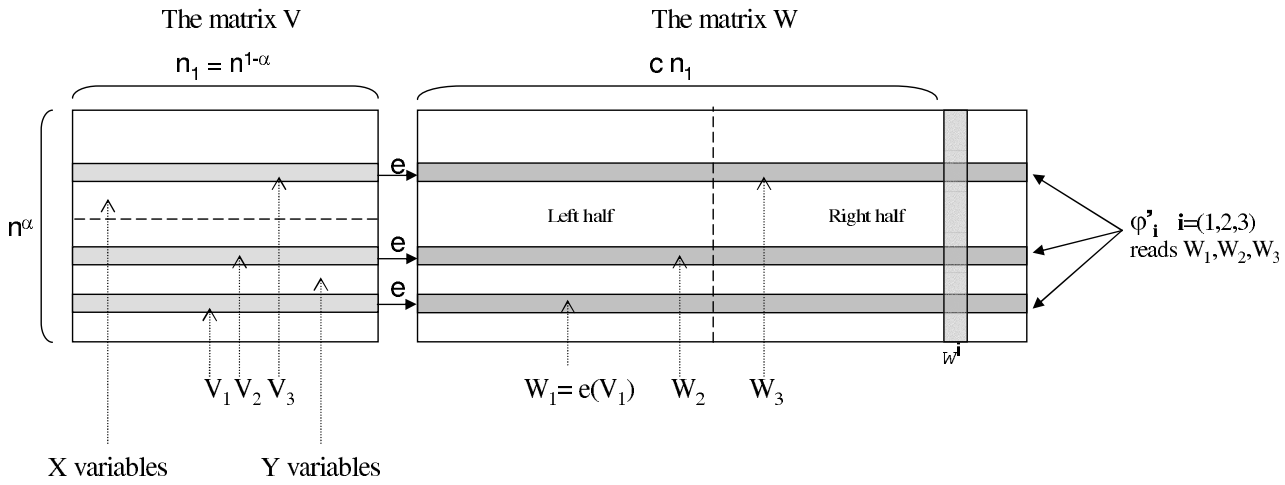

Figure 7: The variables $V = X \cup Y$ and W.

Proposition 6.4 *Let $b : W \to \{0,1\}$, and define $a_b : X \to \{0,1\}$ to be the maximum-likelihood decoding*[7] *of b. If a_b doesn't satisfy φ, then there exists some $\mathbf{i}$ such that $\varphi'_{\mathbf{i}}$ is $\frac{1}{6c}$-far from being satisfied by b.* ■

Proof: If a_b doesn't satisfy φ, then there is some $\varphi_{\mathbf{i}}$, $\mathbf{i} = (i_1, i_2, i_3)$, that falsifies a_b. This means that at least one of $W_{i_1}, W_{i_2}, W_{i_3}$ need to be changed *in more than half of the code's distance,* n_1 number of bits, so that together they encode a value that satisfies $\varphi'_{\mathbf{i}}$. The number of bits that need to be changed is at least $\mathsf{n}_1/2$, which is at least a $\frac{1}{6c}$ fraction of the $3 \cdot c\mathsf{n}_1$ input bits of $\varphi'_{\mathbf{i}}$. ■

Applying $\mathcal{A}$. As discussed in the informal overview, before applying $\mathcal{A}$ on the $\varphi'_{\mathbf{i}}$ circuits we need to turn $\mathcal{A}$ into a robust tester. Fortunately this is can be done by the following lemma whose proof we defer to Section **??**.

Lemma 6.5 *There exists some constant $c_1 > 0$, such that any assignment tester $\mathcal{A}$ can be made into an oblivious assignment tester $\mathcal{A}'$, and the parameters of $\mathcal{A}'$ are equal to $\mathcal{A}$'s parameters computed on input size $\mathsf{n}' = \mathsf{n} \cdot (\log \mathsf{n})^{c_1}$. Furthermore, on an n-bit input, $\mathcal{A}'$ only needs to invoke $\mathcal{A}$ on one n'-bit input.*

Let $\mathcal{A}'$, be the oblivious version of $\mathcal{A}$ guaranteed by Lemma 6.2. Next, apply $\mathcal{A}'$ on each of $\varphi'_{\mathbf{i}}$, denoting the output by $\psi_{\mathbf{i},1}, \ldots, \psi_{\mathbf{i},\mathsf{R}_1}$. Note that this application of $\mathcal{A}'$ only requires applying $\mathcal{A}$ on inputs of size $\mathsf{n}_1 \mathit{poly} \log \mathsf{n}_1 < N$ (by Lemma 6.2 and the definition of N_α and n_1), so $\mathsf{R}_1 = \mathsf{R}(|\varphi'_{\mathbf{i}}| \mathit{poly} \log |\varphi'_{\mathbf{i}}|) = \mathsf{R}(\mathsf{n}_1 \mathit{poly} \log \mathsf{n}_1)$. The circuits $\psi_{\mathbf{i},1}, \ldots, \psi_{\mathbf{i},\mathsf{R}_1}$, are over variables from at most three rows in W and over new variables $Z_{\mathbf{i}}$. Each circuit has size $O(1)$.

Circuit and Variable Aggregation. The next step is to think of each $\psi_{\mathbf{i},1}, \ldots, \psi_{\mathbf{i},\mathsf{R}_1}$ as a row in a matrix, and to define a new test for each column:

$$\forall k = 1, \ldots, \mathsf{R}_1, \qquad \psi^k = \bigwedge_{\mathbf{i}} \psi_{\mathbf{i},k} \,.$$

[7] As in Notation 2.2, the decoding of a word σ is $e^{-1}(\hat{\sigma})$ where $\hat{\sigma}$ is the codeword with minimal Hamming distance to σ (where ties can be broken arbitrarily).

There are $m = \mathsf{n}^{3\alpha}$ rows, and since the size of each $\psi_{\mathbf{i},k}$ is $O(1)$, these new circuits have size $m \cdot O(1) = O(\mathsf{n}^{3\alpha})$. Moreover, as long as the distance parameter of $\mathcal{A}$ is $\delta \leq \frac{1}{6c}$, it follows immediately from the above that

Proposition 6.6 *Let b, a_b be as in Proposition 6.5 above. Let b_1 be any assignment for $\bigcup_{\mathbf{i}} Z_{\mathbf{i}}$. If a_b doesn't satisfy φ, then at most ε of $\psi^1, \ldots, \psi^{\mathsf{R}}$ are satisfied by $b \cup b_1$.* ■

The circuits $\psi^1, \ldots, \psi^{\mathsf{R}}$, are therefore close to being the desired output of $\mathcal{A}_\alpha$. They do however, each depend on many ($\Theta(m)$) Boolean variables, rather than on a constant number of possibly larger-range variables. This is of course where the obliviousness of $\mathcal{A}'$ comes into play. We can simply cluster the variables into 'columns' so that each ψ^k will depend on a constant number of clusters (and the clusters are not too large).

For any $\mathbf{i} = (i_1, i_2, i_3)$, the circuits $\psi_{\mathbf{i},1}, \ldots, \psi_{\mathbf{i},\mathsf{R}_1}$, are defined over variables in three rows of the matrix W, namely $W_{i_1} \cup W_{i_2} \cup W_{i_3}$, and also over $Z_{\mathbf{i}}$, the auxiliary variables generated by running $\mathcal{A}'$ on $\varphi'_{\mathbf{i}}$. Let Z be the matrix whose rows are $Z_{\mathbf{i}}$. Now, for each k, the circuit $\psi'_{\mathbf{i},k}$ depends on some constant q locations in the string $W_{i_1} \cup W_{i_2} \cup W_{i_3}$, and $Z_{\mathbf{i}}$. Since $\mathcal{A}'$ is oblivious, the positions queried in these four strings are only a function of the index k (and are independent of $\mathbf{i}$). We therefore easily obtain the following proposition:

Proposition 6.7 *For each circuit ψ^k, all of the variables read by ψ^k are contained in q columns of W and Z, where $\mathsf{q} = O(1)$ is the query complexity of $\mathcal{A}$.*

The next step is now clear. We replace each column of variables (in both matrices W and Z) by a new (non-Boolean) variable that has width n^α or $\mathsf{n}^{3\alpha}$ respectively. Let $w^1, \ldots, w^{m_1}$ be new variables representing the columns of W, $m_1 = c\mathsf{n}_1$. Similarly and let $z^1, \ldots, z^{m_2}$ be new variables representing the columns of Z, $m_2 \leq \mathsf{R}_1$. The tests simulate the previous ones as follows. For every $k \in [\mathsf{R}_1]$, we define $\bar{\psi}^k$ to be the test that verifies the predicate ψ^k by reading the appropriate $\leq \mathsf{q}$ column variables that, by Proposition 6.7, contain all of the variables read by ψ^k.

The main point is that every variable in $W \cup Z$ appears in *exactly one* new column variable. Therefore, consistency is not an issue, as there is a one-to-one correspondence between assignments to the old variables and assignments to the new ones.

Consistency with the assignment to X. Finally, we must add tests that compare the values of $\{w^i\}_i \cup \{z^i\}_i$ to the assignment for the original X variables. Recall that each variable x belongs to some row V_i, and that $\frac{c}{2}$ entries in the right half of W_i are supposed copies of x. For every $x \in X$ let there be $\frac{c}{2}$ compare tests, each checking the value of x against one of the $\frac{c}{2}$ columns in W that are supposed to contain a copy of x.

We now amplify the detection probability of the compare tests. Let $d = O(\frac{1}{\delta} \log \frac{1}{\varepsilon})$, we define a set of tests by applying Corollary 2.5 on the sequence of the above compare tests, with parameters $\mu = 6\delta$ and $\beta = \varepsilon$. The number of compare tests is $\frac{c}{2}|X| = O(\mathsf{n})$, the size of each one is $O(\mathsf{n}^\alpha)$ (because it reads one bit of X, one variable w^i of width n^α, and later only compares two of its input bits). Denote the new tests by $\{\mathsf{compare}'_1, \ldots, \mathsf{compare}'_{M_1}\}$. By Corollary 2.5, $M_1 = poly(1/\varepsilon)\mathsf{n}$ and each $\mathsf{compare}'_i$ is the AND of d $\mathsf{compare}$ tests. Therefore it accesses $O(d)$ variables, and its size is $O(\mathsf{n}^{3\alpha})$.

Final circuits and their soundness. By appropriate replication, we can assume that the number (M_1) of tests in $\{\mathsf{compare}'_i\}$ is equal to the number of tests in $\{\psi'_i\}$, and both are equal

to $poly(1/\varepsilon)\mathsf{nR}_1$. Let the i-th final output circuit of $\mathcal{A}_\alpha$ be the AND of ψ'_i and $\mathsf{compare}'_i$. We next prove the soundness condition of $\mathcal{A}_\alpha$ (all other properties are easy to verify).

Lemma 6.8 (Soundness) *Let $\delta_1 = 24\delta$ and let $a : X \to \{0,1\}$. If a is δ_1-far from satisfying φ, then for every assignment b for $\{w^1,\ldots,w^{m_1}\} \cup \{z^1,\ldots,z^{m_2}\}$, at most ε of the final circuits can be satisfied.*

Proof: The assignment b can be read as an assignment for $W \cup Z$. Define $a_b : X \to \{0,1\}$ to be the maximum-likelihood decoding according to e of $b|_W$:

$$\forall V_i \subset X, \qquad a_b(V_i) = e^{-1}(b(W_i))\,.$$

If a_b does not satisfy φ, then the lemma follows from Proposition 6.6. Otherwise, a_b is δ_1-far from a. We must consider also the following 'majority' assignment, that can be defined according to b. Recall that the right half of W is supposedly the repetition encoding of V. Let $a_{b,maj} : X \to \{0,1\}$ be the majority decoding (understood to be defined on the appropriate coordinates) of b restricted to the right half of W (denoted W^r with rows W^r_i):

$$\forall V_i \subset X, \qquad a_{b,maj}(V_i) = majority(b(W^r_i))\,.$$

We have three assignments for X: $a, a_{b,maj}$ and a_b. If the first two are $\delta_1/2$-far from each other then by definition an ε-fraction of the compare tests will reject and we are done. We know that a and a_b are δ_1-far from each other, so the only other possibility is that the last two are $\delta_1/2$-far from each other. In this case, there must be some V_i for which $a_b(V_i)$ disagrees with $a_{b,maj}(V_i)$ on at least $\frac{\delta_1}{2}\cdot\mathsf{n}_1$ entries. This, by definition of $a_b, a_{b,maj}$ and e, implies that W_i is at least $\frac{\delta_1}{8}$ far from a legal codeword. Thus every circuit that reads W_i (say for example, $\varphi'_{\mathbf{i}}$ with $\mathbf{i} = (i, i_2, i_3)$) sees an input that is at least $\frac{\delta_1}{24}$-far from a satisfying input. Since $\delta = \frac{\delta_1}{24}$ is the distance parameter of $\mathcal{A}$, this means that at most ε of $\psi_{\mathbf{i},1},\ldots,\psi_{\mathbf{i},\mathsf{R}_1}$ are satisfied by b, which implies the same for $\psi^1,\ldots,\psi^{\mathsf{R}_1}$. ■

■

6.3 Proof of Theorem 1.2

As discussed above, given the construction of $\mathcal{A}_\alpha$, completing the proof of Theorem 1.2, goes along the same lines as the construction of Section 5. In the following, we formalize this idea.

Theorem 1.2 (Formal Statement) *There exist constants $\mathsf{s}_0, \mathsf{q}_0 > 0$ and $\varepsilon, \delta < 1$, and an explicit combinatorial construction of an assignment tester $\mathcal{A}$ with parameters $\mathsf{R}(\mathsf{n}) = n^{poly\log n}$, $\mathsf{s}(\mathsf{n}) = \mathsf{s}_0$, $\mathsf{q}(\mathsf{n}) = \mathsf{q}_0$, $\delta(\mathsf{n}) = \delta$, $\varepsilon(\mathsf{n}) = \varepsilon$.*

Proof: The proof follows by induction on the input size. As discussed above, for the base of the induction, we rely on a given constant-size tester, say like the one in Proposition 5.4 of Section 5. Denote this tester by $\mathcal{A}_0$ and let its parameters for input circuits of size at most n_0, be $(\mathsf{R}_0, \mathsf{s}_0, \mathsf{q}_0, \delta_0, \varepsilon_0)$.[8]

Now we assume by induction that $\mathcal{A}$ has already been constructed for inputs of length smaller than N. Its parameters are $(\mathsf{R}(\mathsf{n}), \mathsf{s}(\mathsf{n}) = \mathsf{s}, \mathsf{q}(\mathsf{n}) = \mathsf{q}, \delta(\mathsf{n}) = \delta, \varepsilon(\mathsf{n}) = \varepsilon)$, with $\mathsf{s}, \mathsf{q}, \varepsilon$ and δ being

[8] Such an $\mathcal{A}_0$ can be derived from an even less efficient tester $\mathcal{A}'_0$ that produces circuits of size smaller than $(n_0)^{1-\beta_0}$ for some constant β_0 rather than circuites that are smaller than $(n_0)^{\alpha_0}$ for arbitrarily small α_0. The desired $\mathcal{A}_0$ can now be obtained as in the proof of Corollary 5.3.

fixed constants (δ will be set by the proof to a small enough constant). Applying Lemma 6.3 for some fixed $\alpha < \frac{1}{3}$, we get an assignment tester $\mathcal{A}_\alpha$ that is defined for inputs of size $O(N)$ (and even larger), and such that $\mathcal{A}_\alpha$'s output circuit size is $\mathsf{s}(\mathsf{n}) = O(\mathsf{n}^{3\alpha})$. The rest of $\mathcal{A}_\alpha$'s parameters are $\mathsf{R}_\alpha(\mathsf{n}) = \mathsf{R}(\mathsf{n}^{1-\alpha} poly \log \mathsf{n}) \cdot \mathsf{n} \cdot poly(1/\varepsilon) = O(\mathsf{R}(\mathsf{n}^{1-\alpha} poly \log \mathsf{n}) \cdot \mathsf{n})$; $\mathsf{q}_\alpha(\mathsf{n}) = 3$ (this is obtained as in Proposition 4.11); $\varepsilon_\alpha(\mathsf{n}) = \varepsilon_\alpha$; $\delta_\alpha(\mathsf{n}) = \delta_\alpha$.

We can now use $\mathcal{A}_\alpha$ to extend $\mathcal{A}$ to n-bit inputs in exactly the same way it is done in Section 5. That is, we apply steps I, IIa, IIb, IIc described in Figure 5. The analysis is as in Section 5, but a bit less delicate as we are allowing ourselves quasi-polynomial R (and thus are less constrained in setting the relations between the various parameters). We therefore quickly go through the details.

I. $\mathcal{A}_I$ is the composition of $\mathcal{A}_\alpha$ with the recursive invocation of $\mathcal{A}$, using Theorem 3.7. This is well defined as long as we set δ (the distance parameter of $\mathcal{A}$) to be a small enough constant. $\mathcal{A}_I$ is defined for inputs of length $O(N)$. The number of circuits produced by $\mathcal{A}_I$ is $\mathsf{R}_I(\mathsf{n}) = O(\mathsf{R}(\mathsf{n}^{1-\alpha} poly \log \mathsf{n}) \cdot \mathsf{n} \cdot \mathsf{R}(O(\mathsf{n}^{3\alpha})))$. $\mathcal{A}_I$ has a (small) constant detection probability γ_I, circuit size s_0, number of queries q_0 and distance parameter δ_α.

IIa. We now apply error-reduction to $\mathcal{A}_I$ to increase the detection probability from γ_I to $\sqrt{\gamma}$, using Theorem 4.8, where $\gamma = 1 - \varepsilon$.

Since both γ_I and γ are constants we have that the resulting assignment tester $\mathcal{A}_{IIa}$ produces $\mathsf{R}_I(\mathsf{n})^{O(1)}$ circuits. Its circuit size is $O(\mathsf{s}_0)$, the number of queries is $O(1)$, the distance parameter is $2\delta_\alpha$ and the detection probability is $\sqrt{\gamma}$ as desired.

IIb. Next, we get $\mathcal{A}_{IIb}$ by reducing the distance parameter of $\mathcal{A}_{IIa}$ from $2\delta_\alpha$ to δ according to Lemma 4.1. As discussed in Remark 4.4, this will require applying $\mathcal{A}_{IIa}$ on circuits of size $M = O(\frac{2\delta_\alpha}{\delta}\mathsf{n}) = O(\mathsf{n})$. This is fine as $\mathcal{A}_I$ is defined for such input lengths.

The number of circuits that $\mathcal{A}_{IIb}$ produces is $\mathsf{R}_I(M)^{O(1)} = \mathsf{R}_I(O(\mathsf{n}))^{O(1)}$. The number of queries is $O(1)$, the circuit size is $O(\mathsf{s}_0)$ the detection probability remains $\sqrt{\gamma}$ and the distance parameter is δ as desired.

IIc. Finally, we define $\mathcal{A}_{IIc}$ as the composition $\mathcal{A}_{IIc} = \mathcal{A}_{IIb} \circ \mathcal{A}_0$. This is well defined as long as n_0 is a large enough constant with respect to s_0 (a more delicate analysis shows that here too it is enough to have $\mathsf{n}_0 = poly(\mathsf{s}_0)$) and δ_0 is a small enough constant that satisfies the condition of Theorem 3.7. The parameters obtained by $\mathcal{A}_{IIc}$ are: number of circuits is $\mathsf{R}_0 \cdot \mathsf{R}_I(O(\mathsf{n}))^{O(1)} = \mathsf{R}_I(O(\mathsf{n}))^{O(1)}$; the circuit size is s_0; the number of queries is q_0; the detection probability is $\sqrt{\gamma} \cdot (1 - \varepsilon_0)$ which is larger that γ in case ε_0 is small enough; the distance parameter is δ.

Finally $\mathcal{A}$ is simply defined to run $\mathcal{A}_{IIc}$ on circuits of length $\mathsf{n} \leq N$. It remains to verify that the parameters obtained by $\mathcal{A}_{IIc}$ for such circuits are as good as those required of $\mathcal{A}$. Note that this holds trivially for all parameters apart of the number of circuits. The number of circuits are $\mathsf{R}_I(O(\mathsf{n}))^{O(1)}$ which opens to the following recursion formula $\mathsf{R}(N) = (O(\mathsf{R}(\mathsf{n}^{1-\alpha} poly \log \mathsf{n}) \cdot \mathsf{n} \cdot \mathsf{R}(O(\mathsf{n}^{3\alpha}))))^{O(1)}$. Letting $a \geq 1$ be such that $(1-\alpha)^{a+1} + (3\alpha)^{a+1} < 1$, $\mathsf{R}(\mathsf{n}) = O(\mathsf{n}^{(\log \mathsf{n})^a})$ solves this recursive formula. ∎

6.4 Turning Testers Oblivious

We now return to proving Lemma 6.2 that shows how every tester can be turned oblivious.

Proof: (of Lemma 6.2) We start by showing the existence of the following "universal circuit": C_U takes as input some encoding of a Boolean circuit φ, an assignment a to the input variables X of φ and an assignment b to some additional "help variables". The following hold: (1) The encoding of φ can be computed in polynomial time. (2) If a satisfies φ then there exists an assignment b, (computed from φ and a in polynomial time), such that $C_U(\varphi, a, b) = 1$. (3) If a does not satisfy φ then for any b, $C_U(\varphi, a, b) = 0$. (4) For input circuits φ of size n, the size of C_U is bounded by $\mathsf{n} \cdot poly \log \mathsf{n}$ (and this in particular bounds the size of the encoding of φ).

Let us first see how such a C_U can produce an oblivious tester $\mathcal{A}'$. We will then describe how to construct C_U.

First apply $\mathcal{A}$ on the universal circuit C_U of the appropriate length. More accurately, $\mathcal{A}$ is applied to C'_U that takes the following inputs: an encoding of φ with the error correcting code of Lemma 2.1, and repetition encoding of the assignment to the X variables and a repetition encoding of the the assignment to the Y variables. In both cases, the number of repetition is set so that the encoding of φ, and the two assignments are of equal length. C'_U verifies that all of the encodings are legal. It then decodes the values it got and applies C_U. Now $\mathcal{A}'$ will run as follows. $\mathcal{A}'$ first applies $\mathcal{A}$ on C'_U. Then, $\mathcal{A}'$ evaluates the description φ that C_U expects, and hardwires the encoding of this description to the circuits produced by $\mathcal{A}$ (i.e., whenever these circuits query a value in the encoded description of φ this value is assigned by $\mathcal{A}'$ and the circuit is possibly simplified). Now the circuits that are obtained are just over the assignment to the X and Y variables. More accurately, the circuits are over the repetition encoding of these variables. However, whenever an assignment to a copy of one of the variables is required, we use the original assignment.

The correctness of $\mathcal{A}'$ follows from the correctness of $\mathcal{A}$ and the properties of C_U. It essentially has the parameters of $\mathcal{A}$ (when applied to inputs of quasi linear size). Finally, it is oblivious as the locations accessed by the circuits produced do not depend on φ at all (as these circuits were first produced by applying $\mathcal{A}$ to the universal circuit).

We now describe how to construct the universal circuit C_U. C_U will operate over a pre-processed φ, transformed into a $3SAT$ formula $C_1 \wedge \cdots \wedge C_m$, where each variable appears only a constant number of times. This can easily be done by adding new help variables Y for each internal gate as well as for multiple copies of X variables with out degree larger than one. Some of the clauses C_i will correspond to gates in φ. These clauses verify that the assignment for the gate variable is consistent with the assignment for the input-variables. In addition, for variables $y_{i_1}, y_{i_2}, y_{i_3}, \ldots$, that are supposed to be copies of a variable x_i, we have clauses verifying that $x_i = y_{i_1}$, and that $y_{i_j} = y_{i_{j+1}}$. Clearly, $|X \cup Y| \leq \mathsf{n}$. This also guarantees that an assignment to X could be extended to an assignment to $X \cup Y$ that satisfies $C_1 \wedge \cdots \wedge C_m$ if and only if the assignment to X satisfies φ.

A standard universal circuit for such a 3SAT formula is not hard to construct. The evaluation of the formula will be done in three phases. First we order the m clauses by the order of their variable of smallest index (i.e., first the clauses that contain x_1 (or its negation), then those that contain x_2 and do not contain x_1, and so on). Together with the clauses we order the variable names and their assignments. Now that both are jointly ordered, we can assign values to one variable from each clause (at this point, the location of each clause in the joint order is a constant distance away from the location of an assignment to one of its variables). Repeating these operations two additional times allows us to assign values to all of the variables of all of the clauses, which implies a value to the formula. It remains to argue that there are circuits of quasi linear size for sorting. This however follows immediately from the existence of sorting networks (where any one of a number of simple sorting networks will do, see, e.g., [CLR90,

Section 55]).

■

7 Discussion and Further Research

We have introduced the notion of assignment testers and provided a simple and truly modular composition theorem for testers. We feel that it is beneficial to state even the original proof of the PCP Theorem via assignment testers, as their composition seems to us simpler and more natural. In addition, we provided various generic transformations for assignment testers: (1) Making assignment testers "robust" (Section 3.4). (2) Reducing the distance parameter (Section **??**). (3) Error reduction via a new method of combinatorial aggregation (Section **??**).

Our first construction of assignment testers (given in Section 5) provides a new proof of the PCP theorem. It relies on the existence of a relatively weak assignment tester which is provided as a black-box. One advantage of this construction over the original proof of the PCP-Theorem is that the algebraic building block requires only the algebraic techniques that are already present in [BFLS91, FGL+91] (in particular, it only needs a weak form of the low-degree test, and it does not use aggregation via low-degree curves). More importantly, *the algebraic techniques are confined to the construction of the black-box*. The way the black-box was constructed can then be forgotten and we only care about its parameters. Finally, we only use a building-block PCP of one particular kind (as opposed to the original proof which also used Hadamard-based PCP). This is made possible through using a non-constant number of composition steps (to which our composition theorem readily yields itself). On the other hand, one may also consider the super-constant number of recursive steps to be a disadvantage. It is indeed harder to know "what's going on" by such a construction and particularly to track how the final variables relate to the original ones.

Our second construction is fully combinatorial, and it only relies on standard objects such as error-correcting codes and hitting sets (which both follow from expander graphs). It also relies on a constant-size tester, which is easy to construct due to its allowed inefficiency. In this respect the construction is even simpler (even though the combinatorial part of the construction is somewhat more complicated). The major disadvantage is of course the quasi-polynomial size. We note however that such a result has similar applications to those of the PCP Theorem (basing hardness of approximation on the still highly conservative assumption that NP is not contained in quasi polynomial time).

On the Robustness Property

We mentioned above that a very related notion to assignment testers was independently introduced by Ben-Sasson et. al. [BSGH+04], where it was named 'Robust PCPs of Proximity'. Interestingly, their motivation was completely different, namely constructing length-efficient PCPs and locally testable codes, yet they came up with essentially the same object.

Just as in [BSGH+04], robustness is an essential part of our composition. However, our composition theorem (Theorem 3.7) applies directly to assignment testers that are not necessarily robust. For that we use in the proof of the composition theorem a generic transformation that turns any assignment tester into a robust one, with the robustness inversely related to the query complexity. Most of our work can therefore ignore this additional parameter of robustness. Emphasizing assignment testers rather than robust assignment testers has the advantage of staying closer to the original definition of a PCP verifier. In addition, the importance of the query

complexity as the effective measure of robustness is emphasized. In particular, it seems more natural to state and prove the aggregation theorem, a central ingredient of our work, in terms of query complexity.

We note that the cost of a generic robustization transformation is too high in the context of [BSGH+04], hence they work only with assignment-testers that are, by definition, robust (indeed they call these objects Robust PCPs of Proximity). Since [BSGH+04] demonstrates the usefulness of having the robustness property as an explicit parameter, it could be interesting to study how our transformations on assignment testers behave in terms of this parameter.

On PCP-Testers and Property Testing

The notion of assignment testers is very related to the area of property testing [RS96, GGR98], and we were most likely inspired by property testing in coming up with this notion. An assignment tester can easily be converted into a test that checks if an assignment is close to being a satisfying assignment of a circuit φ. The object being tested is the assignment (hence the name 'assignment tester'), and the circuit φ is a description of the tested property (usually, in property testing this is a fixed property such as graph connectivity). Of course, the main difference from standard property testing is that assignment testers also rely on a proof (the assignment of the new variables) in order to perform the testing. This is a special case of the notion of proof-assisted-testing of Ergun, Kumar and Rubinfeld [EKR99].

Adding proofs to property testing allows extremely efficient testing of any property computable in polynomial time (as observed in [BSGH+04], this easily extends to properties in NP). Every such property can be tested by reading only $O(1)$ bits of an object X and a proof Y. This can loosely be interpreted as saying that every property has a highly-testable representation. While being a very powerful statement, it is also a flat one as it does not distinguish between different properties. In some sense, it reemphasizes that the richness of property testing is as a study of *specific representations*. (This was already well understood, as some properties behave very differently with respect to different representations.)

Locally Testable-Codes

An interesting aspect of our construction, especially of the fully combinatorial one, is that we do not make any real use of locally-testable codes (apart of the constant size tester). Nevertheless, as was shown in [BSGH+04], assignment testers easily imply locally-testable codes (and also a relaxed form of locally-decodable codes). The focus of [BSGH+04] was to get *short* locally-testable codes. Our Theorem 1.2 implies the first combinatorial construction of locally testable codes with subexponential (in fact even quasi-polynomial) rate.

Further Research

The most obvious problem that was left open by this work, is coming up with a combinatorial proof of the PCP theorem. This has been recently obtained by Dinur [Din05]. Still, one may also hope to give an elementary construction for the constant-size testers used by the combinatorial constructions (constant-sized testers are used also in [Din05]). Nevertheless, the known constructions based on Hadamard or Long codes are already rather simple (especially compared to other parts of the proof of the PCP-Theorem).

A task of a different nature is related to Raz's parallel repetition theorem [Raz98]. Recall that our aggregation method bypasses the complexity of this theorem by adding a few additional

"consistency queries". Nevertheless, we only use this method to reduce errors up to some constant probability. It seems possible that this approach could be extended to a simple, combinatorial way of reducing errors in an exponential rate. Though this will not provide a full substitute for the parallel repetition theorem, we still find it an interesting line for further research.

Acknowledgments

We sincerely thank the authors of [BSGH+04]: Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan and Salil Vadhan, for sharing with us preliminary stages of their own work, and for making excellent suggestions for ours. We thank Sanjeev Arora, Amir Shpilka, Luca Trevisan, and Avi Wigderson, for many useful discussions and comments. We would like to thank William Hesse and Ilan Newman for very helpful discussions regarding the circuit size of universal circuits. Special thanks to Oded Goldreich for his valuable contribution to the presentation of this paper.

References

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998. 1, 2, 3, 8, 12, 20, 26

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998. 1, 2, 3, 8, 12, 20, 26

[Bab85] L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 421–429, 1985. 2

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. 2

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–31, 1991. 1, 2, 4, 39

[BGKW88] 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 113–121, 1988. 2

[BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, June 1998. 5

[BSGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *To appear in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. ACM, 2004. 1, 3, 6, 7, 13, 40, 41

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill Book Company, 1990. 34

[CRVW02] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 659–668, Montréal, CA, May 2002. ACM. In joint session with *CCC '02*. 2

[Din05] I. Dinur. The PCP theorem by gap amplification. ECCC Technical Report TR05-046, 2005. 4, 7, 41

[EKR99] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate PCPs. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 41–50, 1999. 7, 40

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991. 1, 2, 4, 26, 39

[FK94] U. Feige and J. Kilian. Two prover protocols–low error at affordable rates. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 172–183, 1994. 20

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theor. Comput. Sci.*, 134(2):545–557, 1994. 2, 22

[Gal63] R.G̃. Gallager. *Low density parity check codes*. MIT Press, Cambridge, Massachusetts, 1963. 8

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998. 40

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proofs. *SIAM J. of Computation*, 18:186–208, 1989. 2

[Gol97] Oded Goldreich. A sample of samplers - a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(020), 1997. 9, 16

[GS] O. Goldreich and S .Safra. On the probabilistic checkable proofs of n. In preparation. 2

[GS97] Goldreich and 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. 7, 21, 45

[LFKN92] 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. 2, 26

[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. 9

[PS94] A. Polishchuk and D. Spielman. Nearly linear size holographic proofs. In *Proc. 26th ACM Symp. on Theory of Computing*, 1994. 4

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998. 7, 20, 41

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal of Computing*, 25(2):252–271, 1996. 40

[RVW00] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In IEEE, editor, *41st Annual Symposium on Foundations of Computer Science: proceedings: 12–14 November, 2000, Redondo Beach, California*, pages 3–13, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2000. IEEE Computer Society Press. 2, 5, 8, 9

[Sha92] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992. Prelim. version in 1990 FOCS, pages 11–15. 2

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1723–1731, 1996. Codes and complexity. 8

[SS96] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1710–1722, 1996. Codes and complexity. 8

[Sze99] Mario Szegedy. Many-valued logics and holographic proofs. In *ICALP*, pages 676–686, 1999. 7

[Tan81] R.M̃. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, Vol. IT-27,(5):533–547, 1981. 8

Appendix

A Analysis of the Consistency Test

As discussed in Section 4, the heart of our aggregation/parallelization theorem (Theorem 4.8) is a test T for the consistency of a table $F : \overline{X} \to \Sigma^\ell$ that is supposed to contain the values of some function $f : X \to \Sigma$ on all ℓ-tuples of inputs. In other words, $F(x_1, \ldots, x_\ell)$ is supposed to equal $(f(x_1), \ldots, f(x_\ell))$, for some underlying function f. Such a "combinatorial" consistency test was given by Goldreich and Safra [GS97]. We described our test in Figure 3, and stated its properties in Theorem 4.10. This test is in fact a (somewhat stronger) version of Lemma 1.1 from [GS97], which was derived as a simple special case of a more elaborate test which is their main objective. The more sophisticated test is for tables containing a small "derandomized" subset of ℓ-tuples. Interestingly, in the context of our paper, the simple and "inefficient" version of the GS-Test is sufficiently good. We now give a direct proof of Theorem 4.10. Our theorem proves that if the test accepts with high enough probability then the table F is in fact consistent with the *plurality* function of F (as in Definition 4.9), which is the function f that maximizes individual agreements between F and f. This seems more natural than the two-stage plurality function obtained by the proof of [GS97]. We view the main contribution of this section to be the new and direct proof of the consistency test, which relies on a rather natural Markov-Chain approach. Possibly, this proof can be generalized to work for a wider range of parameters (as mentioned in the open problems in Section 7).

Proof: (of Theorem 4.10) Clearly, the completeness condition holds. We prove soundness (that is, we analyze the rejection probability when the table is sufficiently inconsistent with the plurality function).

We will concentrate on the test with respect to the uniform distribution on X (i.e., $\mathcal{D}$ in the statement of the theorem is uniform). The proof for general $\mathcal{D}$ is obtained by simple reduction to the uniform case:

Proposition A.1 *Assuming a restricted version of Theorem 4.10 where $\mathcal{D}$ is the uniform distribution, the theorem holds for an arbitrary $\mathcal{D}$.*

Proof: We can translate any $\mathcal{D}$ over X into the uniform distribution over Z, where Z is obtained from X by duplicating elements in X according to their probability under $\mathcal{D}$. Elements that are not in the support of $\mathcal{D}$ are simply dropped. It is clear that the projection of the uniform distribution on Z to the original set of variables X can be made arbitrarily close to $\mathcal{D}$ (and from now on we will simply assume that the two distributions are identical).

Let $\overline{Z} \equiv Z^\ell$. The duplication of variables naturally defines $\hat{F} : \overline{Z} \to \Sigma^\ell$, where for each $\bar{\mathbf{z}} \in \overline{Z}$, we let $\bar{\mathbf{x}}$ be the corresponding ℓ-tuple in $\overline{X}$ and define $\hat{F}(\bar{\mathbf{z}}) = F(\bar{\mathbf{x}})$. The soundness of the test of Figure 3 when applied to F with respect to $\mathcal{D}$ immediately reduces to the soundness of the same test applied to $\hat{F}$ *with respect to the uniform distribution*. (Note that, by definition, the plurality function of $\hat{F}$ agrees on any two copies $\bar{\mathbf{z}}$ and $\bar{\mathbf{z}}'$ in $\overline{Z}$ that correspond to the same $\bar{\mathbf{x}} \in \overline{X}$.) ■

Another simplifying convention is our assumption that F is rotation consistent. Recall that F is rotation consistent if whenever $\bar{\mathbf{x}}'$ is obtained from $\bar{\mathbf{x}}$ using some cyclic shift of its ℓ components, F is consistent on these two entries (i.e., $F(\bar{\mathbf{x}}')$ can be obtained from $F(\bar{\mathbf{x}})$ by a similar shift). As discussed in Section 4, in our setting we can indeed assume that F is rotation

consistent. Moreover, it is easy to see that the general case can be reduced to the rotation consistent case.

Proposition A.2 *Assuming Theorem 4.10 holds for every F that is rotation consistent, the theorem holds for an arbitrary F, with the slightly revised test described in Figure 3.* ■

Proof: The original test makes two queries into the table F for the values $F(\bar{\mathbf{x}})$ and $F(\bar{\mathbf{x}}')$. The revised test will select $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$ in the same manner but instead of querying for $F(\bar{\mathbf{x}}')$ it will query for the value of F on a random cyclic rotation of $\bar{\mathbf{x}}'$. It is not hard to see that this is equivalent to performing the original test on a related *distribution* F' over rotation consistent tables. More specifically, for every set of ℓ entries $\bar{\mathbf{x}}^1, \ldots \bar{\mathbf{x}}^\ell$ that are cyclic shifts of each other, the value of F' on *all these tuples* is defined as $F(\bar{\mathbf{x}}^i)$ for a random i. This implies that if F violates the soundness of the revised test, then at least one of those rotation consistent tables violates the soundness of the original test. ■

Let us now recall the test T that we are analyzing and introduce some notations. Let $\bar{\mathbf{x}} = (x_1, \ldots, x_\ell)$ and suppose $F(\bar{\mathbf{x}}) = (a_1, \ldots, a_\ell)$, we write $\bar{\mathbf{x}}_i$ to mean x_i and $F(\bar{\mathbf{x}})_i$ to mean a_i.

1. Select $\bar{\mathbf{x}} \in \overline{X}$ uniformly at random.
2. Select a set of indices J_T such that each $j \in [\ell]$ is placed into J_T with probability $\alpha = 1/\sqrt[3]{\ell}$, independent of other choices.
3. Select $\bar{\mathbf{x}}' \in \overline{X}$ as follows: for each $j \in J_T$, we let $\bar{\mathbf{x}}'_j = \bar{\mathbf{x}}_j$, otherwise $\bar{\mathbf{x}}'_j$ is selected uniformly in X, independent of other choices.
4. Accept only if $F(\bar{\mathbf{x}})_j = F(\bar{\mathbf{x}}')_j$ for every $j \in J_T$, otherwise reject.

Let $f = f_F : X \to \Sigma$ be the plurality function of F. For every $\bar{\mathbf{x}} \in \overline{X}$ define the set of indices on which $F(\bar{\mathbf{x}})$ differs from f,

$$\mathsf{wrong}(\bar{\mathbf{x}}) \stackrel{def}{=} \{\, i \in [\ell] \mid F(\bar{\mathbf{x}})_i \neq f(\bar{\mathbf{x}}_i)\} \ .$$

Call a tuple $\bar{\mathbf{x}} \in \overline{X}$ bad if $|\mathsf{wrong}(\bar{\mathbf{x}})| > \frac{1}{\alpha} = \sqrt[3]{\ell}$. Let γ be the fraction of bad tuples. We will prove that the test T rejects with probability at least γ/c, where c is some absolute constant.

The general idea for our proof is the following: with probability γ a bad $\bar{\mathbf{x}}$ is selected by T. Fix some bad $\bar{\mathbf{x}}$, since $\mathsf{wrong}(\bar{\mathbf{x}}) > \frac{1}{\alpha}$ and each index is placed into J_T with probability α, we have that there exists some i in $\mathsf{wrong}(\bar{\mathbf{x}}) \cap J_T$ with constant probability. Recall that for every $i \in J_T$ the test T sets $\bar{\mathbf{x}}'_i = \bar{\mathbf{x}}_i$. Let $a_i = F(\bar{\mathbf{x}})_i$, by the definition of f, for a uniformly distributed $\bar{\mathbf{y}} \in \overline{X}$ such that $\bar{\mathbf{y}}_i = \bar{\mathbf{x}}_i$, we have that $\Pr[F(\bar{\mathbf{y}})_i = a_i \mid \bar{\mathbf{y}}_i = \bar{\mathbf{x}}_i] \leq 1/2$. Ideally, if $\bar{\mathbf{x}}'$ was distributed exactly like $\bar{\mathbf{y}}$, the test would reject with constant probability. Our argument is based on the fact that even though $\bar{\mathbf{x}}'$ has some dependency of $\bar{\mathbf{x}}$, it is still "sufficiently random" and therefore $F(\bar{\mathbf{x}}')_i \neq a_i$ (so the test rejects) with some constant probability. What do we mean by $\bar{\mathbf{x}}'$ being sufficiently random? Consider the stochastic process $G_{x,i}$ of selecting $\bar{\mathbf{x}}'$ given $\bar{\mathbf{x}}$ and conditioned on $i \in J_T$ and $\bar{\mathbf{x}}_i = x$ for some fixed i and x. We will show that this process has good expansion properties. We will later argue that these expansion properties are indeed sufficient to carry out the above intuition.

Definition A.3 *For any $i \in [\ell]$ and $x \in X$ define the set $V_{x,i} \stackrel{def}{=} \left\{ \bar{\mathbf{x}} \in \overline{X} \,\middle|\, \bar{\mathbf{x}}_i = x \right\}$. Note that $|V_{x,i}| = |X|^{\ell-1}$. Define $G_{x,i}$ to be the Markov process over $V_{x,i}$ where starting at $\bar{\mathbf{x}} \in V_{x,i}$*

we move to $\bar{\mathbf{x}}'$ selected as follows: $\bar{\mathbf{x}}'_i = x$ and for each $j \in [\ell] \setminus \{i\}$, with probability α we let $\bar{\mathbf{x}}'_j = \bar{\mathbf{x}}_j$, otherwise $\bar{\mathbf{x}}'_j$ is selected uniformly in X, independent of other choices. Let $A_{x,i}$ be the transition probability matrix corresponding to $G_{x,i}$.

Proposition A.4 *For every $i \in [\ell]$ and $x \in X$, the second eigenvalue of $A_{x,i}$ is $\lambda = \lambda(A_{x,i}) = \alpha$.*

Proof: Since in the space state of $G_{x,i}$, the value in the ith coordinate is fixed (to x), we can simply ignore this coordinate. On the other hand, $G_{x,i}$ acts on each one of the $\ell - 1$ other coordinates independently (in an identical way). Consider the transition probability matrix A^0 corresponding to the action of $G_{x,i}$ on each one of the $\ell - 1$ coordinates in $[\ell] \setminus \{i\}$. This matrix equals the convex sum $(1-\alpha)K_{|X|} + \alpha I_{|X|}$, where $K_{|X|}$ corresponds to moving to a uniformly distributed element (i.e., every entry in $K_{|X|}$ is $1/|X|$), and $I_{|X|}$ is the identity matrix. Consider any vector $P \in R^{|X|}$ which is perpendicular to the all one vector (i.e., the sum of the entries in P is zero), then $A^0 \cdot P = ((1-\alpha)K_{|X|} + \alpha I_{|X|})P = \alpha \cdot P$ (since $K_{|X|} \cdot P = \mathbf{0}$). This implies that the second eigenvalue of A^0 is α. To conclude, $G_{x,i}$ acts on $\ell - 1$ coordinates independently according to a transition probability matrix A^0 that has second eigenvalue α. It is well known and not hard to show that in such a case the second eigenvalue of $A_{x,i}$ is also α.[9] ■

Consider now the forgoing intuition. We know that a bad $\bar{\mathbf{x}}$ is selected by T with probability γ. Furthermore, we know that with constant probability, some $i \in \mathsf{wrong}(\bar{\mathbf{x}})$ is selected into J_T, meaning that $\bar{\mathbf{x}}'_i = \bar{\mathbf{x}}_i$. We hope to argue that with constant probability, this location i will reveal to T that the table F is inconsistent. Let $\bar{\mathbf{x}}_i = x$ and $a_i = F(\bar{\mathbf{x}})_i$, we have that $\bar{\mathbf{x}} \in V_{x,i}$ (as in Definition A.3), and since $i \in \mathsf{wrong}(\bar{\mathbf{x}})$, for most $\bar{\mathbf{y}} \in V_{x,i}$ we have that $F(\bar{\mathbf{y}})_i \neq a_i$. Therefore, if $\bar{\mathbf{x}}'$ was a random element of $V_{x,i}$ the test T would reject with constant probability (conditioned on $\bar{\mathbf{x}}$ being selected and $i \in J_T$). However, the distribution of $\bar{\mathbf{x}}'$ is obtained by taking a random step from $\bar{\mathbf{x}}$ according to the process $G_{x,i}$ (again, conditioned on $\bar{\mathbf{x}}$ being selected and $i \in J_T$). With this choice of $\bar{\mathbf{x}}'$, we can no longer argue that for any *fixed* choice of a bad $\bar{\mathbf{x}}$, with constant probability the test rejects (it may very well be that for all the "neighboring" $\bar{\mathbf{x}}'$'s the value $F(\bar{\mathbf{x}}')$ is consistent with $F(\bar{\mathbf{x}})$). We will therefore make an average argument that will exploit the expansion of $G_{x,i}$. We consider not one possible value of $\bar{\mathbf{x}}$ but rather a set $S_{x,i,a}$ that contains all the values of $\bar{\mathbf{x}}$ such that $\bar{\mathbf{x}}_i = x$ and $F(\bar{\mathbf{x}})_i = a$ (with $f(x) \neq a$). Conditioned on $\bar{\mathbf{x}} \in S_{x,i,a}$ and $i \in J_T$ we do have that with constant probability $\bar{\mathbf{x}}' \notin S_{x,i,a}$ (since $S_{x,i,a}$ contains at most half the tuples in $V_{x,i}$ and due to the expansion of $G_{x,i}$). Therefore, with this conditioning, the test will reject with constant probability (as with constant probability $F(\bar{\mathbf{x}}')_i \neq a$). Details follow.

Consider the process G that corresponds to T selecting $\bar{\mathbf{x}}'$ given $\bar{\mathbf{x}}$ (without further conditioning). The same transition $(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$ is also an edge in various $G_{x,i}$. To complete the proof, we want to lower bound the weight of rejecting edges in each $G_{x,i}$ separately (as outlined above) and deduce a similar lower bound for G (which reflects the rejecting probability of T). However, this may not be sound as the $G_{x,i}$'s may not be a "uniform enough cover" of G. More specifically, consider a transition $(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$, where $\bar{\mathbf{x}}$ is bad. For every $i \in \mathsf{wrong}(\bar{\mathbf{x}}) \cap J_T$, this corresponds to a transition in $G_{\bar{\mathbf{x}}_i,i}$ from some $S_{\bar{\mathbf{x}}_i,i,a}$ of density at most half in $V_{\bar{\mathbf{x}}_i,i}$. However, the cardinality of $\mathsf{wrong}(\bar{\mathbf{x}}) \cap J_T$ may vary quite a lot. Potentially, this could mean that by counting separately for each $G_{\bar{\mathbf{x}}_i,i}$, rejecting edges are counted many times while accepting edges are only counted a few times. Indeed, if we were assured that the size of $\mathsf{wrong}(\bar{\mathbf{x}})$ will either be zero (for all of the

[9] Each eigenvector of $A_{x,i}$ corresponds to an $(\ell-1)$-tuple of eigenvectors of A^0, the corresponding eigenvalue of $A_{x,i}$ is the product of the $\ell - 1$ corresponding eigenvalues of A^0. Therefore, the second eigenvalue of $A_{x,i}$ is α, and it is obtained as the product of $\ell - 2$ times the eigenvalue one and the eigenvalue α once.

good $\bar{\mathbf{x}}$'s) or some fixed value (for all of the bad $\bar{\mathbf{x}}$'s), then the cardinality of $\mathsf{wrong}(\bar{\mathbf{x}}) \cap J_T$ would not vary too much and the proof would become easier. Intuitively, the larger $|\mathsf{wrong}(\bar{\mathbf{x}})|$ is the better, since the test has "more opportunity" to detect an inconsistency and reject. However, the argument is much more subtle, due to the fact that we are not arguing for every value of $\bar{\mathbf{x}}$ separately but rather averaging over sets of values $S_{x,i,a}$.

To help us manipulate the conditional probabilities more elegantly, it is convenient to consider as a "mental-experiment" the following revised test T':

1. Choose $\bar{\mathbf{x}} \in \overline{X}$ uniformly at random.
2. Set $k(\bar{\mathbf{x}}) = \max\{16/\alpha, |\mathsf{wrong}(\bar{\mathbf{x}})|\}$.[10] Select an index $i \in [\ell] \cup \{0\}$ such that each $i \in \mathsf{wrong}(\bar{\mathbf{x}})$ is selected with probability $1/k(\bar{\mathbf{x}})$ and with the remaining probability $i = 0$. If $i = 0$ then T' accepts and halts.
3. Let $x = \bar{\mathbf{x}}_i$. Take a random step from $\bar{\mathbf{x}}$ to $\bar{\mathbf{x}}'$ according to $G_{x,i}$.
4. If $F(\bar{\mathbf{x}}')_i \neq F(\bar{\mathbf{x}})_i$ reject, otherwise accept.

We note that T' is not efficiently implementable, and is not meant to be. T' is merely a tool of the analysis, used to bound the probability that T rejects. The main convenience of T' is that it concentrates on a single possible inconsistency between $F(\bar{\mathbf{x}})$ and $F(\bar{\mathbf{x}}')$, namely inconsistency on the ith coordinate. This way we rather naturally avoid overcounting the rejection probability associated with a particular choice of $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$. Such overcounting may arise by counting the same pair $(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$ separately for every inconsistent coordinate. Note that if $\mathsf{wrong}(\bar{\mathbf{x}})$ is small then $J_T \cap \mathsf{wrong}(\bar{\mathbf{x}})$ is likely to be empty and thus T will accept. Therefore, if $\mathsf{wrong}(\bar{\mathbf{x}})$ is small, we let T' accept too with some probability (ignoring $F(\bar{\mathbf{x}}')$ altogether). We do that by allowing i to be set to zero with some probability that depends on the size of $\mathsf{wrong}(\bar{\mathbf{x}})$. We remark that if i is different than zero, it is uniformly distributed in $\mathsf{wrong}(\bar{\mathbf{x}})$.

To bound the rejection probability of T through the rejection probability of T' we would have liked to show that: (a) $\Pr[T' \text{ rejects}] = O(\Pr[T \text{ rejects}])$, and (b) $\Pr[T' \text{ rejects}] = \Omega(\gamma)$. However, as we do not know how to argue that (a) holds, we first factor out a possible (but rare) bad event B, on which T' may reject with significantly higher probability than T. Taking B into account, we prove Lemma A.8 and Lemma A.9 that are small variations on (a) and (b) above. These two lemmas immediately imply the soundness of T and therefore also Theorem 4.10.

For the definition of the bad event B, we will need a definition of a random variable $J_{T'}$ in analogy to the random variable J_T.

Definition A.5 *Recall the definition of the index i selected by T'. We define the random variable $J_{T'}$ as follows: $J_{T'}$ is set to be empty if $i = 0$. Otherwise $J_{T'}$ is the set of indices $j \in [\ell]$ for which T' sets $\bar{\mathbf{x}}'_j = \bar{\mathbf{x}}_j$ (in particular $i \in J_{T'}$).*

We would like to compare the behavior of T and T', and it would be convenient to do so conditioned on the value of $\bar{\mathbf{x}}$ and on a particular value J of both J_T and $J_{T'}$. However, for sets J such that $J \cap \mathsf{wrong}(\bar{\mathbf{x}})$ is large, we have that the probability that $J_{T'} = J$ may be significantly larger than the probability that $J_T = J$. The reason is that as long as $i \in J \cap \mathsf{wrong}(\bar{\mathbf{x}})$, it is possible that $J_{T'} = J$. Therefore, the larger $|J \cap \mathsf{wrong}(\bar{\mathbf{x}})|$ is, there are more ways to obtain $J_{T'} = J$. For this reason, we define the "bad event" B where $|J_{T'} \cap \mathsf{wrong}(\bar{\mathbf{x}})|$ is too large. The exact threshold is meant to facilitate the proof of Lemma A.8 and is less important for now.

[10] The purpose of the constant 16 is for the proof of Proposition A.11.

Definition A.6 *Define the event B to be $|J_{T'} \cap \mathsf{wrong}(\bar{\mathbf{x}})| / k(\bar{\mathbf{x}}) > 9\alpha$.*

Recall that by definition, $k(\bar{\mathbf{x}})$ is always positive and hence B is well defined. We would now like to argue that B is indeed a rare event.

Proposition A.7 *Conditioned on any particular value of $\bar{\mathbf{x}}$ and of i, the probability that B occurs is smaller than $2/9$.*

Proof: If $i = 0$ then $J_{T'}$ is empty and the proposition follows trivially. Otherwise, each index in $\mathsf{wrong}(\bar{\mathbf{x}}) \setminus \{i\}$ is placed into $J_{T'}$ with probability α (and i is placed into $J_{T'}$ with probability one). Therefore, the expected size of $J_{T'} \cap \mathsf{wrong}(\bar{\mathbf{x}})$ is at most $1 + \alpha \, |\mathsf{wrong}(\bar{\mathbf{x}})| \leq 1 + \alpha \cdot k(\bar{\mathbf{x}}) < 2\alpha \cdot k(\bar{\mathbf{x}})$ (last inequality follows from $k(\bar{\mathbf{x}}) \geq 16/\alpha > 1/\alpha$). Now, by Markov's Inequality, the event $B \equiv (|J_{T'} \cap \mathsf{wrong}(\bar{\mathbf{x}})| > 9\alpha \cdot k(\bar{\mathbf{x}}))$ has probability at most $2/9$. ■

We can now relate T' to T.

Lemma A.8 $\Pr[T' \textit{ rejects} \wedge (\neg B)] = O(\Pr[T \textit{ rejects}])$.

Proof: Both T and T' select $\bar{\mathbf{x}}$ uniformly at random. We prove the inequality separately for every value of $\bar{\mathbf{x}}$ (that is, conditioned on $\bar{\mathbf{x}}$ taking some arbitrary value). Therefore, fix the value of $\bar{\mathbf{x}}$ in an arbitrary way. We first argue that for any value $J \subseteq [\ell]$,

$$\Pr[T' \text{ rejects} \mid J_{T'} = J] \quad \leq \quad \Pr[T \text{ rejects} \mid J_T = J]. \tag{2}$$

First note that the distribution of $\bar{\mathbf{x}}'$ is identical in both cases (i.e., $\bar{\mathbf{x}}'_j = \bar{\mathbf{x}}_j$ for $j \in J$ and $\bar{\mathbf{x}}'_j$ is uniform outside J). If $J \cap \mathsf{wrong}(\bar{\mathbf{x}}) = \phi$, then $\Pr[T' \text{ rejects} \mid J_{T'} = J] = 0$ and we are done. Otherwise, T will reject if for some $j \in J$, we have that $F(\bar{\mathbf{x}}')_j \neq F(\bar{\mathbf{x}})_j$, whereas T' will reject only if $F(\bar{\mathbf{x}}')_i \neq F(\bar{\mathbf{x}})_i$ (recall that in this case $i \neq 0$ as otherwise $J_{T'}$ is empty). This implies inequality (2).

We now want to show that $\Pr[(J_{T'} = J)] = O(\Pr[J_T = J])$. However, this may not be true in two cases which fortunately enough we can ignore. The two cases to ignore are: (a) $J \cap \mathsf{wrong}(\bar{\mathbf{x}}) = \phi$, in which case T' always accepts, and (b) $|J \cap \mathsf{wrong}(\bar{\mathbf{x}})| / k(\bar{\mathbf{x}}) > 9\alpha$, in which case the event B occurs. Let the collection of sets J satisfying either (a) or (b) be denoted $\mathcal{J}_{\mathsf{ignore}}$. For all other values of J, the event B does not occur and therefore,

$$\Pr[T' \text{ rejects} \wedge (\neg B)] = \sum_{J \notin \mathcal{J}_{\mathsf{ignore}}} \Pr[T' \text{ rejects} \mid J_{T'} = J] \cdot \Pr[J_{T'} = J].$$

In addition, for $J \notin \mathcal{J}_{\mathsf{ignore}}$ we have that

$$\begin{aligned}
\Pr[J_{T'} = J] &= \sum_{j \in J \cap \mathsf{wrong}(\bar{\mathbf{x}})} \Pr[i = j] \cdot \alpha^{|J|-1} \cdot (1-\alpha)^{\ell - |J|} \\
&= \sum_{j \in J \cap \mathsf{wrong}(\bar{\mathbf{x}})} (1/k(\bar{\mathbf{x}})) \cdot \alpha^{|J|-1} \cdot (1-\alpha)^{\ell - |J|} \\
&= (|J \cap \mathsf{wrong}(\bar{\mathbf{x}})| / k(\bar{\mathbf{x}})) \cdot \alpha^{|J|-1} \cdot (1-\alpha)^{\ell - |J|} \\
&\leq 9\alpha \cdot \alpha^{|J|-1} \cdot (1-\alpha)^{\ell - |J|} \\
&= 9 \cdot \Pr[J_T = J] \,.
\end{aligned}$$

By inequality (2), we can now conclude that

$$\begin{aligned}\Pr[T' \text{ rejects} \wedge (\neg B)] &= \sum_{J \notin \mathcal{J}_{\mathsf{ignore}}} \Pr[T' \text{ rejects} \mid J_{T'} = J] \cdot \Pr[J_{T'} = J] \\ &\leq \sum_{J \notin \mathcal{J}_{\mathsf{ignore}}} \Pr[T \text{ rejects} \mid J_T = J] \cdot 9 \Pr[J_T = J] \\ &\leq 9 \Pr[T \text{ rejects}]\end{aligned}$$

■

It remains to bound the probability that T' rejects (again, factoring out the bad event B):

Lemma A.9 $\Pr[T' \textit{ rejects} \wedge (\neg B)] = \Omega(\gamma)$.

Proof: Throughout this proof, $\bar{\mathbf{x}}$ will always denote the tuple selected by T' at step 1, and i the index selected at step 2. If, say, we consider the probability of this tuple being equal to a specific tuple $\bar{\mathbf{z}}$, we write $\Pr[\bar{\mathbf{x}} = \bar{\mathbf{z}}]$, etc.

If T' rejects then in particular it selects $i \neq 0$. Recall that $\Pr[\bar{\mathbf{x}} \text{ is bad}] = \gamma$ by definition. By inspecting step 2 in the definition of T', and recalling that $\bar{\mathbf{x}}$ is bad means $|\mathsf{wrong}(\bar{\mathbf{x}})| > \frac{1}{\alpha}$, we have

$$\Pr[i \neq 0 \mid \bar{\mathbf{x}} \text{ is bad}] = \frac{|\mathsf{wrong}(\bar{\mathbf{x}})|}{k(\bar{\mathbf{x}})} = \frac{|\mathsf{wrong}(\bar{\mathbf{x}})|}{\max(|\mathsf{wrong}(\bar{\mathbf{x}})|, 16/\alpha)} \geq 1/16 .$$

So we can conclude that $\Pr[i \neq 0] = \Pr[\bar{\mathbf{x}} \text{ is bad}] \cdot \Pr[i \neq 0 \mid \bar{\mathbf{x}} \text{ is bad}] \geq \gamma/16$. To complete the proof we will show that

$$\Pr[T' \text{ rejects} \wedge (\neg B) \mid i \neq 0] = \Omega(1) . \tag{3}$$

We will do this by showing that for every fixed $i_0 \neq 0$ and $x_0 \in X$, conditioned on T' selecting $\bar{\mathbf{x}}$ and i such that $i = i_0$ and $\bar{\mathbf{x}}_i = x_0$, the test rejects (and B does not hold) with constant probability. Summing over all values of $i_0 \neq 0$ and $x_0 \in X$ this will complete the proof.

So let us fix an arbitrary $i = i_0 \neq 0$ and $x_0 \in X$. Denote $V_0 = V_{x_0,i_0}$, the set of tuples $\bar{\mathbf{z}}$ for which $\bar{\mathbf{z}}_{i_0} = x_0$ (as defined in Definition A.3). We will rely on the expansion of the Markov process $G_0 \stackrel{def}{=} G_{x_0,i_0}$ to show that the probability mass placed on tuples $\bar{\mathbf{z}} \in V_0$ for which $F(\bar{\mathbf{z}})_{i_0} \neq f(x_0)$, 'spreads' after one step of G_0. Thus, starting from such an $\bar{\mathbf{x}}$, with high probability we arrive at an $\bar{\mathbf{x}}'$ for which $F(\bar{\mathbf{x}})_{i_0} \neq F(\bar{\mathbf{x}}')_{i_0}$. To do this, we will partition the tuples $\bar{\mathbf{z}} \in V_0$ according to the value of $F(\bar{\mathbf{z}})_{i_0} = a$, and show that each part in the partition has many outgoing transitions, each causing T' to reject.

For every $a \in \Sigma$, let us define

$$S_a = \{\bar{\mathbf{z}} \in V_0 \mid F(\bar{\mathbf{z}})_{i_0} = a\} .$$

The set S_a also depends on the specific choice of x_0 and i_0, but this is omitted from the notation.

Let $E_{x_0,i_0,a}$ denote the event that T' selects $i = i_0$, and $\bar{\mathbf{x}}$ for which $\bar{\mathbf{x}}_i = x_0$ and $F(\bar{\mathbf{x}})_{i_0} = a$. Observe that conditioned on a specific value $i = i_0$ selected at step 2 of T', not all tuples in V_0 have positive probability of being chosen at step 1 of T'. Indeed, a tuple $\bar{\mathbf{x}} \in V_0$ for which $F(\bar{\mathbf{x}})_{i_0} = f(x_0)$ can not be chosen, because the conditioning requires in particular that $i_0 \in \mathsf{wrong}(\bar{\mathbf{x}})$ which is equivalent to $F(\bar{\mathbf{x}})_{i_0} \neq f(x_0)$. So $\Pr_{T'}[E_{x_0,i_0,a}] > 0$ implies $a \neq f(x_0)$. In that case let P_a denote the probability distribution over V_0, defined by

$$\forall \bar{\mathbf{z}} \in V_0, \quad P_a(\bar{\mathbf{z}}) = \Pr_{T'}[\bar{\mathbf{x}} = \bar{\mathbf{z}} \mid E_{x_0,i_0,a}] .$$

Fix some value $a \in \Sigma$ such that $\Pr[E_{x_0,i_0,a}] > 0$. Note that $\Pr[T' \text{ rejects} \wedge (\neg B) \mid E_{x_0,i_0,a}] \geq \Pr[T' \text{ rejects} \mid E_{x_0,i_0,a}] - \Pr[B \mid E_{x_0,i_0,a}]$. By Proposition A.7, the probability of the bad event B is bounded by $2/9$ even if conditioned on any specific $\bar{\mathbf{x}}$ and i, so in particular: $\Pr[B \mid E_{x_0,i_0,a}] < 2/9$. It is therefore sufficient to show that

$$\Pr[T' \text{ rejects} \mid E_{x_0,i_0,a}] \geq 1/4 \tag{4}$$

(as the probability in (3) will be lower bounded by $1/4 - 2/9 > 0$).

By the definition of T', we take a random step from $\bar{\mathbf{x}}$ according to the process G_0 and arrive at $\bar{\mathbf{x}}'$. The probability distribution over $\bar{\mathbf{x}}'$ is given by $A_0 P_a$, where A_0 is the transition matrix of G_0, and $P_a \in \mathbb{R}^{V_0}$ is a vector of probabilities that corresponds to the initial choice of $\bar{\mathbf{x}}$ conditioned on $E_{x_0,i_0,a}$. Finally, T' rejects if $F(\bar{\mathbf{x}}')_{i_0} \neq a$, or in other words, if $\bar{\mathbf{x}}' \notin S_a$. In conclusion,

$$\Pr\left[T' \text{ rejects} \,\middle|\, E_{x_0,i_0,a}\right] = \Pr[A_0 P_a \notin S_a]\,. \tag{5}$$

Following is a brief outline of how we lower bound (5). We already established that $a \neq f(x)$. This implies that S_a cannot contain more than half of the elements in V_0 (because f is the plurality; this is formally shown in Proposition A.10). Next, observe that if $\bar{\mathbf{x}}$ had been distributed uniformly in S_a, then by expansion, one step according to G_0 leaves this set with good probability. Our situation is slightly more complicated because $\bar{\mathbf{x}}$ is not distributed uniformly over S_a but rather according to P_a. Proposition A.11 proves that P_a is "uniform enough" (or rather, that it has "enough entropy"). We deduce our bound from a (known) variant of the expander mixing lemma (Proposition A.12) that can handle slightly skewed distributions.

Proposition A.10 $|S_a| / |V_0| \leq 1/2$.

Proof: Recall the definition of the plurality function (Definition 4.9). Since $f(x_0) \neq a$, we have that for less than half of $\{(\bar{\mathbf{z}}, j) \mid j \in [\ell],\ \bar{\mathbf{z}} \in V_{x_0,j}\}$, it holds that $F(\bar{\mathbf{z}})_j = a$. Since F is rotation consistent (namely, if $\bar{\mathbf{z}}'$ is obtained from $\bar{\mathbf{z}}$ using some cyclic shift of its ℓ components, then $F(\bar{\mathbf{z}}')$ can be obtained from $F(\bar{\mathbf{z}})$ in the same way), it is easy to verify that for any particular value of j, less than half of $V_{x_0,j}$ have $F(\bar{\mathbf{z}})_j = a$. Fixing $j = i_0$, for less than half of $V_0 = V_{x_0,i_0}$, it holds that $F(\bar{\mathbf{z}})_{i_0} = a$. The proposition follows. ∎

We first show that the distribution P_a is not too far from being uniform over S_a,

Proposition A.11 *Let $\lambda(A_0)$ be the second largest eigenvalue of A_0. Then*

$$\max_{\bar{\mathbf{z}}} \{P_a(\bar{\mathbf{z}})\} \leq 1/(16\,|S_a|\,\lambda(A_0)^2)\,.$$

Proof: Recall from Proposition A.4 that $\lambda(A_0) = \lambda(A_{x_0,i_0}) = \alpha$, and that $\alpha^3 = 1/\ell$. By definition of P_a, we may rewrite what we are trying to prove as

$$\forall \bar{\mathbf{z}}, \quad \Pr[\bar{\mathbf{x}} = \bar{\mathbf{z}} \mid E_{x_0,i_0,a}] \leq \frac{\alpha\ell}{16}\frac{1}{|S_a|}\,.$$

Clearly this probability is zero for all $\bar{\mathbf{z}} \notin S_a$. The point is to show that even though we have fixed i_0 and x_0 and a, this does not give too much information about $\bar{\mathbf{x}}$. We will use Bayes' rule,

$$\Pr[\bar{\mathbf{x}} = \bar{\mathbf{z}} \,|\, E_{x_0,i_0,a}] = \Pr[E_{x_0,i_0,a} \,|\, \bar{\mathbf{x}} = \bar{\mathbf{z}}] \cdot \frac{\Pr[\bar{\mathbf{x}} = \bar{\mathbf{z}}]}{\Pr[E_{x_0,i_0,a}]}\,. \tag{6}$$

We now estimate each of the three factors on the right hand side. Surely, with no conditioning, $\Pr[\bar{\mathbf{x}} = \bar{\mathbf{z}}] = 1/\left|\overline{X}\right|$. To estimate $\Pr[E_{x_0,i_0,a} \,|\, \bar{\mathbf{x}} = \bar{\mathbf{z}}]$ observe that conditioned on $\bar{\mathbf{x}} = \bar{\mathbf{z}}$, the random choice of the index i (at step 2 of T') determines whether the event $E_{x_0,i_0,a}$ occurs or not (because $\bar{\mathbf{z}}$ and $F(\bar{\mathbf{z}})$ and therefore $\bar{\mathbf{z}}_i$ and $F(\bar{\mathbf{z}})_i$ are already fixed). The probability, for a given $\bar{\mathbf{x}} = \bar{\mathbf{z}}$, of selecting $i = i_0$ at step 2 of T' is exactly $1/k(\bar{\mathbf{z}})$. If i_0 was selected, the event occurs iff $\bar{\mathbf{z}} \in S_a$ (which means that $\bar{\mathbf{z}}_{i_0} = x_0$ and $F(\bar{\mathbf{z}})_{i_0} = a$). Thus, $\Pr[E_{x_0,i_0,a} \,|\, \bar{\mathbf{x}} = \bar{\mathbf{z}}] = 1/k(\bar{\mathbf{z}}) \leq \alpha/16$ if $\bar{\mathbf{z}} \in S_a$ and equals zero otherwise.

Finally, we estimate $\Pr[E_{x_0,i_0,a}]$. We write

$$\Pr[E_{x_0,i_0,a}] = \sum_{\bar{\mathbf{z}} \in \overline{X}} \Pr[E_{x_0,i_0,a} \,|\, \bar{\mathbf{x}} = \bar{\mathbf{z}}] \cdot \Pr[\bar{\mathbf{z}}]$$

Just as before, $\Pr[E_{x_0,i_0,a} \,|\, \bar{\mathbf{x}} = \bar{\mathbf{z}}] = 1/k(\bar{\mathbf{z}})$ if $\bar{\mathbf{z}} \in S_a$ and 0 otherwise. Plugging this into the above equation gives $\Pr[E_{x_0,i_0,a}] = \sum_{\bar{\mathbf{z}} \in S_a} \frac{1}{k(\bar{\mathbf{z}})} \cdot \Pr[\bar{\mathbf{z}}] = \sum_{\bar{\mathbf{z}} \in S_a} \frac{1}{k(\bar{\mathbf{z}})} \cdot \frac{1}{|\overline{X}|} \geq \frac{|S_a|}{|\overline{X}|} \cdot \frac{1}{\ell}$, because always $k(\bar{\mathbf{z}}) \leq \ell$. Plugging everything into Equation (6),

$$\Pr[\bar{\mathbf{x}} = \bar{\mathbf{z}} \,|\, E_{x_0,i_0,a}] \leq \frac{\alpha}{16} \cdot \frac{\frac{1}{|\overline{X}|}}{\frac{|S_a|}{|\overline{X}|} \cdot \frac{1}{\ell}} = \frac{\alpha\ell}{16} \cdot \frac{1}{|S_a|}$$

■

It remains to observe that for every vector $\vec{p} = (p_1, \ldots, p_n)$, if $\sum_i p_i = 1$ then

$$\|\vec{p}\|_2^2 = \sum_i p_i^2 \leq \sum_i p_i \cdot \max_i p_i = \max_i p_i = \|\vec{p}\|_\infty \,.$$

So in particular, the previous proposition gives $\|P_a\|_2^2 \leq \max_{\bar{\mathbf{z}}} P_a(\bar{\mathbf{z}}) \leq 1/(16\,|S_a|\,\lambda(A_0)^2)$.

We can now conclude the proof of Lemma A.9, and of the theorem by the following proposition which is a slight generalization of the standard expander mixing lemma (for the case that the initial distribution is not uniform over some subset of the sample space).

Proposition A.12 $\Pr[A_0 P_a \in S_a] \leq |S_a| \,/\, |V_0| + \lambda(A_0)\sqrt{|S_a| \cdot \|P_a\|_2^2}$

Proof: Let $\chi_a \in \{0,1\}^{|V_0|}$ be the characteristic vector of S_a in V_0 (i.e., for every $\bar{\mathbf{z}} \in V_0$ we set $\chi_a(\bar{\mathbf{z}}) = 1$ iff $\bar{\mathbf{z}} \in S_a$). Let $U \in \{0,1\}^{|V_0|}$ be the vector corresponding to the uniform distribution, where all the entries in U equal $1/\,|V_0|$. As usual, we break P_a into a two components, parallel to U, and perpendicular to U. Let $P_a^\perp = P_a - U$. Note that $P_a^\perp$ is perpendicular to U because

$$\left\langle P_a^\perp, U \right\rangle = \langle P_a, U\rangle - \langle U, U\rangle = \sum_{\bar{\mathbf{x}} \in V_0} P_a(\bar{\mathbf{x}}) \cdot \frac{1}{|V_0|} - \sum_{\bar{\mathbf{x}} \in V_0} \frac{1}{|V_0|} \cdot \frac{1}{|V_0|} = \frac{1}{|V_0|}(1-1) = 0,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Now we have that

$$\begin{aligned} \Pr[A_0 P_a \in S_a] &= \langle \chi_a, A_0 P_a \rangle \\ &= \langle \chi_a, A_0 U \rangle + \left\langle \chi_a, A_0 P_a^\perp \right\rangle \end{aligned}$$

Since $A_0 U = U$, the first summand becomes $|S_a| \,/\, |V_0|$. To bound the second summand, we use Cauchy-Schwartz inequality, and get that $\left\langle \chi_a, A_0 P_a^\perp \right\rangle \leq \|\chi_a\|_2 \cdot \|A_0 P_a^\perp\|_2 \leq \sqrt{|S_a|} \cdot \|A_0 P_a^\perp\|_2$. The proposition follows since $\lambda(A_0)$ is the second largest eigenvalue of A_0,

$$\|A_0 P_a^\perp\|_2 \leq \|\lambda(A_0) P_a^\perp\|_2 \leq \lambda(A_0)\|P_a\|_2 \,.$$

■

Using $|S_a| / |V_0| \leq \frac{1}{2}$ and $\lambda(A_0)\sqrt{|S_a| \cdot \|P_a\|_2^2} \leq \lambda(A_0)\sqrt{(1/16) \cdot \lambda(A_0)^2} \leq \frac{1}{4}$ (as established in Propositions A.10 and A.11), we lower bound (5) by $1-(\frac{1}{2}+\frac{1}{4}) = \frac{1}{4}$ as needed. This concludes the proof of Equation (4) and therefore of Lemma A.9, showing that $\Pr[T' \text{ rejects} \wedge (\neg B)] = \Omega(\gamma)$.
■

Combining Lemma A.9 with Lemma A.8 we obtain $\Pr[T \text{ rejects}] = \Omega(\Pr[T' \text{ rejects} \wedge (\neg B)]) = \Omega(\gamma)$, and the theorem follows. ■

B Strengthening the black-box assignment tester from Section 5

Proof:(of Corollary 5.3) We would like to compose $\mathcal{A}_\beta$ with itself a constant number of times to reduce the size of the output circuits from $O(\mathsf{n}^{1-\beta})$ to $O(\mathsf{n}^\alpha)$. For the composition of $\mathcal{A}_\beta$ with itself to be well defined we need the distance parameter to be small enough with respect to the number of queries. To do that we first apply the distance-reduction transformation given by Lemma 4.1, to $\mathcal{A}_\beta$, reducing its distance parameter to another constant $\delta' < 1/(3c_2)$, where c_2 is the constant from the composition theorem, Theorem 3.7. We then reduce the number of queries to 3 as was done in Proposition 4.11. The result of these two steps is an assignment tester $\mathcal{A}'_\beta$ with parameters $(\mathsf{R}(\mathsf{n}) = \mathsf{n}^{O(1)},\ \mathsf{s}(\mathsf{n}) = O(\mathsf{n}^{1-\beta}),\ \mathsf{q}(\mathsf{n}) = 3,\ \delta(\mathsf{n}) = \delta',\ \varepsilon(\mathsf{n}) = \bar{\varepsilon})$, with $\bar{\varepsilon}$ being some fixed constant.

It is now possible to compose $\mathcal{A}'_\beta$ with itself and doing it a constant number of times (which depend on α of course) will reduce the output circuit size to $O(\mathsf{n}^\alpha)$. This almost gives us the assignment tester we are after, with the only problem being that the new error is some small constant that depends on α. Reducing the error parameter to $\varepsilon_\alpha = 0.1$ using Theorem 4.8, will complete our proof as we now have that the circuit size is indeed $O(\mathsf{n}^\alpha)$, and the only parameter that depends on α is the number of circuits produced (and it is polynomial for every fixed α).
■