Smooth and Strong PCPs

Orr Paradise
University of California, Berkeley, CA, USA
http://people.eecs.berkeley.edu/~orrp/
orrp@eecs.berkeley.edu

Abstract

Probabilistically checkable proofs (PCPs) can be verified based only on a constant amount of random queries, such that any correct claim has a proof that is always accepted, and incorrect claims are rejected with high probability (regardless of the given alleged proof). We consider two possible features of PCPs:

- A PCP is *strong* if it rejects an alleged proof of a correct claim with probability proportional to its distance from some correct proof of that claim.
- A PCP is *smooth* if each location in a proof is queried with equal probability.

We prove that all sets in $\mathcal{NP}$ have PCPs that are both smooth and strong, are of polynomial length, and can be verified based on a constant number of queries. This is achieved by following the proof of the PCP theorem of Arora, Lund, Motwani, Sudan and Szegedy (*JACM*, 1998), providing a stronger analysis of the Hadamard and Reed–Muller based PCPs and a refined PCP composition theorem. In fact, we show that any set in $\mathcal{NP}$ has a smooth strong *canonical* PCP of Proximity (PCPP), meaning that there is an efficiently computable bijection of $\mathcal{NP}$ witnesses to correct proofs. This improves on the recent construction of Dinur, Gur and Goldreich (*ITCS*, 2019) of PCPPs that are strong canonical but inherently non-smooth.

Our result implies the hardness of approximating the satisfiability of *"stable"* 3CNF formulae with *bounded variable occurrence*, where *stable* means that the number of clauses violated by an assignment is proportional to its distance from a satisfying assignment (in the relative Hamming metric). This proves a hypothesis used in the work of Friggstad, Khodamoradi and Salavatipour (*SODA*, 2019), suggesting a connection between the hardness of these instances and other stable optimization problems.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Interactive and probabilistic proof systems, Probabilistically checkable proofs, Hardness of approximation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.2

Related Version A full version of the paper is available on the Electronic Colloquium on Computational Complexity as TR-19-023, https://eccc.weizmann.ac.il/report/2019/023/.

Acknowledgements This work was done during my time at the Weizmann Institute of Science. It originated in a question of Irit Dinur, and I am grateful to her. My heartfelt appreciation goes to Oded Goldreich for guiding me through all stages of this work, from communication of Irit's question to me and up to this very write-up. Many thanks to Madhu Sudan and Oded for providing the vector-valued low-degree test (Appendix C). I also thank Elad Granot and Roei Tell for the helpful discussions, and Amey Bhangale, Tom Gur, and Eylon Yogev for suggesting improvements to the write-up. I wish to thank an anonymous reviewer for pointing out an issue with the smoothness of the construction of Section 5.3 as it appeared in a previous version.

1 Introduction

A *probabilistically checkable proof system (PCP)* offers verification based only on a tiny amount of random locations in an alleged proof. It is *complete* and *sound*: correct claims have a proof that is always accepted, and incorrect claims are rejected with high probability

regardless of the alleged proof. The study of these systems culminated in the PCP theorem ([3, 2]), stating that membership in any set in $\mathcal{NP}$ can be verified by reading a *constant* number of random locations from a PCP of *polynomial* length.

While soundness guarantees that incorrect claims are rejected with high probability, what about *incorrect proofs* for *correct claims*? By definition, a proof of a claim is incorrect if it is not always accepted by the probabilistic verifier. But how often is it rejected? In a *strong* PCP, an alleged proof is rejected with probability proportional to its distance from a correct proof.

Strong PCPs are intuitively appealing: simply put, it is desirable to seek verification procedures that are sensitive to the correctness of the given claim *as well as the given proof*. From the perspective of property testing, the verifier of a strong PCP can be viewed as a (proximity oblivious) tester for the property of being a correct proof. In addition, strong PCPs have been used to construct better locally testable codes (see Section 1.3.1).

One immediately wonders if a *strong PCP theorem* holds as well; that is, whether any set in $\mathcal{NP}$ admits a *strong* PCP of polynomial length and constant query complexity. Dinur et al. answer this in the positive in a recent work [10],[1] however, their construction is inherently *non-smooth*, in the sense that certain locations in the proof are much more likely to be read than others.

A PCP is smooth if each location in its proof is equally likely to be read by the verifier. We expect natural PCPs to have smooth verifiers since, intuitively-speaking, we expect the verifier to treat all parts of the proof equally. Concretely, smooth PCPs are *tolerant* of errors, as a few corrupt locations in a correct proof still give high acceptance probability.[2] Prior works considered smoothness in the context of locally decodable codes (see Section 1.3.2).

Before moving on to the main result, let us motivate why smooth and strong PCPs are particularly natural in tandem. Fix a correct claim and consider two innate measures of the "incorrectness" of a proof: its probability of being rejected by the verifier and its distance from a correct proof. Denoting the first by ρ and the second by δ, a strong PCP guarantees that $\rho = \Omega(\delta)$. On the other hand, the tolerance of a smooth, constant-query PCP implies that $\rho = O(\delta)$. Thus, for a constant-query PCP that is both smooth and strong these measures coincide (up to a constant).

This work presents a construction of simultaneously smooth and strong PCPs of polynomial length for any set in $\mathcal{NP}$, verifiable by reading a constant number of bits from the proof. Specifically, we reanalyze and enhance the construction used in [3, 2] (with proof composition as in [6]) to obtain smooth and strong PCPs. The enhancements include the introduction of multi-piece PCPs, a smooth and strong-preserving transformation of these to single-piece PCPs, and a new composition theorem for smooth and strong PCPs.

Our result implies the hardness of approximating the satisfiability of *stable* 3CNF formulae with *bounded variable occurrence*, where *stability* means that the number of clauses violated by an assignment is proportional to its distance from a satisfying assignment (in the relative Hamming metric). We believe that the hardness of approximating 3SAT even under stability guarantees is related to the hardness of other stable optimization problems. Friggstad et al. provide evidence to this in [12], as they show that this result implies the hardness of approximating *perturbation-stable Euclidean k-means* (see Section 1.3.3).

---

[1] For technical reasons, their result is stated only for the class $\mathcal{UP} \subseteq \mathcal{NP}$, but can be easily adapted to suit all of $\mathcal{NP}$.

[2] Indeed, tolerance is (oppositely) related to strong PCPs which guarantee detection of errors in a correct proof. See next paragraph.

1.1 Main notions

In this section we formally define *strong PCPs* and *smooth PCPs*. But first, a reminder of standard PCPs and their basic properties.

▶ **Definition 1.1** (PCP). *A* probabilistically checkable proof system (PCP) *for a set* $S \subseteq \{0,1\}^*$ *is a probabilistic polynomial-time oracle machine* V*, called a* verifier *and denoted* V*, that satisfies the following conditions:*

- Completeness: *For all* $x \in S$ *there exists a proof* $\pi \in \{0,1\}^*$ *such that the verifier* V *accepts explicit input* x *and proof oracle* π *with probability* 1*.*
- Soundness: *For all* $x \notin S$ *and proof oracle* $\pi \in \{0,1\}^*$*, the verifier* V *rejects explicit input* x *and proof oracle* π *with probability at least* $1/2$*.*[3]

The maximal number of random coin tosses made by verifier V *on inputs of length* n *is its* randomness complexity*, denoted* $r(n)$*. The maximal number of queries made by the verifier* V *on inputs of length* n *is its* query complexity*, denoted* $q(n)$*.*

A PCP is *nonadaptive* if it determines all queries solely by its random coins and explicit input. All PCPs in this work are nonadaptive, and furthermore, they query the same number of bits regardless of the sampled coin sequence.

Notice that Definition 1.1 doesn't mention the *length* of the proof itself. That is because the number of possible locations the verifier might read in the proof can be upper-bounded based on its randomness and query complexities: a nonadaptive PCP that tosses r random coins and then makes q queries can query at most $q \cdot 2^r$ different locations in the proof. Thus, from here on we will ignore the proof length and focus on the randomness and query complexities.

1.1.1 Strong PCPs

Strong PCPs are PCPs that reject incorrect proofs even for correct claims with probability proportional to the distance of such proofs from correct ones. Throughout this work, *distance* refers to the relative Hamming distance: For a fixed alphabet Σ (one can think of $\{0,1\}$, but we will use different alphabets later), the *relative hamming distance* between strings $x, y \in \Sigma^n$ equals the fraction of locations on which they differ, and is denoted $\delta(x,y)$. If $\delta(x,y) < d$ then x is said to be *d-close* to y, and if $\delta(x,y) \geq d$ then x is *d-far* from y. The distance of a string x from a set $T \subseteq \{0,1\}^*$ is defined to be $\delta(x,T) := \min_{x' \in T \cap \{0,1\}^{|x|}} \delta(x,x')$, with the minimum over the empty set defined to be 1.

▶ **Definition 1.2** (Strong PCP). *A* strong PCP *for membership in set* S *with* strongness parameter $\alpha \in (0,1]$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted* V*, that satisfies the following conditions:*

- Completeness: *For all* $x \in S$ *there exists a proof* $\pi \in \{0,1\}^*$ *such that the verifier* V *accepts explicit input* x *and proof oracle* π *with probability* 1*. Such a proof* π *is called a* correct proof for x*.*
- Strong soundness:
  - *If* $x \notin S$ *then* x *has no correct proof.*
  - *Let* $\mathscr{P}(x)$ *denote the set of correct proofs for* x*. Then, the verifier rejects explicit input* x *and proof oracle* π *with probability at least* $\alpha \cdot \delta(\pi, \mathscr{P}(x))$*.*

[3] The constant $1/2$ can be replaced with any other constant $\alpha \in (0,1)$.

Note that strong soundness implies standard soundness, i.e. rejection of instances $x \notin S$ with constant probability regardless of the given proof oracle, because for these instances the verifier rejects with probability $\alpha \cdot \delta(x, \emptyset) = \alpha$.

1.1.2 Smooth PCPs

Smoothness is a straightforward notion and is defined for any oracle Turing machine. To us, oracles always have finite domains, and we associate the oracle $f\colon [n] \to \{0,1\}$ with an n-bit string $f(1)\cdots f(n)$.

▶ **Definition 1.3** (Smooth oracle machine). *A probabilistic oracle Turing machine M is* smooth *if for any explicit input and oracle, the probability that M queries each location of its oracle (in any of its queries) is equal. That is, given access to oracle f and letting $Q(j)$ be the event that M queries location j of f in any of its queries, it holds that $\mathbb{P}[Q(j)] = \mathbb{P}[Q(j')]$ for every $j, j' \in [|f|]$, where $|f|$ denotes the length of the oracle f.*

Additional discussion on smoothness and a nearly-equivalent definition can be found in Appendix B.

1.2 Contributions

1.2.1 Smooth and Strong PCPs for $\mathcal{NP}$

The main contribution is a proof of the following result.

▶ **Theorem 1.4** (Main result). *Every set in $\mathcal{NP}$ has a smooth and strong PCP with logarithmic randomness and constant query complexities.*

At this point one might wonder: A strong PCP rejects incorrect proofs with probability proportional to their distance from correct proofs – but *who are these correct proofs*? In the case of Theorem 1.4, we can say that the correct proofs for any fixed instance are obtained by a polynomial-time computable bijection of $\mathcal{NP}$-witnesses to correct proofs.

▶ **Theorem 1.5** (Main result, strengthened). *Fix a set $S \in \mathcal{NP}$, and let $\mathscr{W}(x)$ denote the set of $\mathcal{NP}$-witnesses for an instance x of S. Then, S has a PCP as in Theorem 1.4 with a polynomial-time computable* canonical proof strategy $\Pi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *such that for every $x \in \{0,1\}^*$, $\Pi(x, \cdot)$ is a bijection between the set $\mathscr{W}(x)$ and the set of correct proofs $\mathscr{P}(x)$.*

1.2.2 Hardness of approximation

Recall that the PCP theorem implies that for some $\rho \in (0,1)$, it is $\mathcal{NP}$-hard to distinguish 3CNF formulas that are satisfiable from ones in which any assignment violates at least a ρ fraction of the clauses. Theorem 1.4 yields a similar result for 3CNF formulas that are *"stable"* and have each variable occurring in a bounded number of clauses. Here *stability* means that the number of clauses violated by an assignment is (at least) proportional to its distance from a correct assignment (in the relative Hamming metric). Formally,

▶ **Definition 1.6** ((α, b)-`stable3SAT`). *A 3CNF formula φ is* α-stable *if any assignment that that is δ-far from a satisfying assignment violates at least an $\alpha\delta$ fraction of clauses in φ. A formula has* b-bounded-occurrence *if any variable occurs in at most b clauses. For constants α and b, the promise problem (α, b)-`stable3SAT` is distinguishing b-bounded-occurrence 3CNF formulas that are α-stable and satisfiable from ones in which any assignment violates at least an α fraction of the clauses.*

One motivation for our interest in stable and bounded-occurrence formulas is that they exhibit an interesting structural property. For a fixed satisfiable formula, we consider two natural measures of the "cost" (i.e., "badness") of an assignment. The first and most common one is the *fraction of clauses* violated by the assignment. The second is the *fraction of variables* on which the assignment disagrees with the closest satisfying assignment (i.e. the relative Hamming distance of the assignment from the set of satisfying assignments). We denote the first by δ and the second by ρ. Now, stable formulas have $\delta = \Omega(\rho)$, while bounded-occurrence formulas satisfy $\delta = O(\rho)$, since changing the value of a variable affects a bounded number of clauses. Hence, these two measures coincide for stable and bounded-occurrence formulas (up to a constant factor); the fraction of clauses unsatisfied by an assignment approximately reflects its distance from a satisfying assignment. Theorem 1.5 implies a hardness of approximation result for such formulas.

▶ **Corollary 1.7.** *There exist $\alpha \in (0,1)$ and $b \in \mathbb{N}$ such that (α, b)-*`stable3SAT` *is $\mathcal{NP}$-hard. Furthermore, it is $\mathcal{NP}$-hard under parsimonious Karp reductions.*

The proof of Corollary 1.7 are deferred to Appendix A. We proceed with a discussion of two of its implications.

Consider a *distance oracle* that, given a 3CNF formula φ and assignment σ, returns the (relative Hamming) distance of σ from the set of satisfying assignments of φ if φ is satisfiable, and answers arbitrarily otherwise. Efficiently finding a satisfying assignment given such an oracle can be done by greedily minimizing the distance returned by the oracle. What if instead we are given access to an *approximate* distance oracle, which returns the distance of an assignment from the set of satisfying assignments *up to some multiplicative constant*? Corollary 1.7 and the observation that precedes it imply that an approximate distance oracle is not enough to find even an approximately satisfying assignment; that is, that for some constant $\alpha \in (0,1)$, finding an assignment that satisfies more than an α-fraction of clauses is $\mathcal{NP}$-hard even when given access to an approximate distance oracle. This is because, as observed, the answers of an approximate distance oracle can be efficiently emulated for stable and bounded-degree formulas, and Corollary 1.7 asserts $\mathcal{NP}$-hardness of finding an approximately satisfying assignment for such formulas.

In addition to the aforementioned intrinsic motivation for the study of stable and bounded occurrence instances, Corollary 1.7 implies a hypothesis used in the recent work of Friggstad et al. [12, Hypothesis 1], yielding the first hardness of approximation result for perturbation-stable Euclidean k-means. More on this in Section 1.3.3.

1.2.3 Smooth and Strong Canonical PCPs of Proximity for $\mathcal{NP}$-relations

PCPs of Proximity (abbreviated *PCPPs*, aka *assignment testers*), introduced in [11, 6], are PCPs placed on an even tighter budget, with access to their input accounted for in their query complexity. Since PCPPs cannot read the entirety of their input oracle, they aren't able to distinguish inputs in the set from inputs *close* to being in the set. As such, PCPPs should satisfy a relaxed notion of soundness that requires them to reject (with high probability) only input oracles *far* from correct ones.

More generally, PCPPs verify membership of a *pair* $(x; y)$ in a *relation* $R \subseteq \{0,1\}^* \times \{0,1\}^*$, when given explicit (i.e. unaccounted) access to x and oracle (i.e. accounted) access to y, as well as access to a proof oracle.

▶ **Definition 1.8** (PCP of Proximity (PCPP)). *A* PCP of Proximity system (PCPP) *for relation* $R \subseteq \{0,1\}^* \times \{0,1\}^*$ *with* proximity parameter $\delta > 0$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted* V *such that the following hold:*

- Completeness: *If* $(x, y) \in R$ *then there exists a proof* π *such that the verifier* V *accepts explicit input* x*, input oracle* y *and proof oracle* π *with probability* 1*.*
- Soundness: *If* y *is* δ*-far from* $\{y' : (x, y') \in R\}$*, then for any proof oracle* π*, the verifier rejects explicit input* x*, input oracle* y *and proof oracle* π *with probability at least* $1/2$*.*

Strong canonical PCPPs

PCPP soundness is somewhat reminiscent of strong soundness, but note that in the former rejection probability is related to the distance of the *input oracle* from being correct, rather than the distance of the *proof oracle* from being correct (here we think of the explicit input as fixed).[4] Indeed, the adaptation of strong soundness to the setting of proximity verification, i.e. *strong PCPPs*,[5] combines these two requirements: a *strong PCPP* is required to reject with probability proportional the *maximum* between the distance of the input oracle y to a correct input oracle y', and the proof oracle π to a correct proof oracle π' for y'.

Actually, we won't bother to formally define strong PCPPs, because we show the existence of even *stronger* (pun intended) constructs. Our PCPPs have a *canonical* transformation of correct inputs to correct proofs, meaning that for each correct input $(x; y)$, our PCPPs have a unique *canonical proof* $\Pi(x; y)$ that is always accepted by the verifier, whereas all other strings are rejected with nonzero probability.[6] But with what probability? Right, the maximum between the distance of the input oracle y to a correct input oracle y', and the proof oracle π to the canonical proof for x and y' (i.e. $\delta(\pi, \Pi(x; y'))$).

▶ **Definition 1.9** (Strong canonical PCPP). *A* strong canonical PCPP *for relation* R *with* strongness parameter $\alpha \in (0, 1]$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted* V*, coupled with a polynomial-time computable* canonical proof strategy *denoted* $\Pi\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$*, such that when the verifier is given explicit input* x *and access to an* input oracle y *and a* proof oracle π*, the following hold:*

- Canonical completeness: *The verifier accepts with probability* 1 *if and only if* $(x; y) \in R$ *and* π *is the corresponding canonical proof, i.e.* $\pi = \Pi(x; y)$*.*
- Strong canonical soundness: *Let* $R(x) \coloneqq \{y' : (x; y') \in R\}$*. Then, the verifier rejects with probability at least*

$$\alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y, y'), \delta(\pi, \Pi(x; y'))\right)\}$$

In particular, if $R(x)$ *is empty then the verifier rejects with probability* α*.*

Again, strong canonical soundness implies standard PCPP soundness, e.g. rejection of any input y that is 0.1-far from $R(x)$ with constant probability, because in this case the verifier rejects with probability at least $\alpha \cdot \min\{0.1, 1\}$, where α is the (constant) strongness parameter.

---

[4] Furthermore, Definition 1.9 is *proximity-oblivious*, in the sense that rejection probability grows with the distance of the oracles from being correct, whereas Definition 1.8 offers rejection with constant probability of inputs whose distance from being correct is farther than some constant.

[5] Not to be confused with the related [17][Definition 5.7], which we will soon refer to as *strong canonical PCPPs*. See also Footnote 7.

[6] This is the bijection mentioned in Theorem 1.5.

▶ Remark 1.10. Previously (e.g. [17, 15, 10]), strong canonical PCPPs were considered only for *unambiguous relations*; that is, only for relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which $|R(x)| \leq 1$ for any x. Our definition is more general as it does not place a restriction on the relation R.[7] Furthermore, Definition 1.9 allows the verifier to take an explicit input (as in [6]), whereas past works studied PCPPs that have access to input and proof oracles but no (auxiliary) explicit input.

Smooth PCPPs

PCPP verifiers are oracle machines that have *two* oracles, and we say that a PCPP is smooth if it is smooth on each of its oracle. Formally, we generalize Definition 1.3 to suit a *t-oracle* Turing machine for any constant $t \in \mathbb{N}$, which is a machine that has access to a t oracles (where t is a constant).

▶ **Definition 1.11** (Smooth multi-oracle machine)**.** *A probabilistic t-oracle Turing machine M is* smooth *if for any explicit input and oracles, the probability that M queries each location of each of its oracle (in any of its queries) is equal. That is, given access to oracles $f_1, \ldots, f_t$ and letting $Q_\ell(i,j)$ be the event that the ith query of M is to location j of f_ℓ, it holds that $\mathbb{P}\left[\bigcup_{i=1}^{q} Q_\ell(i,j)\right] = \mathbb{P}\left[\bigcup_{i=1}^{q} Q_\ell(i,j')\right]$ for each $\ell \in [t]$ and every $j, j' \in [|f_\ell|]$, where $|f_\ell|$ denotes the length of the ℓth oracle f_ℓ.*

We prove the following theorem.

▶ **Theorem 1.12.** *Every $\mathcal{NP}$-relation has a smooth and strong canonical PCPP with logarithmic randomness complexity and constant query complexities.*

Notice that a PCPP for relation R yields a PCP for the set $S_R \coloneqq \{x : \exists y \ (x;y) \in R\}$: a proof that $x \in S_R$ (in the PCP for S_R) is composed of some y such that $(x;y) \in R$, followed a proof (in the PCPP for R) that $(x;y) \in R$. Furthermore, the PCP for S_R retains the strongness of the PCPP for R and, under a reweighing of the input oracle (presented in Section 2), smoothness is retained as well. Therefore, Theorem 1.5 follows from Theorem 1.12.

1.3 Related work

1.3.1 Strong (canonical) soundness

The term *strong* in the definition of strong PCPs is inspired by *strong locally testable codes (strong LTCs)*, which are codes whose local test rejects with probability proportional to the distance of the input from the code. In fact, strong canonical PCPPs have seen numerous uses in works on strong LTCs, as follows.

Goldreich and Sudan [17] defined strong canonical PCPPs in their work on strong LTCs,[8]and constructed such PCPPs for certain linear codes. An extension of this initial construction saw use by Gur and Rothblum [19] as they obtained strong LTCs that allow for a relaxed notion of local decoding (of [6, Section 4.2]). Goldreich et al. [15] later improved these

[7] In fact, our work is the first to make a semantic distinction between *strong* and *strong canonical*, recognizing strong canonical PCPPs as special type of strong PCPPs in which there is an efficient *canonical* transformation between $\mathcal{NP}$-witnesses and proofs, in addition to strong soundness. In previous works, the terms *strong PCPP* and *strong canonical PCPP* were used synonymously, which is consistent with our distinction, given that these works considered PCPPs only for unambiguous relations. See Section 1.3.1 for more on previous works using strong canonical PCPPs.

[8] See Footnote 7 for a warning on the usage of the terms *strong* and *strong canonical* in previous works.

codes, again utilizing strong canonical PCPPs. Gur et al. [18] employed strong canonical PCPPs in their work on relaxed locally correctable codes. We stress that all these works featured PCPPs for linear subspaces (e.g. linear codes).

As mentioned, Dinur et al. [10] characterized *unambiguous* $\mathcal{NP}$ *(*$\mathcal{UP}$*)* in terms of strong canonical PCPPs, under the original and more restricted definition of strong canonicality generalized in this work (see Remark 1.10).

In [14, Section 13.2.2], strong PCPs are presented from the perspective of property testing as *locally testable proofs*, in analogy to locally testable codes.

1.3.2 Smoothness

Like strongness, smoothness too has its roots in coding theory, with the work Katz and Trevisan [21] examining *smooth locally decodable codes*. These are codes whose local decoding algorithm reads each bit in an alleged codeword with *approximately* equal probability (cf. Definition 1.3, which requires the probability to be *exactly* equal). Since its inception, this property has appeared in numerous works relating to locally decodable codes, e.g. [16, 24, 13]). Goldreich and Sudan [17, Definition 5.14] considered a similar feature for their Linear Inner Proof Systems (LIPS), which are fundamentally different from PCPs.

To avoid possible confusion, it is worth pointing out that the smoothness referred to in this work is as in the aforementioned works in coding theory, and *not* as in the *smooth label cover* of Khot [22].

The PCPPs of Theorem 1.12 are used in the recent work of Alman and Chen [1] which shows an explicit construction of rigid matrices using an $\mathcal{NP}$ oracle. They make use only of the smoothness of these PCPPs, but not the additional strong canonicality feature.[9]

1.3.3 Hardness of perturbation-stable Euclidean k-means

The hardness of approximating bounded-degree `stable3SAT` (Corollary 1.7) is the starting point of the first hardness of approximation result for perturbation-stable instances of Euclidean k-means, due to Friggstad et al. [12]. This connection between `stable3SAT` and perturbation-stable problems is an interesting direction for future research, so we provide a brief description of their result.

The study of optimization on perturbation-stable instances was initiated by Bilu and Linial [8] and Awasthi et al. [4] as a way of focusing on instances that can "occur in reality" (to quote the former). We consider the *Euclidean k-means* problem and its perturbation-stable instances, which are defined as follows:

- An instance of the k-means problem consists of $k \in \mathbb{N}$, *dimension* $d \in \mathbb{N}$, a *metric* $\mu : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$, *data points* $X \subseteq \mathbb{R}^d$, and *candidate centers* $C \subseteq \mathbb{R}^d$. The objective is to choose *centers* $S \subseteq C$ such that $|S| = k$ so as to minimize $\mathbb{P}_{x \in X} \left[\min_{c \in S} (\mu(x, c))\right]$. An instance is *Euclidean* if μ is the Euclidean metric.
- Fix $\gamma > 1$ and metric μ. A γ-perturbation of μ is a function $\mu' : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$ such that for any $x \neq y \in \mathbb{R}^d$,

  $$1 \leq \frac{\mu'(x, y)}{\mu(x, y)} \leq \gamma$$

  Notice that μ' is not necessarily a metric.

[9] As opposed to [12] (see Section 1.3.3) which uses both features simultaneously.

- For some fixed $\gamma > 1$, an instance (k, d, X, C, μ) of k-means is *γ-perturbation-stable* if it has a unique optimal solution $S^* \subseteq C$, and for any γ-perturbation μ' of μ, S^* is the optimal solution of the related instance (k, d, X, C, μ'). The *γ-perturbation-stable Euclidean k-means* problem is the k-means problem as previously described, under the promise that instances are *Euclidean* and are γ-perturbation-stable.

While solving (general, non-stable) Euclidean k-means is known to be $\mathcal{NP}$-hard to approximate [5], Awasthi et al. showed that introducing some perturbation-stability makes the problem easy; namely, they show that $(\sqrt{2}+3)$-stable Euclidean k-means can be solved exactly in polynomial time. Could the introduction of *any* amount of perturbation-stability render Euclidean k-means easy, or even just easy to approximate? The answer is *no*, as demonstrated by Friggstad et al. [12]. Their result is conditioned on an hypothesis asserting hardness of approximation of *bounded-occurrence and* `stable3SAT` (see Section 1.2.2), which is implied by Corollary 1.7.

▶ **Theorem 1.13** ([12]). *Assuming Corollary 1.7, there are $\gamma > 1$ and $\varepsilon > 1$ such that γ-perturbation-stable Euclidean k-means cannot be approximated within factor ε, unless $\mathcal{RP} = \mathcal{NP}$.*

1.4 Proof outline

To prove Theorem 1.12 we construct a PCPP with the necessary properties for the *circuit valuation* relation, denoted CircuitVal, which consists of all pairs $(C; y)$ such that circuit C accepts when given y as input. Then, any $\mathcal{NP}$-relation R has a PCPP (with the same properties) that, given explicit input x, efficiently computes a circuit C_x such that $(x; y) \in R$ if and only if C_x accepts y, and then runs the PCPP of CircuitVal on explicit input C_x and the same input and proof oracles. Following are highlights of our smooth and strong canonical PCPP for CircuitVal.

Multi-piece PCPPs (Section 2)

The PCPP for CircuitVal will be a variant of the PCPP presented in [3, 2, 6]. However, even a high-level inspection of this construction reveals that it is not at all smooth: for example, one building block (the Hadamard-based PCPP, mentioned below) consists of two "pieces" from which the verifier samples uniformly random locations, with the first piece substantially shorter than the second (so its bits are queried far more often). Indeed, this PCPP is not smooth when viewed as a single proof, but when partitioned into two *proof-pieces* (given as two proof-piece oracles), the verifier is smooth *as a three-oracle machine* (one input oracle and two proof-piece oracles). We present a transformation of multi-piece PCPPs to single-piece ones that *simultaneously preserves smoothness and strong canonicality*. This is done by replacing each proof-piece with a list of copies, so that the length of each list of copies is roughly the same.[10]

[10] To be clear, the Hadamard-based PCPP is but one example of non-smoothness (or rather, multi-piece smoothness) in the construction of [3, 2, 6]; the Reed–Muller-based PCPP, as well as PCPP composition, exhibit similar traits. Thus, our transformation of multi-piece PCPPs to single-piece PCPPs is used by all components of our construction, not just by the Hadamard-based PCPP.

Composing smooth strong canonical PCPPs (Section 3)

The run of a nonadaptive PCPP verifier can be viewed as a two-step process: first, it tosses some random coins and generates a *residual (decision) circuit* and *query locations* based on the coins it tossed, and then it queries its oracles and accepts or rejects according to the residual circuit's computation of their answers. A strong canonical and *robust* PCPP is such that, in expectation, the distance of its oracles' answers from satisfying the residual circuit reflects the distance of the oracles (in their entirety) from correct ones (i.e. a correct input oracle and a canonical proof oracle). Our *composition theorem* asserts that for a smooth strong canonical and robust PCPP, replacing the residual circuit's computation with an additional probabilistic verification (by an *inner* smooth strong canonical PCPP) yields a smooth strong canonical PCPP.

With a composition theorem at hand, we turn to the construction of smooth strong canonical and robust PCPPs, whose composition gives the PCPP postulated by Theorem 1.12.

The Hadamard-based smooth strong canonical PCPP (Section 4)

This is the Hadamard-based PCPP presented in [2], used as the innermost PCPP of the composition. Its proofs are based on the Hadamard encoding of the input oracle, and its verifier checks that the proof oracle encodes the input oracle (a "consistency check"), and uses the structure of the Hadamard code to check that the input oracle satisfies the (explicitly given) circuit. To show strong canonicality, we consider three cases:

Case 1: Both the input and proof oracles are close to correct ones. That is, the input oracle y is close to an input y' that is accepted by the (explicitly given) circuit, and the proof oracle π is close to its canonical proof oracle Π' of y'. The verifier checks consistency by performing a *strong codeword test* on the proof oracle, and checks that the decoding of the proof oracle agrees with the input oracle on a random bit. Strong testability means precisely that the first check rejects with probability $\Omega(\pi, \Pi')$, and the local decoding of a random location rejects with probability $\Omega(y, y')$.

Case 2: The input oracle is far from any correct input oracle. Then, standard PCPP soundness guarantees rejection with high probability.

Case 3: The proof oracle is far from the canonical proof of the input oracle. If the proof oracle is close to the canonical proof for some input $y' \neq y$ then the consistency check between the proof oracle and the input oracle rejects with probability $\Omega(\delta(y, y'))$. Otherwise the proof oracle is far from any canonical proof, and is therefore rejected with high probability by the strong codeword test.

Lastly, as mentioned above, we observe that it is smooth as a two-piece PCPP, and can therefore be transformed to a single-piece strong canonical PCPP.

The Reed–Muller-based smooth strong canonical robust PCPP (Section 5)

This PCPP is used in the outer layers of the composition, so we must show that it is smooth and *robust* strong canonical. Again, smoothness amounts to observing that it is multi-piece smooth. The analysis of its strong canonicality follows the same lines of the Hadamard-based PCPP, this time relying on a generalization of the strong Reed–Muller codeword test (provided by Oded Goldreich and Madhu Sudan in Appendix C). We first present a smooth strong canonical robust PCPP whose proofs are over a large alphabet, and then show that alphabet reduction (encoding each letter in an error correcting code) preserves smoothness and strong canonicality.

2 Multi-piece PCPPs

As said in Section 1.4, several of the PCPPs in the [2] construction are not smooth per se, but they spread their queries smoothly on each of a few significant *proof-pieces*. Such *multi-piece* PCPPs are required to satisfy a stricter form of strong canonicality, in which the rejection probability is related to the maximum between *each proof-piece oracle* to its corresponding proof-piece in a correct proof (and, as usual, the distance of the input oracle y to a satisfying input oracle y'). Motivated by this observation, we define *multi-piece strong canonical PCPPs*.

▶ **Definition 2.1.** *For a constant t, a* t-piece strong canonical PCPP system *for relation R with* strongness parameter $\alpha \in (0,1]$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted V, coupled with a sequence of polynomial-time computable* canonical proof-piece strategies*, denoted $\Pi_1, \ldots, \Pi_t \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$, such that when the verifier is given explicit input x and access to an* input oracle y *and a* proof-piece oracles $\pi_1, \ldots, \pi_t$*, the following hold:*

- Canonical completeness: *The verifier accepts with probability* 1 *if and only if $(x;y) \in R$ and π_i is the corresponding canonical proof-piece, i.e. $\pi_i = \Pi_i(x;y)$, for each $i \in [t]$.*
- Strong canonical soundness: *Let $R(x) \coloneqq \{y' : (x;y') \in R\}$. Then, the verifier rejects with probability at least*

$$\alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y,y'), \delta(\pi_1, \Pi_1(x;y')), \ldots, \delta(\pi_t, \Pi_t(x;y'))\right)\}$$

In particular, if $R(x)$ is empty, then the verifier rejects with probability α.

For each $i \in [t]$, the ith proof-piece length complexity*, denoted $\ell_i(n)$, is the length of the ith proof piece. The* ith proof-piece query complexity*, denoted $q_i(n)$, is the number of queries the verifier issues to the ith proof-piece given an input of length n.*[11]

2.1 From multi-piece to single-piece

We present a smoothness-preserving transformation of strong canonical multi-piece PCPPs to strong canonical PCPPs that use a single proof oracle. This transformation is not only convenient (for example, composition of multi-piece PCPPs gives one a multi-headache), but is also necessary for Theorem 1.12 which asserts the existence of *single-piece* smooth strong canonical PCPPs for every $\mathcal{NP}$ relation.

▶ **Lemma 2.2.** *Suppose that, for some constant t, a relation R has a* t-piece *smooth strong canonical PCPP with strongness parameter α, randomness complexity $r(n)$, and query complexity $q(n)$. Then, R has a* single-piece *smooth strong canonical PCPP with strongness parameter $\alpha/3$, randomness complexity $O(r(n) + \log q(n))$ and query complexity $O(q(n))$.*

Proof. Notice that if proof-piece lengths vary significantly, then simply concatenating the proof-pieces will not do, because bits of shorter pieces are sampled with higher probability than bits of longer pieces. Instead, each proof-piece is replaced with a list of copies such that each list is of equal length (up to a factor related to the proof-piece's query complexity). That is, the number of copies in the ith list is proportional to q_i/ℓ_i, which, by multi-piece smoothness, equals the probability that a bit in the ith proof-piece is queried by the multi-piece verifier.

[11] *Tedious comment:* We require the number of queries that the verifier issues to each proof-piece to depend only on the explicit input's length. This requirement is met by the all PCPPs used in this work.

The single-piece verifier emulates the multi-piece verifier on a random choice of proof-pieces (each proof-piece sampled uniformly from its list of alleged copies), and checks consistency of the copies in each list. Some care must be taken so that the consistency check does not harm smoothness (for example, checking consistency against a fixed copy would result in its bits being queried more often than others).

Strong canonicality of the single-piece verifier then follows from examining two possible cases: The first is that the given lists are noticeably inconsistent (i.e., there is a large discrepancy between the alleged copies), in which case the consistency check rejects with good probability. Otherwise, the lists are almost entirely consistent (i.e., each list of the given proof essentially consists of copies of some proof-piece), and then strong canonicality follows from strong canonicality of the multi-piece verifier.

Following is a detailed description and analysis of the construction. Let V be the postulated t-piece smooth strong canonical PCPP verifier with canonical proof-piece strategies $\Pi_1, \ldots, \Pi_t \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. We construct a single-piece PCPP $\overline{V}$ and start by describing its canonical proof strategy $\overline{\Pi}$. Fixing $(x, y) \in R$, we use the following notation:

- r denotes the number of random coins tossed by V when given x as explicit input.
- q_i denotes the number of queries V makes to the ith proof-piece oracle when given x as explicit input. The total number of queries V makes to all proof-piece oracles is $q \coloneqq \sum_{i=1}^{t} q_i$
- $\Pi_i \coloneqq \Pi_i(x; y)$ denotes the ith canonical proof-piece of $(x; y)$, and $\ell_i \coloneqq |\Pi_i|$ denotes its length.

Now, let $\overline{\Pi_i}$ be a list of $m_i \coloneqq (q_i + 1) \prod_{j \neq i} \ell_j$ copies of Π_i.[12] Indeed, m_i is proportional to the probability that a certain fixed bit of Π_i is sampled, namely q_i/ℓ_i.[13] The canonical proof of $(x; y)$ in the single-piece PCPP $\overline{V}$ is then the concatenation of all $\overline{\Pi_i}$'s.

We describe the run of the new verifier $\overline{V}$ given explicit input x and access to input oracle y and proof oracle $\overline{\pi} = (\overline{\pi_1}, \ldots, \overline{\pi_t})$, where $\overline{\pi_i} = (\pi_i^1, \ldots, \pi_i^{m(i)})$ is a list of $m(i)$ strings, each of length ℓ_i:

1. *Emulation:* For each $i \in [t]$, sample a uniformly random index $c_i \in [m_i]$. Next, emulate V on explicit input x, input oracle y and proof-piece oracles $\pi_1^{c_1}, \ldots, \pi_t^{c_t}$, rejecting if the emulation rejected. Let J_i denote the set of locations that V queries in $\pi_i^{c_i}$.
2. *Consistency check:* For each $i \in [t]$, sample uniformly from the remaining copies $c_i' \in [m_i] \setminus \{c_i\}$ and a uniformly random $j_i \in J_i$ and check that $\pi_i^{c_i'}[j_i] = \pi_i^{c_i}[j_i]$. That is, check that $\pi_i^{c_i}$ and $\pi_i^{c_i'}$ agree on a uniformly random location from the locations queried by the emulated verifier V in Step 1.[14]

   (Note that smoothness of V implies that j_i is uniformly distributed in $[\ell_i]$ (as detailed in Appendix B.2), and that only $\pi_i^{c_i'}[j_i]$ needs to be queried in this step.)

The single-piece verifier $\overline{V}$ uses t more queries than the emulated (multi-piece) verifier V since the consistency check requires querying an additional location from each list. The number of random coins is upper-bounded by $O(t^2 \cdot (r + \log q)) = O(r + \log q)$.

---

[12] The value $q_i + 1$ is used instead of q_i to account for the additional query made by the consistency check (Step 2 in the description of $\overline{V}$).

[13] Letting $m_i \coloneqq (q_i + 1)L/\ell_i$ for any L that is a common multiple of $\{\ell_i\}_{i \in [t]}$ would have worked as well.

[14] Indeed, the consistency check can be implemented in several other ways, for example without reusing the emulation copy c_i in the consistency check (i.e. checking consistency between c_i' and some c_i''), or by uniformly sampling j_i from all of $[\ell_i]$ rather than from J_i. However, other implementations require setting m_i to other (less informative) values.

Canonical completeness follows by observing that a proof is accepted with probability 1 if and only if it is formed of consistent lists (i.e. each list holds copies of some proof-piece), such that the proof-pieces in each list form a canonical proof in the multi-piece PCPP. We prove the remaining properties:

Smoothness

Fix a location in the proof oracle $\overline{\pi}$, which is the jth location of $\pi_i^{k_i}$ for some $i \in [t]$, $k_i \in [m(i)]$ and $j \in [\ell_i]$. This location is queried if and only if one of two disjoint events occur:

- In Step 1, $k_i = c_i$ *and* $j_i \in J_i$, i.e. the jth location of $\pi_i^{c_i}$ was queried by the emulated verifier. By multi-piece smoothness of the emulated verifier V, this event occurs with probability $\frac{1}{m_i} \cdot \frac{q_i}{\ell_i}$.
- In Step 1, $k_i \neq c_i$ *and* $j_i \in J_i$. In addition, in Step 2, $k_i = c_i'$ *and* the jth location is chosen from J_i (i.e. from the set of all locations queried by the verifier). This event occurs with probability $(1 - \frac{1}{m_i}) \cdot \frac{q_i}{\ell_i} \cdot \frac{1}{m_i - 1} \cdot \frac{1}{q_i} = \frac{1}{m_i} \cdot \frac{1}{\ell_i}$.

All in all, we have that the probability that this location is queried is

$$\frac{1}{m_i} \cdot \frac{q_i}{\ell_i} + \frac{1}{m_i} \cdot \frac{1}{\ell_i} = \frac{q_i + 1}{m_i \cdot \ell_i} = \frac{1}{\prod_{i=1}^{t} \ell_i}$$

Therefore, each bit in $\overline{\pi}$ is queried with equal probability.

Strong canonical soundness

Let α be the strongness parameter of the multi-piece verifier V. We show that the (single-piece) PCPP $\overline{V}$ has strongness parameter $\alpha/3$. Fix explicit input x, input oracle y and proof oracle $\overline{\pi} = (\overline{\pi_1}, \ldots, \overline{\pi_t})$, where $\overline{\pi_i}$ is purported to contain m_i copies of the ith canonical proof-piece for $(x; y)$. If $R(x)$ is empty, then the emulated verifier (and therefore the single-piece verifier) rejects with probability $\alpha \cdot 1$ for any choice of (alleged) copies c_i, so we may focus on the case that $R(x) \neq \emptyset$. Let $y' \in R(x)$ be a minimizer of ρ, defined

$$\rho \coloneqq \max\left(\delta(y, y'), \delta\left(\overline{\pi_1}, \overline{\Pi}_1(x; y')\right), \ldots, \delta\left(\overline{\pi_t}, \overline{\Pi}_t(x; y')\right)\right) \geq \max\left(\delta(y, y'), \delta\left(\overline{\pi}, \overline{\Pi}(x; y')\right)\right) \quad (1)$$

It suffices to show that the verifier rejects with probability at least $\frac{\alpha}{3} \cdot \rho$. Assume wlog that the maximum in the left hand side of Equation (1) is obtained in the first proof-piece oracle, so $\rho = \max\left(\delta(y, y'), \delta(\overline{\pi}_1, \overline{\Pi}_1(x; y'))\right)$. We proceed by examining the case that the first proof-piece has noticeable inconsistency, and the case where it is almost entirely consistent.

Case 1: The first proof-piece list is noticeably inconsistent: $\mathbb{E}_{c_1 \neq c_1' \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^{c_1'})\right] \geq \rho/3$. Smoothness of the emulated verifier implies that a uniformly random location from the set of locations queried in $\overline{\pi}_1$ is distributed uniformly in $[\ell_i]$ (as detailed in Appendix B.2). Hence, the probability that the consistency check rejects equals the expected distance between two distinct (alleged) copies sampled uniformly, which is assumed to be at least $\rho/3$.

Case 2: The first proof-piece list is almost entirely consistent: $\mathbb{E}_{c_1 \neq c_1' \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^{c_1'})\right] < \rho/3$. By an averaging argument, there exists a "typical" copy such that all other (alleged) copies in first-proof piece list are close to it in expectation; namely, there is an $a \in [m_1]$ such that $\mathbb{E}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^{a})\right] \leq \rho/3$. We argue that since the alleged copies are close to the typical copy (in expectation), the multi-piece verifier rejects with probability similar to that of the strong canonical single-piece verifier given the typical copy as proof.

For each c_1, let $y^{c_1} \in R(x)$ be a minimizer of $\max\left(\delta(y, y^{c_1}), \delta(\pi_1^{c_1}, \Pi_1^{c_1})\right)$ where $\Pi_1^{c_1} := \Pi_1(x; y^{c_1})$. Using strong canonicality of the emulated verifier, we can lower-bound the probability that the single-piece verifier rejects in the emulation check by

$$\begin{aligned}
&\mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\alpha \cdot \max\left(\delta(y, y^{c_1}), \delta(\pi_1^{c_1}, \Pi_1^{c_1})\right)\right] \\
&\quad \geq \alpha \cdot \mathop{\mathbb{E}}_{c_1}\left[\max\left(\delta(y, y^{c_1}), \delta(\pi_1^{a}, \Pi_1^{c_1})\right)\right] - \alpha \cdot \mathop{\mathbb{E}}_{c_1}\left[\delta(\pi_1^{c_1}, \pi_1^{a})\right] \\
&\quad \geq \alpha \cdot \max\left(\delta(y, y^{a}), \delta(\pi_1^{a}, \Pi_1^{a})\right) - \alpha \cdot \frac{\rho}{3} \qquad (2)
\end{aligned}$$

where the second inequality used the minimality of y^a, which means that for any c_1 it holds that $\max\left(\delta(y, y_1^c), \delta(\pi_1^a, \Pi_1^{c_1})\right) \geq \max\left(\delta(y, y^a), \delta(\pi_1^a, \Pi_1^a)\right)$. We can also lower-bound the distance of the typical copy π_1^a from its corresponding canonical proof piece Π_1^a by

$$\begin{aligned}
\delta(\pi_1^a, \Pi_1^a) &\geq \mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \Pi_1^a)\right] - \mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^a)\right] \qquad (3) \\
&\geq \mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \Pi_1^a)\right] - \frac{\rho}{3} = \delta\big(\overline{\pi}_1, \overline{\Pi}_1(x; y^a)\big) - \frac{\rho}{3}
\end{aligned}$$

Combining Equations (2) and (3), we have that the verifier rejects with probability at least

$$\alpha \cdot \max\left(\delta(y, y^a), \delta\big(\overline{\pi}_1, \overline{\Pi}_1(x; y^a)\big) - \frac{\rho}{3}\right) - \alpha \cdot \frac{\rho}{3} \geq \alpha \cdot \rho - \alpha \cdot \frac{2\rho}{3} = \alpha \cdot \frac{\rho}{3}$$

where the inequality is because $y^a \in R(x)$ is not necessarily the minimizer of Equation (1), i.e. it could be that $y^a \neq y'$ (and then $\max\left(\delta(y, y^a), \delta(\overline{\pi}_1, \overline{\Pi}_1(x; y^a))\right) \geq \rho$). ◀

3 Composing smooth strong canonical PCPPs

In this section, we adapt the composition theorem of Ben-Sasson et al. [6] to the strong canonical setting.

We can think of a run of nonadaptive PCPP verifier as a two-step process: first, it tosses some random coins and generates a *residual (decision) circuit* and *query locations* based on the coins it tossed, and then it queries its oracles and feeds their answers to the residual circuit, accepting or rejecting accordingly. Hence, the verifier accepts if and only if the residual circuit and oracle answers are in CircuitVal. *PCPP composition* replaces the naive verification of this claim of membership (in CircuitVal) by a probabilistic verification. That is, an *inner verifier* probabilistically checks that the oracles' answers satisfy the *outer verifier's* residual decision circuit. The resulting *composite verifier* accepts or rejects according to the inner verifier's decision.

The strong inner verifier rejects with probability that is proportional to the distance of the oracles' answers from satisfying the outer residual circuit. As such, this distance should reflect the distance of the outer oracles (in their entirety) from correct ones. In other words, if the outer oracles are far from correct ones, the outer verifier's queries should not only be rejected, but be *far* from being accepted by its residual circuit. In such a case we say that the outer verifier is *robust*.

As in the composition of [6], when it comes to randomness and query complexities, the composite verifier enjoys the best of both worlds: broadly speaking, its query complexity is inherited from the inner verifier, and its randomness complexity is (mostly) determined by the outer verifier. So, composing an outer verifier of low randomness complexity with an inner verifier that issues a few queries yields a composite verifier with low randomness *and* query complexities. The contribution of this section is in showing that, in addition, such composition can be made to preserve smoothness and strong canonicality.

3.1 Strong canonical robust PCPPs

First, we describe the run of a nonadaptive verifier as a two-step process: first sampling random coins, then querying its oracles and computing a decision. Formally,

▶ **Definition 3.1** (PCPPs, restated)**.** *Given explicit input x and oracle access input y and proof π, a (nonadaptive) PCPP verifier for relation R of randomness complexity $r(n)$ and query complexity $q(n)$ runs as follows:*

1. Sample. *The verifier uniformly samples a* coin sequence $c \in \{0,1\}^{r(|x|)}$*. Based on x and c, the verifier generates* query locations $I := I_c := (i_1, \ldots, i_{q(|x|)})$ *and* residual circuit $D := D_c$*. Note that I contains the locations of queries to both y and π.*
2. Query and compute. *The verifier queries oracles y and π according to the query locations I. Denoting the answers to these queries by $y\pi[I]$, the verifier then computes $D(y\pi[I])$ and outputs 1 ("Yes") if and only if $y\pi[I]$ satisfies D.*

The event that the verifier V accepts, i.e. "$V^{y,\pi}(x) = 1$", is equal by definition to "$y\pi[I_c] \in \mathtt{Sat}(D_c)$", where $\mathtt{Sat}(D_c)$ denotes the set of satisfying inputs of D_c, and the randomness in both events being the coin sequence c. The decision complexity *$d(n)$ is the maximal size of a residual circuit generated when the verifier is given explicit input of length n.*

Strong canonical *robustness* guarantees that the expected distance of the oracles' answers from satisfying the residual circuit is proportional to the distance of these oracles from being correct, and is formally defined as follows.

▶ **Definition 3.2** (Strong canonical robust PCPPs)**.** *A strong canonical PCPP V for relation R with canonical proof strategy $\Pi\colon \{0,1\}^* \to \{0,1\}^*$ is a strong canonical* robust *PCPP* (RPCPP) *with strongness parameter $\alpha \in (0,1]$ if, in addition to the conditions of Definition 1.8, it satisfies* strong canonical robust soundness*:*

- *For any $(x;y)$ such that $(x;y) \notin R$ it holds that $\Pi(x;y) = \emptyset$. When given explicit input x, input oracle y and proof oracle π, the expected distance of the bits queried by V from being accepted by the residual circuit is at least*

$$\alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y,y'), \delta(\pi, \Pi(x;y'))\right)\} \tag{4}$$

That is,

$$\mathop{\mathbb{E}}_{c}\left[\delta(y\pi[I_c], \mathtt{Sat}(D_c))\right] \geq \begin{cases} \alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y,y'), \delta(\pi, \Pi(x;y'))\right)\} & \text{if } R(x) \neq \emptyset \\ \alpha & \text{if } R(x) = \emptyset \end{cases}$$

▶ Remark 3.3. This definition differs from a natural adaptation of the original definition of robustness ([6, Definition 2.6]) in that it requires the *expected* distance to be large, rather than require the distance be large with high probability. Markov's inequality implies that Definition 3.2 is stronger.

3.2 The composition theorem

Following the two-step description of a run of a PCPP verifier in Definition 3.1, a PCPP verifier accepts explicit input x, input oracle y and proof oracle π if and only if the answers received from its proof oracle (denoted $y\pi[I]$) satisfy its residual circuit D. In a nutshell, PCPP composition is done by replacing the verification of the claim "$y\pi[I]$ *satisfies* D" with a probabilistic verification by an *inner PCPP verifier*.

Before turning to the theorem and its proof, we define an additional property we require from outer verifiers.

▶ **Definition 3.4** (Residual circuit distance). *For some constant $\Delta > 0$, we say that a PCPP verifier V has* residual circuit distance Δ *if for any input x and coin sequence c, any two distinct inputs satisfying the residual circuit D_c are at least Δ-far apart.*

In later sections, we show that the outer PCPP whose composition yields Theorem 1.12 satisfies this additional property for some constant $\Delta > 0$. Hence, it suffices to prove the composition theorem for PCPPs with constant residual circuit distance.

▶ **Theorem 3.5.** *Assume there are $\alpha_{\text{out}}, \alpha_{\text{in}}, \Delta \in (0,1]$ and $r_{\text{out}}, r_{\text{in}}, d_{\text{out}}, d_{\text{in}}, q_{\text{in}} \colon \mathbb{N} \to \mathbb{N}$ such that the following holds:*

- *The relation R has a smooth strong canonical RPCPP, denoted V_{out}, with residual circuit distance Δ, strongness parameter α_{out} and randomness and decision complexities r_{out} and d_{out} respectively.*
- CircuitVal *has a strong canonical PCPP, denoted V_{in}, with strongness parameter α_{in} and randomness, query and decision complexities r_{in}, q_{in} and d_{in} respectively.*

Then, the relation R has a strong canonical PCPP, denoted V_{comp}, with the following properties:

- *Randomness complexity $r_{\text{out}} + r_{\text{in}} \circ d_{\text{out}}$.*
- *Query complexity $q_{\text{in}} \circ d_{\text{out}}$.*
- *Decision complexity $d_{\text{in}} \circ d_{\text{out}}$*
- *Strongness parameter $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \Delta/4$*

Furthermore, if V_{in} is smooth then R has a smooth strong canonical PCPP with strongness shrinking by a third (to $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \Delta/12$). If V_{in} is robust (resp. has residual circuit distance) then V_{comp} is robust (resp. has residual circuit distance).

Notice that smoothness of the outer verifier is used for strong canonicality of the composite verifier.

Proof of Theorem 3.5. We affix the term *outer*, *inner* or *composite* when discussing components of the *outer*, *inner* or *composite* verifiers. For example, an *inner canonical proof* is a proof obtained from the canonical proof strategy of the inner PCPP.

We start by describing the canonical proof strategy of the composite PCPP, which consists of two proof-piece oracles: an *outer proof-piece* denoted Π, and an *inner proof-piece* denoted T. For any explicit input x and input oracle y with $(x; y) \in R$, the outer proof-piece is the outer canonical proof that $(x; y) \in R$ (for V_{out}), and the inner proof-piece is the concatenation (over all possible coin sequences c of the outer verifier) of the inner canonical proof that $y\Pi[I_c]$ satisfies D_c (for V_{in}), which we denote by T_c. With these in mind, we proceed by describing the composite verifier.

▶ **Algorithm 3.5.1** (PCPP composition [6, Section 2.4]). The composite PCPP system has verifier V_{comp} that takes explicit input x, input oracle y, and two proof-piece oracles: an outer proof-piece π, and an inner proof-piece $\tau = (\tau_c)_c$, with c ranging over all possible coin sequences of the outer verifier. It runs as follows:

1. Emulates the *Sample* step of the outer verifier, obtaining a coin sequence c, outer query locations I_c and outer residual circuit D_c. (V_{comp} does not issue any queries yet!)
2. Emulates the inner verifier with explicit input D_c, input oracle $y\pi[I_c]$ and proof oracle τ_c. V_{comp} accepts if and only if the inner verifier accepted.

Since Algorithm 3.5.1 is the same composition used in the proof of [6, Theorem 2.7], the composite PCPP enjoys all properties described there, and in particular it is a PCPP with the required complexities. Canonical completeness follows from the canonical completeness of the outer and inner verifiers, and so we turn to prove strong canonical soundness. Then,

if V_{in} is smooth then V_{comp} is two-piece smooth, and we may apply Lemma 2.2 to obtain a smooth strong canonical composite PCPP for R while losing an additional factor $1/3$ in strongness.

Fix an explicit input x, input oracle y, and alleged proof-piece oracles and π and τ, where π corresponds to the outer proof-piece and $\tau \coloneqq (\tau_c)_c$ corresponds to the inner proof-piece. If $R(x)$ is empty then we may refer to the standard soundness analysis of the composite PCPP, so we focus on the case that $R(x)$ is nonempty. Let y' be a satisfying input that minimizes the maximal distance between the given oracles and correct oracles; that is, the maximum between $\delta(y, y')$, $\delta(\pi, \Pi)$ and $\delta(\tau, T)$, where Π and T are the canonical outer and inner proof-pieces for y'. Denote the distance between the given oracles and the correct oracles by $\delta_y \coloneqq \delta(y, y')$, $\delta_\pi \coloneqq \delta(\pi, \Pi)$ and $\delta_\tau \coloneqq \delta(\tau, T) = \mathbb{E}_c\left[\delta(\tau_c, T_c)\right]$. We shall show that the verifier rejects with probability $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \Delta \cdot \max\left(\delta_y, \delta_\pi, \delta_\tau\right)/4$. The analysis considers two cases.

Case 1: $\max\left(\delta_y, \delta_\pi\right) \geq \frac{\Delta}{4} \cdot \delta_\tau$. For any fixed outer coin sequence c, the strong canonical inner verifier rejects the circuit D_c, input oracle $y\pi[I_c]$ and proof oracle τ_c with probability $\alpha_{\text{in}} \cdot \delta(y\pi[I_c], \texttt{Sat}(D_c))$. The rejection probability of the composite verifier equals the expected rejection probability of the inner verifier (over random c), therefore the composite verifier rejects with probability at least $\mathbb{E}_c\left[\alpha_{\text{in}} \cdot \delta(y\pi[I_c], \texttt{Sat}(D_c))\right]$ which, by the robust strong canonicality of the outer verifier, is at least $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \max\left(\delta_y, \delta_\pi\right) \geq \alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \frac{\Delta}{4} \cdot \max\left(\delta_y, \delta_\pi, \delta_\tau\right)$.

Case 2: $\max\left(\delta_y, \delta_\pi\right) < \frac{\Delta}{4} \cdot \delta_\tau$. As a warm-up, consider the case that $\delta_y = \delta_\pi = 0$. In this case, $y\pi[I_c]$ satisfies D_c for any outer coin sequence c. Recall that the outer verifier has constant residual circuit distance, so all other inputs that satisfy D_c are far from $y\pi[I_c]$, and therefore the inner verifier's strong canonicality implies that the composite verifier rejects with probability $\Omega(\delta(\tau_c, T_c))$. Taking expectation over the coin sequence c, we have that the composite verifier rejects with probability $\Omega(\delta_\tau) = \Omega(\max\left(\delta_y, \delta_\pi, \delta_\tau\right))$.
In the actual analysis, δ_y and δ_π are not necessarily zero, but the general ideas of the warm-up are still applicable: using the smoothness of the outer verifier, we can relate $\mathbb{E}_c\left[\delta(y\pi[I_c], \delta(y'\Pi[I_c]))\right]$ with δ_y and δ_π. Since δ_y and δ_π are assumed to be $O(\delta_\tau)$, then for sufficiently many c's it holds that $y\pi[I_c]$ is close to $y'\Pi[I_c]$. Then, the constant residual circuit distance of the verifier implies that for these c's the inner verifier rejects with probability proportional to $\delta(\tau_c, T_c)$. Details follow.
We say that a fixed coin sequence c is *good* if $y\pi[I_c]$ is $\Delta/2$-close to $y'\Pi[I_c]$. Strong canonicality of the inner verifier means that the probability it rejects circuit D_c, input oracle $y\pi[I_c]$ and proof oracle τ_c is at least

$$\alpha_{\text{in}} \cdot \min_{s \in \texttt{Sat}(D_c)} \left\{\max\left(\delta(y\pi[I_c], s), \delta(\tau_c, T_c(D_c; s))\right)\right\}$$

The outer verifier has residual circuit distance Δ, so for good coins c, $\delta(y\pi[I_c], s) \geq \Delta/2$ for all $s \in \texttt{Sat}(D_c) \setminus \{y'\Pi[I_c]\}$. Therefore, for good c's, $\max\left(\delta(y\pi[I_c], s), \delta(\tau_c, T_c(D_c; s))\right)$ is at least $\delta(\tau_c, T_c)$ when taking $s = y'\Pi[I_c]$, and is at least $\Delta/2$ when s satisfies D_c but differs from $y'\Pi[I_c]$. Hence,

$$\mathbb{P}\left[V_{\text{in}}^{\tau_c}(D_c) = 0 \,|\, c \text{ good}\right] \geq \alpha_{\text{in}} \cdot \min\left\{\Delta/2, \delta(\tau_c, T_c)\right\} \geq \frac{\alpha_{\text{in}} \cdot \Delta}{2} \cdot \delta(\tau_c, T_c) \tag{5}$$

Now, we show that since δ_y and δ_π are $O(\delta_\tau)$, then the expected distance between inner proofs over good c's upper-bounds the distance between all inner proofs δ_τ (up to constants). Smoothness of the outer verifier implies that sampling a random coin

sequence c and location j in I_c is the same as uniformly sampling from $[|y\pi|]$ (as detailed in Appendix B.2). Therefore,

$$\delta(y\pi, y'\Pi) \coloneqq \mathop{\mathbb{P}}_{k}[y\pi[k] \neq y'\Pi[k]] = \mathop{\mathbb{P}}_{j,c}[y\pi[I_c][j] \neq y'\Pi[I_c][j]] = \mathop{\mathbb{E}}_{c}[\delta(y\pi[I_c], y'\Pi[I_c])]$$

Thus,

$$\delta(y\pi, y'\Pi) \geq \frac{\Delta}{2} \cdot \mathop{\mathbb{P}}_{c}\left[\delta(y\pi[I_c], y'\Pi[I_c]) > \frac{\Delta}{2}\right] = \frac{\Delta}{2} \cdot \mathop{\mathbb{P}}_{c}[c \neg\text{good}]$$

Note that $\max(\delta_y, \delta_\pi) \geq \delta(y\pi, y'\Pi)$, so by the assumption that $\max(\delta_y, \delta_\pi) < \Delta \cdot \delta_\tau/4$ we have that $\delta_\tau/2 \geq \mathbb{P}[c \neg\text{good}]$. In addition,

$$\delta_\tau = \mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c)] \leq \mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c) \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] + \mathop{\mathbb{P}}_{c}[c \neg\text{good}]$$

Which implies

$$\mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c) \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] \geq \delta_\tau/2 \tag{6}$$

Using Equations (5) and (6), the probability that the composite verifier rejects the given input and proof is at least

$$\mathop{\mathbb{E}}_{c}[\mathbb{P}[V_{\text{in}}^{\tau_c}(x) = 0] \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] \geq \frac{\alpha_{\text{in}} \cdot \Delta}{2} \cdot \mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c) \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] \geq \frac{\alpha_{\text{in}} \cdot \Delta}{4} \cdot \delta_\tau$$

If V_{in} is a strong canonical *robust* PCPP, then we note that the lower bounds on the *rejection probability* hold for the *expected distance of the bits read from* $(y\pi[I_c], \tau_c)$ *from satisfying* V_{comp}*'s residual circuit.*[15] As for residual circuit distance: any two strings satisfying the composite residual circuit satisfy the inner residual circuit, so the residual circuit distance of the inner verifier is inherited by the composite verifier. ◀

3.3 Composing the construct of Theorem 1.12

Now that we know *how* to compose PCPPs, let's talk about *which* PCPPs we compose. For now, we postulate two smooth strong canonical (R)PCPPs of certain complexities and reckon that their composition yields a PCPP of logarithmic randomness and constant queries (their actual construction and analysis constitutes the rest of this work).

▶ **Proposition 3.6** (Hadamard-based PCPP)**.** *There exists a smooth strong canonical PCPP for* CircuitVal *of quadratic randomness and constant query complexities.*

▶ **Proposition 3.7** (Reed–Muller-based RPCPP)**.** *There exists a smooth strong canonical robust PCPP for* CircuitVal *of logarithmic randomness complexity, polylogarithmic decision complexity, and constant residual circuit distance.*

As in [2, 6], the Reed–Muller-based RPCPP is composed with itself to obtain a smooth strong canonical RPCPP of logarithmic randomness complexity, poly log log decision complexity and constant residual circuit distance. The resulting RPCPP is then composed (as an outer verifier) with an inner Hadamard-based PCPP to obtain a smooth strong canonical PCPP of logarithmic randomness and constant query complexities, proving Theorem 1.12.

[15] To transform the multi-piece robust PCPP to a single-piece robust PCPP, we note that Lemma 2.2 holds for robust PCPPs as well: for strong canonicality, rather than analyzing the *rejection probability of the verifier*, we can analyze the *expected distance from being accepted by the verifier* (using exactly the same reasoning).

4 The Hadamard-based smooth strong canonical PCPP

We now turn to the actual construction of smooth strong canonical PCPPs, starting with the Hadamard-based PCPP of [2, Section 4] as presented in [20, Chapter 4].

4.1 Algebrization and the Hadamard code

A circuit and its input can be expressed as a system of quadratic equations and an assignment, such that the input satisfies the circuit if and only if the assignment satisfies the equation system. The Hadamard-based PCPP verifier capitalizes on this fact, with the proof for an input consisting of the truth tables of evaluations of all (linear and) quadratic equations on the assignment corresponding to the input. First, let's take a more detailed look at this correspondence, which is sometimes called an *algebrization* of the combinatorial problem of circuit valuation to the algebraic problem of quadratic equation valuation.

▶ **Definition 4.1** (Computational extension). *Let C be a circuit of size n with n_0 input gates and let $y \in \{0,1\}^{n_0}$. The* computational extension of y w.r.t C *is the string corresponding to the values output by each gate of C when computing y, and is denoted $y_C \in \{0,1\}^n$. That is, $y_C[i]$ is the output of the ith gate of C when computing input y. Assuming wlog that the first n_0 gates of C are its input gates, it holds that $y = y_C[1] \cdots y_C[n_0]$.*

▶ **Definition 4.2** (Outer product). *The* outer product *of vectors $u, v \in \{0,1\}^n$, denoted $u \otimes v$, is the n^2 dimensional vector obtained by "flattening" the $n \times n$ matrix whose entry in the ith column and jth row is $u[i] \cdot v[j]$. That is, $u \otimes v[(i-1)n + j] := u[i] \cdot v[j]$ for all $i, j \in [n]$.*

▶ **Proposition 4.3.** *There exists a polynomial-time computable mapping that maps circuit C of size n with m input bits to an $n \times n^2$ matrix A_C and vector $b_C \in \{0,1\}^n$ such that input $y \in \{0,1\}^m$ satisfies C if and only if $y_C \in \{0,1\}^n$ satisfies $A_C(y_C \otimes y_C) = b_C$.*

Proof sketch. Let C be a circuit of size n that has n_0 input gates. The reduction computes $b \in \{0,1\}^n$ and $a_1, \ldots, a_n \in \{0,1\}^{n^2}$ according to Table 1, and lets A_C be the matrix whose ith row is a_i. ◀

Table 1 Mapping of gates to equations. $e_{i,j} \in \{0,1\}^{n^2}$ is all zeroes except for coordinate $(i-1)n + j$.

Gate	First input gate	Second input gate	Gate type	Reduction output
i	j	k	`AND`	$a_i := e_{i,i} + e_{j,k}$ $b_i := 0$
			`OR`	$a_i := e_{i,i} + e_{j,j} + e_{k,k} + e_{j,k}$ $b_i := 0$
			`NOT`	$a_i := e_{i,i} + e_{j,j}$ $b_i := 1$
			`OUTPUT`	$a_i := e_{i,i}$ $b_i := 1$
			`INPUT`	$a_i := \overline{0}$ $b_i := 0$

The *Hadamard encoding* of a string $y \in \{0,1\}^n$ is the evaluation of all linear equations (over $\mathbb{F}_2$) on n variables on the assignment y. Formally,

▶ **Definition 4.4.** *The* Hadamard encoding *of a vector $y \in \{0,1\}^n$ is the function* $\mathrm{Had}_y \colon \{0,1\}^n \to \{0,1\}$ *given by* $\mathrm{Had}_y(z) \coloneqq y \cdot z = \sum_{i=1}^{n} y[i]z[i] \mod 2$.

Recall some useful facts about this encoding:

▶ **Fact 4.5.** *For all $n \in \mathbb{N}$:*

Distance. *A function $f \colon \{0,1\}^n \to \{0,1\}$ is linear if and only if there exists $y \in \{0,1\}^n$ such that $f = \mathrm{Had}_y$. For all $y \neq y'$ it holds that $\delta(\mathrm{Had}_y, \mathrm{Had}_{y'}) = 1/2$, associating Had_y with its 2^n-bit-long truth table. That is to say that the Hadamard code has* relative distance $1/2$.

Strong Testability. *Consider a test that given $f \colon \{0,1\}^n \to \{0,1\}$ uniformly samples $x, x' \in \{0,1\}^n$ and accepts if and only if*

$$f(x) + f(x') = f(x + x')$$

As shown in [9], if f is δ-far from all Hadamard codewords then the test rejects with probability at least $\min(\delta/2, 1/6)$. *Note that the test makes three queries to f and tosses $2n$ random coins.*

Self-correction. *Consider a self-correction procedure that given $f \colon \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$, uniformly samples $r \in \{0,1\}^n$ and outputs $f(x+r) - f(r)$. It holds that if f is δ-close to the Hadamard codeword $\widetilde{f}$ then for all x the procedure outputs $\widetilde{f}(x)$ with probability at least $1 - 2\delta$. Note that the procedure makes two queries to f and tosses n random coins.*

4.2 The PCPP and its analysis

▶ **Algorithm 4.6** (The Hadamard-based PCPP [2])**.** The canonical proof-piece strategies of circuit C and satisfying input y are the Hadamard encodings of y_C and $y_C \otimes y_C$ (the computational extension of y, and the outer product of the latter with itself, respectively). That is, $\Pi_1(C; y) \coloneqq \mathrm{Had}_{y_C}$ and $\Pi_2(C; y) \coloneqq \mathrm{Had}_{y_C \otimes y_C}$. The *verifier* takes explicit input C and is given access to input oracle y and alleged proof-piece oracles π_1 and π_2. It performs the following tests, and accepts if and only if all of them passed:

1. *Strong codeword checks.* Perform the strong codeword test of Fact 4.5 on both π_1 and π_2.
2. *Oracle-oracle consistency check:* Check that the assignment allegedly encoded by π_2 is the outer product of the assignment allegedly encoded by π_1; that is, uniformly sample $u, v \in \{0,1\}^n$ and check that $\pi_1(u)\pi_1(v) = \pi_2(u \otimes v)$, using self-correction on π_2.
3. *Satisfaction check.* Compute the quadratic equation system (A_C, b_C) corresponding to circuit C via Proposition 4.3, and check that the assignment allegedly encoded by π_2 satisfies the quadratic equation system (A_C, b_C) by checking that it satisfies a random linear combination of equations; that is, uniformly sample $w \in \{0,1\}^n$ and check that $\pi_2(w^\top A_C) = \mathrm{Had}_{b_C}(w)$, using self-correction on π_2. Note that the verifier computes A_C and Had_{b_C} based only on the circuit C, which is given explicitly.
4. *Oracle-witness consistency check.* Check that π_1 encodes the computational extension of y; that is, uniformly sample $i \in [m]$ and check that $y[i] = \pi_1(e_i)$ using self-correction on π_1, where $e_i \in \{0,1\}^m$ is the unit vector $0^{i-1}10^{n-i}$.

It is known that Algorithm 4.6 is a PCPP for CircuitVal with polynomial randomness complexity and constant query complexity (see [2] or later reformulation in [20, Section 4.1]). It is smooth (as a two-piece PCPP), as only a single uniformly random location of y is queried,

and locations queried in π_1 and π_2 are all uniformly random. We show that it is strong canonical, and can therefore be transformed into a single-piece smooth strong canonical PCPP for CircuitVal using Lemma 2.2, thereby proving Proposition 3.6.

Proof. Fix circuit C, input oracle y and proof-piece oracles π_1 and π_2. We show that the verifier rejects with probability at least

$$\frac{5}{72} \cdot \min_{y' \in \mathtt{Sat}(C)} \{\max\left(\delta(y, y'), \delta(\pi_1, \Pi_1(C; y')), \delta(\pi_2, \Pi_2(C; y'))\right)\} \tag{7}$$

where $\mathtt{Sat}(C)$ denotes the set of satisfying inputs of C.

Let $\widetilde{\pi_1}$ and $\widetilde{\pi_2}$ be the Hadamard codewords closest to π_1 and π_2 respectively. Let $\widetilde{z} \in \{0,1\}^n$ and $\widetilde{Z} \in \{0,1\}^{n \times n}$ be the decodings of $\widetilde{\pi_1}$ and $\widetilde{\pi_2}$ respectively, so that $\widetilde{\pi_1} = \mathrm{Had}_{\widetilde{z}}$ and $\widetilde{\pi_2} = \mathrm{Had}_{\widetilde{Z}}$. Fact 4.5 implies that if π_1 or π_2 are 1/12-far from $\widetilde{\pi_1}$ or $\widetilde{\pi_2}$ (respectively) then one of the linearity tests reject with probability at least 1/24. Hence, we may assume $\max\left(\delta(\pi_1, \widetilde{\pi_1}), \delta(\pi_2, \widetilde{\pi_2})\right) \leq 1/12$. Let $\widetilde{y}$ be the n_0-bit-long prefix of $\widetilde{z}$, which corresponds to the values $\widetilde{z}$ gives to the input gates of C.

We start by reckoning that if $\widetilde{\pi_1}, \widetilde{\pi_2}$ aren't the canonical proof-pieces for $(C; \widetilde{y})$ then rejection occurs with high probability, and then show that if they *were* the canonical proof-pieces then the oracle-witness consistency check (resp. strong codeword check) rejects the input oracle (resp. proof-piece oracles) with probability proportional to $\delta(y, \widetilde{y})$ (resp. $\max\left(\delta(\pi_1, \widetilde{\pi 1}), \delta(\pi_2, \widetilde{\pi_2})\right)$).

Observe that $\widetilde{\pi_1}, \widetilde{\pi_2}$ are the canonical proof-pieces of $(C; \widetilde{y})$ if and only if the following conditions hold:

- $\widetilde{y}$ satisfies the circuit C. (Otherwise, $\widetilde{y}$ has no canonical proof.)
- $\widetilde{\pi_1}$ is the Hadamard encoding of the computational extension of $\widetilde{y}$, denoted $\widetilde{y}_C$.
- $\widetilde{\pi_2}$ is the Hadamard encoding of $\widetilde{y}_C \otimes \widetilde{y}_C$.

By Proposition 4.3, we have that $\widetilde{\pi_1}, \widetilde{\pi_2}$ are the canonical proof-pieces of $(C; \widetilde{y})$ if and only if $\widetilde{Z} = \widetilde{z} \otimes \widetilde{z}$ and $\widetilde{z}$ satisfies the equation system (A_C, b_C). With this observation in mind, we examine the following cases.

Case 1: $\widetilde{\pi_1}, \widetilde{\pi_2}$ are not the canonical proof-pieces of $\widetilde{y}$.

We claim that rejection occurs with high probability. Indeed, by the foregoing observation, one of two sub-cases must hold:

Case 1.1: $\widetilde{Z} \neq \widetilde{z} \otimes \widetilde{z}$.

In this case, oracle-oracle consistency check rejects with high probability: The inequation $\widetilde{\pi_1}(u)\widetilde{v} \neq \widetilde{\pi_2}(u \otimes v)$ holds for at least a quarter of possible $u, v \in \{0,1\}^n$,[16] therefore $\pi_1(u)\pi_1(u) \neq \widetilde{\pi_2}(u \otimes v)$ with probability at least $1/4 - 2 \cdot \delta(\pi_1, \widetilde{\pi_1}) \geq 1/12$. The self-corrected query to π_2 gives the value of $\widetilde{\pi_2}$ with probability $1 - 2 \cdot (\delta(\pi_2, \widetilde{\pi_2})) \geq 5/6$, so we have that $\pi_1(u)\pi(v) \neq \pi_2(u \otimes v)$ occurs with probability at least $(1/12) \cdot (5/6) = 5/72$.

Case 1.2: $\widetilde{Z} = \widetilde{z} \otimes \widetilde{z}$ but $\widetilde{z}$ does not satisfy the quadratic equation system (A_C, b_C).

In this case, it is the satisfaction check that rejects with high probability: By Proposition 4.3 it holds that $A_C \widetilde{Z} \neq b_C$, and notice that $\mathrm{Had}_{\widetilde{Z}}(w^\top A_C)(w) = \mathrm{Had}_{A_C \widetilde{Z}}(w)$ for all $w \in \{0,1\}^n$. Therefore, the inequation $\widetilde{\pi_2}(w^\top A_C)(w) \neq \mathrm{Had}_{b_C}(w)$ holds with probability 1/2 over the choice of a uniformly random w, as the Hadamard code has relative distance 1/2. Accounting for a single self-corrected query, the satisfaction test rejects with probability at least $(1/2) \cdot (1 - 2\delta(\pi_2, \widetilde{\pi_2})) \geq 5/12$.

[16] This follows from the fact that for two distinct $n \times n$ matrices A and B, the inequation $u^\top A v \neq u^\top B v$ for at least a quarter of all possible $u, v \in \{0,1\}^n$.

Case 2: $\widetilde{\pi_1}, \widetilde{\pi_2}$ are the canonical proof-pieces of $\widetilde{y}$.

In particular, $\widetilde{y}$ is a satisfying input for circuit C. By definition of the Hamming distance and accounting for self-correction, the oracle-witness test rejects with probability $\delta(y, \widetilde{y})(1 - 2\delta(\pi_1, \widetilde{\pi_1})) \geq 5\delta(y, \widetilde{y})/6$. Additionally the linearity tests reject with probability greater than both $\delta(\pi_1, \widetilde{\pi_1})/2$ and $\delta(\pi_2, \widetilde{\pi_2})/2$. All in all, in this case the verifier rejects with probability at least

$$\max\left(\frac{5}{6}\cdot\delta(y,\widetilde{y}), \frac{1}{2}\cdot\delta(\pi_1,\widetilde{\pi_1}), \frac{1}{2}\cdot\delta(\pi_2,\widetilde{\pi_2})\right) \geq$$
$$\frac{5}{72}\cdot\min_{y'\in\mathtt{Sat}(C)}\{\max\left(\delta(y,y'), \delta(\pi_1, \Pi_1(C;y')), \delta(\pi_2, \Pi_2 C; y')\right)\}$$

where the inequality is justified by the fact that $\widetilde{y}$ satisfies circuit C (however it does not necessarily minimize the expression on the right hand side). ◀

5 The Reed–Muller-based smooth strong canonical robust PCPP

The construction of the Reed–Muller-based RPCPP is more involved than that of the Hadamard-based PCPP. We follow its presentation in [20, Part 1] (a reorganization of [6]), noting that some components mentioned therein appeared in prior works (for example [3, 2]). We find this presentation appealing as it breaks the construction down to four bite-sized steps. Essentially, we prove that the first step yields a PCPP that is smooth and strong canonical, and that the transition between each step preserves smoothness and strong canonicality. Along the way, we will also keep track of the residual circuit distance (see Definition 3.4) and observe that it is constant.

So how do we go about proving this? First, it's worth noting that the intermediate PCPPs are multi-piece and, more importantly, issue queries to oracles of larger, non-binary alphabets. Don't worry, eventually (Section 5.4) we show how to reduce alphabet size back to binary, and of course our old friend Lemma 2.2 will reduce the number of pieces to one.[17] Anyways, the first step is constructing a smooth and strong canonical RPCPP for the algebraic problem of determining whether a function is a low-degree polynomial that is identically zero on a subcube (Section 5.1). This problem is closely related to circuit satisfiability: fixing a circuit, inputs can be encoded in a way such that the input satisfies the circuit if and only if its encoding is Zero on Subcube. Section 5.2 capitalizes on this to derive an RPCPP for the problem of circuit satisfiability, which is then transformed to an RPCPP for CircuitVal by adding a consistency test (Section 5.3).

PCPPs over larger alphabets

Though the final result of this section is a (Boolean) smooth and strong canonical RPCPP, the PCPPs used along the way are non-Boolean, meaning that their queries are answered not with bits but with elements of an alphabet of larger size. Furthermore, the intermediate multi-piece PCPPs may have different alphabets for different pieces.

So far, we did not explicitly mention which alphabet was used (indeed, it was always binary) so the previously introduced notions and definitions need not be changed, except for replacing *relative Hamming distance* with the *generalized relative Hamming distance*.

[17] As in Section 3, to transform the multi-piece robust PCPP to a single-piece robust PCPP we note that Lemma 2.2 holds for robust PCPPs as well: for strong canonicality, rather than analyzing the *rejection probability of the verifier*, we can analyze the *expected distance from being accepted by the verifier* (using exactly the same reasoning exactly).

▶ **Definition 5.1** (Generalized relative Hamming metric). *Fix sets $\Sigma_1, \ldots, \Sigma_k$ and let $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$ such that $x_i, y_i \in \Sigma_i$ for all $i \in [k]$. The* generalized (relative) Hamming distance *between x and y is the fraction of locations in which x and y differ, that is $\delta(x, y) \coloneqq \mathbb{P}_i [x_i \neq y_i]$ for a uniformly random $i \in [k]$.*

Note that some of the sets $\Sigma_1, \ldots, \Sigma_k$ may not be distinct. In particular, if $\Sigma_1 = \cdots = \Sigma_k$, then the generalized relative Hamming distance coincides with the relative Hamming distance as defined way back in Section 1.1.1.

▶ Remark 5.2. A letter in the alphabet Σ_i can be represented by $\log |\Sigma_i|$ bits. However, the size of Σ_i is (deliberately) not accounted for in the generalized Hamming distance, therefore the generalized Hamming distance between x and y is not necessarily the same as the Hamming distance between the binary representation of x and y (in which x_i and y_i are each replaced with $\log |\Sigma_i|$ bits for each $i \in [k]$).

All distances referred to in Section 5 are in the generalized Hamming metric.

5.1 A smooth strong canonical RPCPP for Zero on Subcube

In this section we follow [20, Section 5.3.2] in constructing an RPCPP for the algebraic problem of determining whether a function is (close to) a low-degree polynomial that is identically zero on a subcube (formalized in Definition 5.9). Before we turn to the main construction, we present a generalization of a property tester for low-degree polynomials to sequences (vectors) of functions, which will be used in the RPCPP construction. Specifically, given oracle access to a function $f : \mathbb{F}^m \to \mathbb{F}^k$, we test whether it is close to a sequence (vector) of k low-degree polynomials (see Definition 5.3 quoted below).

▶ **Definition 5.3** (Vector-valued low-degree polynomial). *A function $f \colon \mathbb{F}^m \to \mathbb{F}^k$ is a* vector-valued multivariate polynomial of degree at most d *if for all $i \in [k]$ the projection of f to the ith coordinate is a multivariate polynomial of (total) degree at most d, where the projection of f to the ith coordinate is denoted $f_i : \mathbb{F}^m \to \mathbb{F}$ and defined by $f_i(x) \coloneqq f(x)_i$ for all $x \in \mathbb{F}^m$.*

The distance between such vector-valued functions is measured according to the generalized Hamming metric of Definition 5.1; that is, $f : \mathbb{F}^m \to \mathbb{F}^k$ is δ-far from being a low-degree vector-valued polynomial if $\mathbb{P}_x [f(x) \neq \widetilde{f}(x)] > \delta$ for any vector-valued polynomial $\widetilde{f}$ of degree at most d. We stress that $f(x) \neq \widetilde{f}(x)$ when there *exists* i such that $f_i(x) \neq \widetilde{f}_i(x)$.

Recall that the *line* $\mathcal{L}$ through $\mathbb{F}^m$ with *intercept* $x \in \mathbb{F}^m$ and *slope* $h \in \mathbb{F}^m$ is the set of points $\mathcal{L} \coloneqq \{x + ih : i \in \mathbb{F}\}$. A uniformly random line (through $\mathbb{F}^m$) is obtained by sampling $x, h \in \mathbb{F}^m$ uniformly at random and letting $\mathcal{L}$ be the line with slope h and intercept x. Throughout this section we will use $f[\mathcal{L}]$ to denote the restriction of f to the line $\mathcal{L}$.

▶ **Algorithm 5.4** (PL-VLDT). The *point-line vector-valued low-degree test (PL-VLDT)* is given access to an oracle $f \colon \mathbb{F}^m \to \mathbb{F}^k$ and a "lines" proof oracle g that maps line $\mathcal{L}$ to vector-valued univariate polynomial $g_{\mathcal{L}} : \mathcal{L} \to \mathbb{F}^k$ of degree at most d. It samples a uniformly random line $\mathcal{L}$ through $\mathbb{F}^m$ and uniformly random point $x \in \mathcal{L}$, and accepts if and only if $f(x) = g_{\mathcal{L}}(x)$.

PL-VLDT uses $2m \log |F|$ random coins and makes a single query to each of its two oracles. Its soundness proof is due to Oded Goldreich and Madhu Sudan and can be found in Appendix C. We quote the relevant proposition:

▶ **Proposition 5.5** (Proposition C.4). *Assuming $|\mathbb{F}| > 25k$, if the input oracle $f \colon \mathbb{F}^m \to \mathbb{F}^k$ is δ-far from being a vector-valued polynomial of degree at most d then for any lines oracle g, PL-VLDT rejects f and g with probability at least $\delta/40$.*

We will use a variant of PL-VLDT that has higher query complexity, but does not require an auxiliary "lines" oracle; instead, it queries f on an entire line. As shown in Proposition 5.7, this test has robust soundness, meaning that the expected distance of answers to its queries from agreeing with a univariate low-degree polynomial (and therefore being accepted by the test) is proportional to the distance of the input function from being low-degree.

▶ **Algorithm 5.6** (VLDT). The *vector-valued low-degree test* is given explicit inputs $\mathbb{F}$, m and d and access to an input oracle $f\colon \mathbb{F}^m \to \mathbb{F}^k$. It samples a uniformly random line $\mathcal{L}$ through $\mathbb{F}^m$, queries f on all points in $\mathcal{L}$, and accepts if and only if the obtained values (describing the restriction of f to $\mathcal{L}$) agree with a univariate vector-valued polynomial of degree at most d.

Algorithm 5.6 uses $2m \log |\mathbb{F}|$ random coins and makes $|\mathbb{F}|$ queries to its input. Its robust soundness is asserted in Proposition 5.7.

▶ **Proposition 5.7.** *Assuming $|\mathbb{F}| > 25k$, if the input f is δ-far from being a vector-valued polynomial of degree at most d, then the expected distance of answers to the queries of VLDT from agreeing with a univariate low-degree polynomial (and therefore being accepted by the test) is at least $\beta \cdot \delta$, for $\beta = 1/40$.*

Proof. Given a function f, we define a lines oracle g that assigns each line $\mathcal{L}$ the (univariate, vector-valued) low-degree polynomial $g_{\mathcal{L}}$ closest to $f[\mathcal{L}]$. By definition, when line $\mathcal{L}$ is sampled by VLDT, the distance of answers received from f (namely $f[\mathcal{L}]$) from being accepted is $\delta(f[\mathcal{L}], g_{\mathcal{L}})$. On the other hand, PL-VLDT rejects input oracle f and the lines oracle $g \coloneqq (g_{\mathcal{L}})_{\mathcal{L}}$ with probability $\mathbb{E}_{\mathcal{L}}\left[\mathbb{P}_{x\in\mathcal{L}}\left[f(x) \neq g_{\mathcal{L}}(x)\right]\right] = \mathbb{E}_{\mathcal{L}}\left[\delta(f[\mathcal{L}], g_{\mathcal{L}})\right]$ which, invoking Proposition 5.5, is at least $\delta/40$. Therefore, the *expected* distance of the answers that VLDT receives from being accepted is at least $\delta/40$. ◀

Finally, we recall a basic and important property of low-degree polynomials.

▶ **Fact 5.8** (The Schwartz-Zippel Lemma). *For any finite field $\mathbb{F}$ and integers m and d, if $P\colon \mathbb{F}^m \to \mathbb{F}$ is a nonzero polynomial of degree at most d, then $\mathbb{P}_x\left[P(x) = 0\right] \leq d/|\mathbb{F}|$ for $x \in \mathbb{F}^m$ sampled uniformly at random.*

With these tools in hand, the construction can commence. We analyze the smoothness and strongness of a robust PCPP for Zero on Subcube [20, Section 5.3.2], an algebraic analogue of CircuitVal, starting with a formal definition of this property.

▶ **Definition 5.9** (ZoS). *Fix a finite field $\mathbb{F}$, positive integers m and d and set $H \subseteq \mathbb{F}$. A degree d polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ is* Zero on the Subcube H^m *(*Zero on Subcube *for short) if its restriction to H^m is identically 0. We denote the set of all Zero on the Subcube polynomials by $\text{ZoS}(\mathbb{F}, m, d, H)$. When the parameters are obvious from context, the shorthand* ZoS *is used instead.*

We quote a useful characterization of Zero on Subcube polynomials that gives rise to a natural robust PCPP for the set ZoS.

▶ **Fact 5.10** ([20, Proposition 5.3.4]). *For any field $\mathbb{F}$, positive integers m and d and set $H \subseteq \mathbb{F}$, a polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ of degree at most d is zero on the subcube H^m if and only if there exists a sequence of polynomials $P_1, \ldots, P_m : \mathbb{F}^m \to \mathbb{F}$ each of degree at most d, and a sequence of polynomials $Q_1, \ldots, Q_m : \mathbb{F}^m \to \mathbb{F}$ each of degree at most $d - |H|$, such that for all $x_1, \ldots, x_m \in \mathbb{F}$ and $i \in [m]$:*

$$\begin{aligned} P_{i-1}(x_1, \ldots, x_m) &= \eta(x_i) \cdot Q_i(x_1, \ldots, x_m) + P_i(x_1, \ldots, x_m) \\ P_m(x_1, \ldots, x_m) &= 0 \end{aligned} \tag{8}$$

where $P_0 \coloneqq f$, and η is a univariate polynomial of degree $|H|$ that vanishes on H which we define by $\eta(x) \coloneqq \prod_{h\in H}(x-h)$.

The vector-valued polynomials $P = (P_1, \ldots, P_m)$ and $Q = (Q_1, \ldots, Q_m)$ are called the division witnesses *of f.*[18]

▶ **Algorithm 5.11** (ZoS-RPCPP)**.** The canonical proof-pieces for $f \in \text{ZoS}$ are the division witnesses of f, denoted $P(f) \coloneqq (P_1(f), \ldots, P_m(f))$ and $Q(f) \coloneqq (Q_1(f), \ldots, Q_m(f))$, as guaranteed by Fact 5.10. For input field $\mathbb{F}$, dimension m and degree d, given oracle access to input $f\colon \mathbb{F}^m \to \mathbb{F}$ and proof-pieces $p : \mathbb{F}^m \to \mathbb{F}^m$ and $q : \mathbb{F}^m \to \mathbb{F}^m$, the verifier samples a random line $\mathcal{L}$ through $\mathbb{F}^m$, queries f, p and q on each point in $\mathcal{L}$, and performs the following checks:

1. *Low-degree checks.* Check that the restrictions of f and p to the line $\mathcal{L}$, denoted $f[\mathcal{L}]$ and $p[\mathcal{L}]$, are vector-valued univariate polynomials of degree at most d.[19] Check that $q[\mathcal{L}]$ is a vector-valued univariate polynomial of degree at most $d - |H|$.
2. "Unbundle" the answers received from p and q to obtain $p_i(x)$ and $q_i(x)$ for each $i \in [m]$ and $x \in \mathcal{L}$. Check the following:
   a. *Division checks.* For each $i \in [m]$ and $(x_1, \ldots, x_m) \in \mathcal{L}$, check that
   $$p_{i-1}(x_1, \ldots, x_m) = \eta(x_i) \cdot q_i(x_1, \ldots, x_m) + p_i(x_1, \ldots, x_m)$$
   where $p_0 \coloneqq f$, and η is a univariate polynomial of degree $|H|$ that vanishes on H, defined $\eta(x) \coloneqq \prod_{h\in H}(x-h)$.
   b. *Identity check.* For each $(x_1, \ldots, x_m) \in \mathcal{L}$, check that
   $$p_m(x_1, \ldots, x_m) = 0$$

The verifier accepts if and only if all the above checks passed.

ZoS-RPCPP is piecwise-smooth, makes $|\mathbb{F}|$ queries to each of its three oracles and uses $2m \log |\mathbb{F}|$ random coins. Its residual circuit distance (Definition 3.4) is $1/2$ as any satisfying input to its residual circuit is composed of three univariate polynomials of degree at most d, each occupying a third of the input. As such, any two different satisfying inputs agree on at most a $2/3 \cdot d/|\mathbb{F}| \le 1/2$ fraction of locations.

▶ **Lemma 5.12.** *Suppose* ZoS-RPCPP *is given field $\mathbb{F}$ and degree d such that $1 - \beta \ge 4d/|\mathbb{F}|$, and access to oracles f, p and q. Then the expected distance (over a random line) of the answers to* ZoS-RPCPP*'s queries from being accepted is at least*

$$\frac{\beta}{12} \cdot \min_{f' \in \text{ZoS}(\mathbb{F},m,d,H)} \{\max\left(\delta(f,f'), \delta(p, P(f')), \delta(q, Q(f'))\right)\} \tag{9}$$

where β is the constant of Proposition 5.7, and $P(f')$ and $Q(f')$ are the canonical proof-pieces of f' as described in Algorithm 5.11.

Proof. Fix input oracle $p_0 \coloneqq f\colon \mathbb{F}^m \to \mathbb{F}$ and proof oracles $p = (p_1, \ldots, p_m)$ and $q = (q_1, \ldots, q_m)$ with $p_i, q_i : \mathbb{F}^m \to \mathbb{F}$. Let $\widetilde{f}$ and $\widetilde{p}$ be the vector-valued polynomials of degree at most d closest to f and p, and $\widetilde{q}$ be the polynomial of degree at most $d - |H|$ closest to q. Let $\widetilde{p_i}$ and $\widetilde{q_i}$ be the projections of p and q to their ith coordinate. Note that $p_0 \coloneqq f$ and so $\widetilde{p_0} \coloneqq \widetilde{f}$.

We start by showing that unless $\widetilde{f}$ is Zero on Subcube and $\widetilde{p}, \widetilde{q}$ are its canonical proof-pieces, then the expected distance is $\Omega(1)$.

[18] This ad-hoc term alludes to the proof of Fact 5.10: an iterative process that starts with the polynomial $f = P_0$, and in the ith iteration divides P_{i-1} by $\eta(x_i)$ to obtain quotient Q_i and remainder P_i.
[19] Indeed the term "vector-valued" is degenerate for $f\colon \mathbb{F}^m \to \mathbb{F}$ as its range is one-dimensional.

Case 1: Either f, p or q are $1/4$-far from $\widetilde{f}$, $\widetilde{p}$ or $\widetilde{q}$. Assume wlog that p is $1/4$-far from $\widetilde{p}$. By Proposition 5.7, the expected distance of $p[\mathcal{L}]$ from a univariate polynomial of degree at most d is at least $\beta/4$. However the answers of p are just a third of the answers received by all oracles, so the expected distance of $(f[\mathcal{L}], p[\mathcal{L}], q[\mathcal{L}])$ from satisfying the low-degree checks is at least $\frac{1}{3} \cdot \frac{\beta}{4}$.

Case 2: f, p and q are close to low-degree polynomials that do not satisfy Equation (8); that is, f, p and q are $1/4$-close to $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ but for some i it holds that $\widetilde{p_{i-1}}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ don't satisfy Equation (8). By Fact 5.8, either $\widetilde{p_{i-1}}$, $\widetilde{p_i}$ or $\widetilde{q_i}$ do not satisfy Equation (8) on at least a fraction of $1 - d/|F|$ of points in $\mathbb{F}^m$. Since p_{i-1}, p_i and q_i are $1/4$-close to $\widetilde{p_{i-1}}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ respectively,[20] then p_{i-1}, p_i and q_i do not satisfy Equation (8) on at least a fraction of $1 - d/|\mathbb{F}| - 3/4 = 1/4 - d/|\mathbb{F}|$ points in $\mathbb{F}^m$. Therefore the expected distance of $(f[\mathcal{L}], p[\mathcal{L}], q[\mathcal{L}])$ from satisfying the division checks is at least $1/3 \cdot (1/4 - d/|\mathbb{F}|) \geq \beta/12$.

Case 3: p_m is $1/4$-close to $\widetilde{p_m}$ but $\widetilde{p_m} \not\equiv 0$. By Fact 5.8, $p_m(x) \neq 0$ on at least a fraction of $1/4 - d/|\mathbb{F}|$ of points $x \in \mathbb{F}^m$. As the answers of p are a third of the answers received by all oracles, the expected distance of $(f[\mathcal{L}], p[\mathcal{L}], q[\mathcal{L}])$ from satisfying the identity check is at least $1/3 \cdot (1/4 - d/|\mathbb{F}|) \geq \beta/12$.

If all three previous cases are are false, then the input oracle is close to a Zero on Subcube polynomial, and the given proof-piece oracles are close to the canonical proof-pieces for that polynomial. These proof-pieces are low-degree polynomials, therefore the distance of the given oracles from proving that the input is Zero on Subcube is exactly their distance from being low-degree polynomials (similarly, the distance of the input from being Zero on Subcube is its distance from being low-degree), and VLDT rejects with probability proportional to this distance. Details follow.

As we said, in this case $\widetilde{f}$ is zero on the subcube H^m, and $\widetilde{p}, \widetilde{q}$ are its canonical proof-pieces. Since $\delta(f, \widetilde{f}) < 1/4$ then $\widetilde{f}$ is the only low-degree polynomial $1/4$-close to f, and since $\widetilde{f} \in \text{ZoS}$ it must be that $\widetilde{f}$ is function from $\text{ZoS}(\mathbb{F}, m, d, H)$ closest to f. We conclude the proof by showing that the expected distance of all answers from satisfying is greater than Expression 9: Assume that $\delta(q, \widetilde{q})$ maximizes Expression 9. The expected distance of answers to queries made by VLDT to oracle q from satisfying is at least $\beta \cdot \delta(q, \widetilde{q})$. But as the answers of q account for a third of the all answers received, we have that the expected distance of all answers on a random line from satisfying is at least $\beta \cdot \delta(q, \widetilde{q})/3 = \beta \cdot \delta(q, Q(\widetilde{f}))/3$. If f or p maximized Expression 9 then similar arguing gives the required result. ◀

5.2 A smooth strong canonical RPCP for CircuitSat

To construct a robust PCP for CircuitSat, we first recall an algebrization (i.e. algebraic description) of circuits and their inputs such that an input satisfies the circuit if and only if an encoding of the input is zero on a certain (fixed) subcube.

▶ **Definition 5.13** (Low-degree extension). *Let $z\colon H^m \to \mathbb{F}$. The* low-degree extension of z to $\mathbb{F}^m$ *is the unique polynomial $\widehat{z} : \mathbb{F}^m \to \mathbb{F}$ of degree at most $m|H|$ that agrees with z on H^m.*[21]

[20] Indeed, notice that $\delta(p_i, \widetilde{p_i}) \leq \delta(p, \widetilde{p})$ for all $i \in [m]$ and similarly for q and $\widetilde{q}$.

[21] We will only use the uniqueness and low-degree properties of the extension, but if you insist, know that it is given by

$$\widehat{z}(x) \coloneqq \sum_{h \in H^m} z(h) \cdot \prod_{\substack{i \in [m] \\ h' \in H \setminus \{h_i\}}} \frac{x_i - h'}{h_i - h'}$$

We loosely summarize the algebrization of [20, Section 5.4.1]. A circuit C of size n is associated with a function $C'\colon [n]^3 \times \{0,1\}^3 \to \{0,1\}$ such that $C'(i_1,i_2,i_3,b_1,b_2,b_3) = 1$ if and only if assigning gates i_1, i_2 and i_3 the values $\neg b_1$, $\neg b_2$ and $\neg b_3$ necessarily results in an *invalid* or *unsatisfying* computation of the circuit. The interested reader is referred to [20] for a formal definition of these,[22] and we will describe them by example: if gate 5 is the output gate of C then $C'(5,1,2,1,0,0) = 1$ because (by definition) a satisfying computation does not have the output gate outputting the value 0. As another example, suppose that the 7th gate is an `OR` gate taking input from gates 2 and 4, then $C'(2,4,7,1,1,0) = 1$ because in a valid computation an `OR` gate taking two 0 inputs cannot output 1.

So, C' indicates when an assignment to C's gates results in an invalid or unsatisfying computation. Thus for any alleged computational extension z (see Definition 4.1), the product $F' = C'(i_1,i_2,i_3,b_1,b_2,b_3)\cdot(z[i_1]-b_1)(z[i_2]-b_2)(z[i_3]-b_3)$ will be zero if and only if z is a computational extension of a satisfying input. Another key observation is that replacing C' by its low-degree extension $\widehat{C}$ and z by its low-degree extension $\widehat{z}$ results in a product F that has low-degree. Conversely, F is Zero on Subcube (and in particular low-degree) only if z was a satisfying assignment.

We refer the reader to [20, Section 5.4.1] for a deeper discussion, which is summarized in the following statement.

▶ **Fact 5.14** (Algebrization of CircuitSat)**.** *There exists a polynomial reduction that maps a circuit C of size n to a polynomial $\widehat{C}\colon \mathbb{F}^{3m+3} \to \mathbb{F}$ of degree at most d where $m := \log n/\log\log n$, $H := [n^{1/m}]$, $d = (3m+3)|H|$ and $|\mathbb{F}| = O((d+3m|H|)^3)$, such that $\widehat{C}$ satisfies the following:*

- *For any $e\colon \mathbb{F}^m \to \mathbb{F}$, let $F_{C,e} : \mathbb{F}^{3m+3} \to \mathbb{F}$ be given by*

$$\begin{aligned} F_{C;e}(x_1,\ldots,x_{3m+3}) :=& \widehat{C}(x_1,\ldots,x_{3m+3})\cdot(e(x_1,\ldots,x_m)-x_{3m+1}) \\ &\cdot(e(x_{m+1},\ldots,x_{2m})-x_{3m+2})\cdot(e(x_{2m+1},\ldots,x_{3m})-x_{3m+3}) \end{aligned}$$

  Then, for any polynomial $e\colon \mathbb{F}^m \to \mathbb{F}$ of degree at most $m|H|$, the function $F_{C;e}$ is identically zero on H^{3m+3} if and only if e is the low-degree extension of a computational extension of an assignment that satisfies C.

▶ **Algorithm 5.15** (CircuitSat-RPCP)**.** For any circuit C and satisfying input y the canonical proof consists of four pieces: the low-degree extension of the computational extension of y denoted $\widehat{y_C}$, the polynomial $F_{C;\widehat{y_C}}$ as defined in Fact 5.14, followed by the proof-pieces (for ZoS-RPCPP) that $F_{C;\widehat{y_C}}$ is zero on the subcube H^{3m+3}, namely its division witnesses (of Fact 5.10) denoted $P(C;y)$ and $Q(C;y)$. Given explicit access to circuit C of size n, and oracle access to $e\colon \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, $p : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ and $q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ where $|\mathbb{F}| = O((d+1m|H|)^3)$ and $m' := 3m+3$, the verifier samples a uniformly random line $\mathcal{L}'$ through $\mathbb{F}^{m'}$ and performs the following checks:

1. *Zero on Subcube check.* Check that f is a polynomial of degree at most d' that is zero on the subcube $H^{m'}$ using ZoS-RPCPP with the random line $\mathcal{L}'$, p and q as its proof-piece oracles, and with $d' := m'|H|$ and $H = [n^{1/m}]$.
2. *Low-degree check.* Check that e is a polynomial of degree at most d using VLDT where $d = m|H|$, with the projection of $\mathcal{L}'$ onto its first m coordinates used as the random line. That is, check that $\{e(x_1,\ldots,x_m) : (x_1,\ldots,x_m,\ldots,x_{m'}) \in \mathcal{L}'\}$ agrees with a univariate polynomial of degree at most d.

[22] In [20] these notions are both referred to as *invalid configurations*.

3. *Satisfaction check.* For each $x = (x_1, \ldots, x_{m'}) \in \mathcal{L}'$, check that

$$f(x_1, \ldots, x_{3m+3}) \coloneqq \widehat{C}(x_1, \ldots, x_{3m+3}) \cdot (e(x_1, \ldots, x_m) - x_{3m+1}) \cdot (e(x_{m+1}, \ldots, x_{2m}) - x_{3m+2}) \cdot (e(x_{2m+1}, \ldots, x_{3m}) - x_{3m+3}) \tag{10}$$

where $\widehat{C}$ is as guaranteed by the algebrization of CircuitSat (Fact 5.14).
The verifier accepts if and only if all the above checks passed.

CircuitSat-RPCP tosses $m' \log |\mathbb{F}| = O(m \log |\mathbb{F}|)$ random coins. It issues a total of $6|\mathbb{F}|$ queries to all of its oracles. Namely, for each $x \in \mathcal{L}'$ it issues the following queries:

$$f(x), p(x), q(x), e(x_1, \ldots, x_m), e(x_{m+1}, \ldots, x_{2m}), e(x_{2m+1}, \ldots, x_{3m})$$

It has constant residual circuit distance (Definition 3.4), as any answer string that satisfies its decision circuit consists of 6 univariate low-degree polynomials, each composing a sixth of the entire string. Thus, two different strings satisfying its decision circuit agree on at most a $\frac{5}{6} \cdot \frac{d}{|\mathbb{F}|} \leq 1/2$ fraction of locations.

▶ **Lemma 5.16.** *Suppose* CircuitSat-RPCP *is given satisfiable circuit C and access to proof-piece oracles $e\colon \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, and $p, q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$, with $1 - \beta \geq 5d'/|\mathbb{F}|$ where β is the constant of Proposition 5.7. Then the expected distance of the answers to* CircuitSat-RPCP*'s queries from satisfying is at least*

$$\frac{\beta}{30} \cdot \min_{y' \in \mathtt{Sat}(C)} \left\{ \max \left(\delta\left(e, \widehat{y'_C}\right), \delta\left(f, F_{C;\widehat{y'_C}}\right), \delta(p, P(C; y')), \delta(q, Q(C; y')) \right) \right\} \tag{11}$$

where $\mathtt{Sat}(C)$ denotes the set of satisfying inputs of C, and $\widehat{y'_C}$, $F_{C;\widehat{y'_C}}$, $P(C; y')$ and $Q(C; y')$ are the canonical proof-pieces of y' as described in Algorithm 5.15.

Proof. Fix circuit C and oracles e, f, p and q. Let $\widetilde{e}$ be the closest polynomial to e of degree at most d, and let $\widetilde{f}$, $\widetilde{p}$, $\widetilde{q}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ be the closest low-degree polynomials (as in Lemma 5.12). Consider the following cases:

Case 1: Either of the oracles are 1/5-far from their closest low-degree polynomial. By Proposition 5.7, at least $\beta/5$ of the points read on a random line must be changed in order to satisfy VLDT. The answers to queries made by the low-degree checks (either of Step 2 or Step 1) make up for at least a sixth of the all answers received, therefore the expected distance of all received answers from satisfying is at least $\beta/30$.

Case 2: All oracles are 1/5-close to low-degree polynomials, but $\widetilde{p}, \widetilde{q}$ are not the canonical proof-pieces (for ZoS-RPCPP) that $\widetilde{f}$ is Zero on Subcube. Just like in the proof of Lemma 5.12, the answers to queries made in Step 1 are $\beta/5$-far from satisfying. These answers are a half of answers to all queries, therefore the expected distance of all received answers from satisfying is at least $\beta/10$.

Case 3: All oracles are 1/5-close to low-degree polynomials, but $\widetilde{e}$ and $\widetilde{f}$ do not satisfy Equation (10). Then the expected number of points (on a random line) on which e and f violate Equation (10) is at least $1 - d'/\mathbb{F} - 4 \cdot 1/5 = 1/5 - d'/\mathbb{F}$, which is larger than $\beta/5$. Since the answers to queries made in Step 3 make up for at least two-thirds of answers to all queries, the expected distance of all answers from satisfying is at least $2\beta/15$.

If all of the above are false, then the given oracles are close to proof-pieces proving that C is satisfiable. These proof-pieces are low-degree polynomials, therefore the distance of the given oracles from proving that C is satisfiable is exactly their distance from being low-degree, and VLDT rejects with probability proportional to this distance. Details follow.

In this case, $\widetilde{e}$ is a low-degree extension of a computational extension of a satisfying assignment to C, and $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ are its canonical proof-pieces. Since $\delta(e,\widetilde{e}) < 1/5$ then $\widetilde{e}$ is the only low-degree polynomial $1/5$-close to e, so it must be that $\widetilde{e} = \widehat{y'_C}$ (where y' is the minimizer of Expression 11). Assume that q maximizes Expression 11. The expected distance of answers to queries made by VLDT to q from satisfying is at least $\beta \cdot \delta(q,\widetilde{q})$. But as the answers of q account for a sixth of all answers received, the expected distance of all answers from satisfying is at least $\beta\delta(q,\widetilde{q})/6$. If the maximizers were e, f or p the claim follows using similar reasoning. ◀

5.3 A smooth strong canonical RPCPP for CircuitVal

In this section we transform CircuitSat-RPCP to an RPCP *of proximity* for CircuitVal. The RPCPP is constructed by (additionally) testing consistency of the input oracle (an allegedly satisfying input to the circuit) with the proof-piece oracle corresponding to the alleged low-degree extension of its computational extension (see Definition 4.1). Low degree extension is systematic, in the sense that the extended function is embedded in its extension, so consistency may be checked by testing that the input oracle agrees with the systematic part of its alleged low-degree extension on a uniformly random location. However, the systematic part of the proof-piece oracle is only a tiny fraction of it (as $n = o(|\mathbb{F}^m|)$), so a particularly nasty proof-piece oracle could be close to an arbitrary low-degree polynomial (therefore passing the low-degree check), whose systematic part was changed to match the input oracle (therefore passing the consistency check) – causing the verifier to accept a proof that's extremely far from the canonical one. This is resolved via self-correction: the verifier checks that the proof-piece oracle is a low-degree univariate polynomial on a random (punctured, see below) line through the location queried in the systematic part.

Unlike previous steps in this construction, some care must be taken so that the consistency check does not harm smoothness. If the verifier reads the entire line, locations in the systematic part would be queried more often than others. Indeed, the verifier does not need to read the systematic point on the line and can interpolate it from the others. But if the verifier *always* avoids reading the systematic part, its locations would be queried less often than others. Thus, the verifier chooses the puncture (i.e. point on the line not to be read) to be the location in the systematic part with probability $1 - \Theta(1/|F|^{m-1})$, and to be a different point on the line with the remaining probability.

▶ **Algorithm 5.17** (CircuitVal-RPCPP)**.** The canonical proof of a circuit and its satisfying input consists of the same four proof-pieces as in Algorithm 5.15. Given explicit access to circuit C of size n with n_0 input gates, and oracle access to input $y\colon [n_0] \to \{0,1\}$ and proof-pieces $e : \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, $p : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ and $q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ where $m' \coloneqq 3m+3$, the verifier performs the following checks:

1. *Satisfiability check.* Check that C is satisfiable using CircuitSat-RPCP with e, f, p and q as proof-piece oracles.
2. *Consistency check.* Sample a location in the input oracle by uniformly sampling $x \in [n_0] \equiv H^m$. Sample a random line $\mathcal{L}$ with intercept x by sampling $h \in \mathbb{F}^m$ and letting $\mathcal{L} \coloneqq (x + ih : i \in \mathbb{F})$. Let the puncture z be $z \coloneqq x$ with probability $1 - (|\mathbb{F}| - 1)/(|\mathbb{F}|^m + 1)$, and $z \coloneqq x + h$ with the remaining probability.
   Query e on the line $\mathcal{L}$ punctured at z, i.e. $e[\mathcal{L} \setminus \{z\}]$. Query $y(x)$, giving the query a weight of $|\mathcal{L}| = |\mathbb{F}|$.[23]. Do the following:

[23] Reweighting is achieved by repeating the input gate corresponding to $y(x)$ in the verifier's residual circuit.

a. Check that $e[\mathcal{L} \setminus \{x\}]$ agrees with a univariate polynomial of degree at most d. If the check passed, let $e_{\mathcal{L}} : \mathcal{L} \to \mathbb{F}$ be the unique low-degree polynomial that agrees with $e[\mathcal{L} \setminus \{z\}]$.
b. Check that $y(x) = e_{\mathcal{L}}(x)$: If $z = x$, obtain $e_{\mathcal{L}}(x)$ using interpolation (*not* by querying $e(x)$). If $z = x + h$ then $e_{\mathcal{L}}(x) = e(x)$ has already been queried and is known to the verifier.

The verifier accepts if and only if the above checks passed.

CircuitVal-RPCPP tosses $(m + m') \log |\mathbb{F}| = O(m \log |\mathbb{F}|)$ random coins. It issues a total of $8|\mathbb{F}| - 1$ (weighted) queries to its oracles: in addition to the $6|\mathbb{F}|$ queries of CircuitSat-RPCP, it queries e on $\mathcal{L} \setminus \{z\}$ in Step 2a, and issues a query of weight $|\mathbb{F}|$ to w. It has constant residual circuit distance (Definition 3.4), which is shown by an application of Fact 5.8, just as with ZoS-RPCPP and CircuitSat-RPCP.

Smoothness of CircuitVal-RPCPP

We explain how the randomized choice of puncture in Step 2 preserves smoothness. Let α denote the probability that the puncture z is at x (i.e., verifier reads $e[\mathcal{L} \setminus \{x\}]$), and $1 - \alpha$ be the probability that the puncture is at $x + h$. We show that each point in $\mathbb{F}^m$ is queried with equal probability.

- *The non-systematic part:* A point from the non-systematic part, i.e. $\mathbb{F}^m \setminus [n_0]$, is read only if it lies on the line $\mathcal{L}$ and does not equal the puncture. This occurs with probability

$$\frac{|\mathbb{F}| - 2}{|\mathbb{F}|^m} + \frac{1}{|\mathbb{F}|^m} \cdot \alpha \tag{12}$$

- *The systematic part:* Consider a point from the systematic part, i.e. $[n_0]$. If it is chosen to be the intercept x of line $\mathcal{L}$, then it is read only if the puncture is not at x but at $x + h$ – which occurs with probability $1 - \alpha$. If it is not chosen to be the intercept x, then it is queried with probability equal to that of points in the non-systematic part, as analyzed above. Thus, points in the systematic part are queried with probability

$$\frac{1}{n_0} \cdot (1 - \alpha) + \left(1 - \frac{1}{n_0}\right) \cdot \left(\frac{|\mathbb{F}| - 2}{|\mathbb{F}|^m} + \frac{1}{|\mathbb{F}|^m} \cdot \alpha\right) \tag{13}$$

Indeed, choosing $\alpha = 1 - (|F| - 1)/(|\mathbb{F}|^m + 1)$ equalizes Expressions 12 and 13.

Soundness of CircuitVal-RPCPP

Robust strong canonical soundness of CircuitVal-RPCPP follows from the following lemma.

▶ **Lemma 5.18.** *Suppose* CircuitVal-RPCPP *is given a satisfiable circuit C, input oracle $y \colon [n_0] \to \{0,1\}$ and proof-piece oracles $e : \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, and $p, q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$, with $1 - \beta \geq d/|\mathbb{F}|$. Then the expected distance of the answers to* CircuitVal-RPCPP*'s queries from satisfying is at least*

$$\frac{\beta}{64} \cdot \min_{y' \in \mathtt{Sat}(C)} \left\{ \max \left(\delta(y, y'), \delta\left(e, \widehat{y'_C}\right), \delta\left(f, F_{C;\widehat{y'}}\right), \delta(p, P(C; y')), \delta(q, Q(C; y')) \right) \right\} \tag{14}$$

where $\mathtt{Sat}(C)$ *denotes the set of satisfying inputs of C, β is the constant of Proposition 5.7, and $\widehat{y'_C}$, $F_{C;\widehat{y'_C}}$, $P(C; y')$ and $Q(C; y')$ are the canonical proof-pieces of y' as described in Algorithm 5.15.*

Proof. Fix input C and oracles y, e, f, p and q. Let $\widetilde{e}$, $\widetilde{f}$, $\widetilde{p}$, $\widetilde{q}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ as in the proof of Lemma 5.16, and let y' be the minimizer of Expression 14.

Following the analysis of CircuitSat-RPCP, if $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ are not the canonical proof-pieces of $\widetilde{e}$ proving that it is the low-degree extension of a satisfying assignment to C, then the expected distance of answers to queries issued to e ,f, p and q from satisfying is at least $\beta/30$. Since these answers account for more than $6/8$ of all answers, then the expected distance of answers to queries issued to all oracles is at least $6/8 \cdot \beta/30$.

Otherwise $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ are the canonical proofs proving that $\widetilde{e}$ encodes a satisfying assignment denoted $\widetilde{y}$. Again using the analysis of CircuitSat-RPCP we have that the expected distance of answers to queries issued to all oracles is at least

$$\frac{6}{8} \cdot \frac{\beta}{30} \cdot \max\left(\delta\left(e, \widehat{y'_C}\right), \delta\left(f, F_{C;\widehat{y'_C}}\right), \delta(p, P(C;y')), \delta(q, Q(C;y'))\right)$$

We now argue that the expected distance is proportional also to $\delta(y, y')$. With probability $\Omega(\delta(y, y'))$, either the query from y must be changed (and it has a constant fraction of the weight of all queries), or e must be changed on a constant fraction of locations on the line $\mathcal{L}$ (due to self-correction). Details follow.

Notice that if e is $1/4$-far from $\widetilde{e}$ then by Proposition 5.7 and a similar weighting argument, the distance of all answers from satisfying is at least $1/4 \cdot \beta \cdot (1/8 - 1/|\mathbb{F}|)$,[24] so what's left is to show that the expected distance is proportional to $\delta(y, \widetilde{y})$. Indeed, if the sampled location x in Step 2 is one on which y and $\widetilde{y}$ differ, i.e. such that $y(x) \neq \widetilde{e}(x)$, then either the value of $y(x)$ must be changed or $e[\mathcal{L} \setminus \{z\}]$ must be changed to agree with a univariate degree d polynomial whose value at x is $y(x)$. Even with x fixed, all points in $\mathcal{L} \setminus \{x\}$ are marginally uniform, so by linearity of expectation,

$$\mathop{\mathbb{E}}_{\text{Random } \mathcal{L} \text{ with intercept } x} [\delta(e[\mathcal{L} \setminus \{x\}], \widetilde{e}[\mathcal{L} \setminus \{x\}])] = \delta(e, \widetilde{e}) \leq 1/4$$

By Markov's inequality, for any $x \in \mathbb{F}^m$ and random line $\mathcal{L}$ with intercept x, with probability at least $1/2$ it holds that $e[\mathcal{L} \setminus \{x\}]$ is $1/2$-close to $\widetilde{e}[\mathcal{L} \setminus \{x\}]$ and therefore $e[\mathcal{L} \setminus \{z\}]$ is $2/3$-close to $\widetilde{e}[\mathcal{L} \setminus \{z\}]$ (this accounts for the case that $z = x + h$ in Step 2. Conditioned on this event, $e[\mathcal{L} \setminus \{z\}]$ must be changed on at least $1 - d/|\mathbb{F}| - 2/3 \geq 1/4$ fraction of locations to make the consistency check of Step 2 pass.

All in all, we have that with probability at least $\delta(y, \widetilde{y})/2$, at least a quarter of the answers received either from y or from e (in Step 2) must be changed so that Step 2 does not reject. As these account for at least a $(1/8 - 1/|\mathbb{F}|)$ of all answers received, we have that the expected distance is at least $\delta(y, \widetilde{y})/64$. ◀

5.4 Alphabet Reduction

So we have an RPCPP for CircuitVal, but we aren't done just yet: CircuitVal-RPCPP is *non-Boolean*, meaning that it's proofs are written using non-binary symbols. It's time to show how to transform it to a Boolean RPCPP. Actually, we will show how to transform *any* non-Boolean RPCPP to a Boolean one of comparable complexities – provided its symbols are not much larger than the number of queries it issues. Most importantly, we prove that this transformation preserves smoothness and strong canonicality.

We show this even for multi-piece RPCPPs that have different-sized alphabets for different pieces. That is, letting the ith proof-piece oracle answers its queries with symbols in the alphabet $\Sigma_i = \{0,1\}^{\sigma_i}$ it could be that not all σ_i's are equal. We call σ_i the *ith proof-piece answer-length complexity* of the RPCPP.

[24] Due to Step 2, but in fact low-degree tests occur in Step 1 as well.

▶ **Lemma 5.19.** *Suppose* CircuitVal *has a smooth and strong canonical multi-piece RPCPP with constant residual circuit distance, ith answer complexity σ_i and query complexity q, and the additional property that the verifier makes the same amount of queries to each of its oracles up to a constant multiplicative factor ρ.*[25] *Then,* CircuitVal *has a Boolean smooth and strong canonical multi-piece RPCPP with query complexity $O(q \cdot \max_{i\in[k]} \sigma_i)$, decision complexity growing by an additive factor of $\widetilde{O}(q \cdot \max_{i\in[k]} \sigma_i)$, and strongness parameter shrinking by a constant factor.*

Following [20, Section 5.4.3], the Boolean PCP is obtained by replacing each letter of the non-Boolean PCP with its encoding in an error correcting code of constant relative distance. Since each proof-piece may have different answer complexity, we choose a different code for each proof-piece. Starting with a good code (i.e., of constant relative distance and rate) for the proof-piece with maximal answer-length complexity, we take the other codes to have equal distance and *block length*, increasing the rate if necessary using code repetition. Equal block length is necessary so that the generalized Hamming distance between two non-Boolean proofs is reflected accurately by their corresponding Boolean proofs (and is not scaled with differences in answer complexities as in Remark 5.2).

The Boolean verifier emulates the non-Boolean one: for each symbol the non-Boolean verifier queries, the Boolean verifier queries all bits in the corresponding block and answers the emulated verifier with their decoding (rejecting if the block was not decodable). We then prove that any Boolean proof is either far from being composed of codewords (therefore far from being accepted by the Boolean verifier in expectation), or its expected distance from satisfying the Boolean verifier is similar to the expected distance of its *decoding* from satisfying the non-Boolean verifier, where its *decoding* refers to the non-Boolean string obtained by decoding each of its blocks. Since the distance of the Boolean proof from the canonical proof (for the Boolean verifier) is proportional to the distance of the decoded proof from the canonical proof (for the non-Boolean verifier), strong canonicality of the non-Boolean verifier implies strong canonicality of the Boolean one. A formal proof follows.

Proof of Lemma 5.19. Let $\widehat{V}$ be the smooth strong canonical RPCPP for CircuitVal with corresponding proof-piece strategies $\Pi_1, \ldots, \Pi_t$, with Π_i answering its queries with symbols in the alphabet $\Sigma_i = \{0,1\}^{\sigma_i}$. For each $i \in [t]$ let ℓ_i denote the length complexity of the ith proof-piece oracle, and let $\mathrm{ECC}_i \colon \Sigma_i \to \{0,1\}^b$ with $b = O(\max_{i\in[t]} \sigma_i)$ be an error correcting code of constant minimal distance $2c > 0$, decodable using circuits of size $\widetilde{O}(b)$. The ith canonical proof-piece of circuit C and input y is $\Gamma_i(C;y)$, where $\Gamma_i(C;y)[j] \coloneqq \mathrm{ECC}_i(\Pi_i(C;y))$ is a binary string of length b.[26]

The Boolean RPCPP verifier, denoted V, expects circuit C, input oracle y and proof-piece oracles $\gamma_1, \ldots, \gamma_t$. It emulates $\widehat{V}$ on circuit C and input oracle y as follows:

- When $\widehat{V}$ queries the jth location of the ith proof-piece oracle, issue b queries to γ_i to receive a binary string $\gamma_i[j]$ of length b. Decode $\gamma_i[j]$ using the decoding algorithm of ECC_i. If decoding failed then reject, and otherwise answer $\widehat{V}$'s query with the decoded string.
- When $\widehat{V}$ queries its input oracle, answer it according to the input oracle y, giving the query weight b.

[25] That is, for any two oracles, the ratio between the number of queries that the verifier makes to each oracle is at least ρ.

[26] To be clear, $\Pi_i(C;y)$ is a string of length ℓ_i over alphabet $\Sigma_i = \{0,1\}^{\sigma_i}$, and $\Gamma_i(C;y)$ is a string of length $\ell_i \cdot b$ over the binary alphabet.

Smoothness of V follows immediately from the smoothness of $\widehat{V}$: for each proof-piece, a bit is queried if and only if the emulated verifier queried the location which it allegedly encodes, and the the probability that the latter is queried is the same for all locations.

We show that V is a strong canonical RPCPP for CircuitVal. Fix circuit C, input oracle y and alleged proof-piece oracles $\gamma_1, \ldots, \gamma_t$. For each $i \in [t]$ and $j \in [\ell_i]$, let $\widehat{\pi}_i[j] \in \Sigma_i$ be the "best decoding" of $\gamma_i[j]$, namely, the string $x \in \Sigma_i$ that minimizes the expression $\delta(\mathrm{ECC}_i(x), \gamma_i[j])$. Let $\widehat{\gamma}_i[j]$ be the encoding of $\widehat{\pi}_i[j]$, that is $\widehat{\gamma}_i[j] \coloneqq \mathrm{ECC}_i(\widehat{\pi}_i[j])$.

▷ Claim 5.19.1. For all $i \in [t]$, the expected distance of answers to V's queries from being accepted is greater than

$$\frac{\rho}{t+1} \cdot \delta(\gamma_i, \widehat{\gamma}_i)$$

Proof of Claim 5.19.1. Fix $i \in [t]$ and denote by J the locations that V queries in γ_i. To satisfy V, the answers of γ_i to V's queries (denoted $\gamma_i[J]$) must be codewords, therefore the expected distance of $\gamma_i[J]$ from satisfying V (over J generated according to V's random coins) is at least the expected distance of $\gamma_i[J]$ from being codewords; that is, at least

$$\mathop{\mathbb{E}}_{J}\left[\delta(\gamma_i[J], \widehat{\gamma}_i[J])\right] = \mathop{\mathbb{E}}_{J,\, j \in J}\left[\delta(\gamma_i[j], \widehat{\gamma}_i[j])\right] \tag{15}$$

Smoothness of V implies that a uniformly random element of $j \in J$ is distributed uniformly at random in $[\ell_i]$ (as detailed in Appendix B.2). Thus,

$$\textit{Equation (15)} = \mathop{\mathbb{E}}_{k \in [\ell_i]}\left[\gamma_i[k], \widehat{\gamma}_i[k]\right] = \delta(\gamma_i, \widehat{\gamma}_i)$$

The distance shrinks by $\rho/(t+1)$ because $\gamma_i[J]$ make up for only (at least) a $\rho/(t+1)$ fraction of all bits read by V. ◁

▷ Claim 5.19.2. Fix a random coin sequence and suppose that the Boolean verifier V received answers $s = s_1 \cdots s_q$ from the oracles $y, \gamma_1, \ldots, \gamma_t$. Let $\widehat{s} = \widehat{s}_1 \cdots \widehat{s}_1$ denote the answers that the non-Boolean verifier $\widehat{V}$ received from oracles $y, \widehat{\pi}_1, \ldots, \widehat{\pi}_t$ when it samples the same coin sequence; in other words, letting $s_k = \gamma_i[j]$ for some i and j, we denote $\widehat{s}_k = \widehat{\pi}_i[j]$.

Let D and $\widehat{D}$ be the residual circuits generated by V and $\widehat{V}$ upon sampling the fixed random coin sequence. It holds that $\delta(s, \mathtt{Sat}(D)) \geq c \cdot \delta(\widehat{s}, \mathtt{Sat}(\widehat{D}))$, where $\mathtt{Sat}(\cdot)$ denotes the set of satisfying inputs of a circuit, and $2c$ is the distance of the error correcting codes ECC_i.

Proof of Claim 5.19.2. The proof follows from the Markov bound and a triangle inequality. Suppose there is $s' = s'_1 \cdots s'_q$ that satisfies D such that $\delta(s, s') = \Delta$. For at least a $1 - \Delta/c$ fraction of $k \in [q]$ it holds that s_k is c-close to s'_k. For these k's, s'_k is the closest codeword to s_k, because distinct codewords are $2c$-far apart, and s'_k is a codeword for it satisfies D. Therefore, for a $1 - \Delta/c$ fraction of k's it holds that s'_k encodes $\widehat{s}_k$, and it follows that $\widehat{s}$ is Δ/c-close to satisfying $\widehat{V}$. ◁

Concluding the proof of Lemma 5.19

Fix $y' \in \mathtt{Sat}(C)$ such that $\max_{i \in [t]}\{\delta(y, y'), \delta(\widehat{\pi}_i, \Pi_i(C; y'))\}$ is minimal, and let $\Pi'_i \coloneqq \Pi_i(C; y')$ and $\Gamma'_i \coloneqq \Gamma_i(C; y')$. Suppose we give $\widehat{V}$ oracle access to $y, \widehat{\pi}_1, \ldots, \widehat{\pi}_t$. Robust strong canonicality means that the expected distance of locations it reads from satisfying its residual circuit is at least

$$\alpha \cdot \max_{i \in [t]}\left(\delta(y, y'), \delta(\widehat{\pi}_i, \Pi'_i)\right)$$

By Claim 5.19.2, this implies that when the Boolean verifier V is given $y, \gamma_1, \ldots, \gamma_t$, the expected distance of locations it reads from satisfying its residual circuit is at least

$$\alpha \cdot c \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\widehat{\pi}_i, \Pi'_i)\right)$$

Noticing that $\delta(\widehat{\pi}_i, \Pi'_i) = 2c \cdot \delta(\widehat{\gamma}_i, \Gamma'_i)$, this is at least

$$\alpha c \cdot 2c \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\widehat{\gamma}_i, \Gamma'_i)\right) \tag{16}$$

Claim 5.19.1 means that the expected distance of answers to V from satisfying is lower bounded not merely by Expression 16, but by

$$2\alpha c^2 \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\widehat{\gamma}_i, \Gamma'_i), \frac{\rho}{t+1} \cdot \delta(\gamma_i, \widehat{\gamma}_i)\right) \tag{17}$$

For each $i \in [t]$, if $\delta(\gamma_i, \widehat{\gamma}_i) \leq \delta(\widehat{\gamma}_i, \Gamma'_i)/2$ then by the triangle inequality $\delta(\widehat{\gamma}_i, \Gamma'_i) \geq \delta(\gamma_i, \Gamma'_i)/2$, and otherwise $\delta(\gamma_i, \widehat{\gamma}_i) > \delta(\widehat{\gamma}_i, \Gamma'_i)/2$, so Expression 17 is at least

$$2\alpha c^2 \cdot \frac{\rho}{2(t+1)} \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\gamma_i, \Gamma'_i)\right) \geq \frac{\alpha \rho c^2}{(t+1)} \cdot \min_{y'' \in \mathtt{Sat}(C)} \left\{ \max_{i \in [t]} \left(\delta(y, y''), \delta(\gamma_i, \Gamma_i(x; y''))\right) \right\}$$

where the inequality is because y' is a satisfying input for C, but not necessarily a minimizer of the above quantity. Thus, V is robust strongly canonical.

If $\widehat{V}$ has residual circuit distance c' (Definition 3.4), then V has residual circuit distance $c \cdot c'$, because an answer string that satisfies the decision circuit of V must be formed of encodings of an answer string that satisfies $\widehat{V}$. ◀

5.5 Putting it all together

Because CircuitVal-RPCPP is the Reed–Muller-based RPCPP of [20, Section 5.4], it enjoys the (standard) soundness analysis presented in that work. Paired with the strong canonical soundness of Lemma 5.18, we have a non-Boolean smooth strong canonical RPCPP for CircuitVal of logarithmic randomness complexity, polylogarithmic query complexity and constant residual circuit distance. Decision complexity is also polylogarithmic (though we did not keep track of it explicitly). Using Lemma 5.19 we get a Boolean RPCPP for CircuitVal of similar properties, and a final application of Lemma 2.2 reduces the number of oracles to one – proving Proposition 3.7.

References

1 Josh Alman and Lijie Chen. Efficient Construction of Rigid Matrices Using an NP Oracle. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, MD, USA, November 9-12, 2019*, pages 1034–1055. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00067.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

3 Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

4 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability Yields a PTAS for k-Median and k-Means Clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 309–318. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.36.

5 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 754–767. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.

6 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM J. Comput.*, 36(4):889–974, 2006. doi:10.1137/S0097539705446810.

7 Yonatan Bilu and Nathan Linial. Lifts, Discrepancy and Nearly Optimal Spectral Gap*. *Combinatorica*, 26(5):495–519, 2006. doi:10.1007/s00493-006-0029-7.

8 Yonatan Bilu and Nathan Linial. Are Stable Instances Easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012. doi:10.1017/S0963548312000193.

9 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.

10 Irit Dinur, Oded Goldreich, and Tom Gur. Every Set in P Is Strongly Testable Under a Suitable Encoding. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.30.

11 Irit Dinur and Omer Reingold. Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006. doi:10.1137/S0097539705446962.

12 Zachary Friggstad, Kamyar Khodamoradi, and Mohammad R. Salavatipour. Exact Algorithms and Lower Bounds for Stable Instances of Euclidean k-MEANS. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2958–2972. SIAM, 2019. doi:10.1137/1.9781611975482.183.

13 Anna Gál and Andrew Mills. Three-Query Locally Decodable Codes with Higher Correctness Require Exponential Length. *TOCT*, 3(2):5:1–5:34, 2012. doi:10.1145/2077336.2077338.

14 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

15 Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong Locally Testable Codes with Relaxed Local Decoders. *TOCT*, 11(3):17:1–17:38, 2019. doi:10.1145/3319907.

16 Oded Goldreich, Howard J. Karloff, Leonard J. Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. *Computational Complexity*, 15(3):263–296, 2006. doi:10.1007/s00037-006-0216-3.

17 Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006. doi:10.1145/1162349.1162351.

18 Tom Gur, Govind Ramnarayan, and Ron D. Rothblum. Relaxed Locally Correctable Codes. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 27:1–27:11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.27.

19 Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Computational Complexity*, 27(1):99–207, 2018. doi:10.1007/s00037-016-0136-9.

20 Prahladh Harsha. *Robust PCPs of Proximity and Shorter PCPs*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2004.

21 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 80–86. ACM, 2000. doi:10.1145/335305.335315.

22 Subhash Khot. Hardness Results for Coloring 3 -Colorable 3 -Uniform Hypergraphs. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 23–32. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181879.

23 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.

24 Sergey Yekhanin. Locally Decodable Codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012. doi:10.1561/0400000030.

A Hardness of approximating bounded-occurrence stable3SAT

To see the connection between our PCP result (i.e. Theorem 1.5) and the hardness of approximating bounded-occurrence stable3SAT (i.e. Corollary 1.7), first recall the connection between standard PCPs and constraint satisfaction problems (CSPs).[27] At its heart stands a correspondence between the probability that a nonadaptive PCP verifier V for a set S accepts an input x and proof π, and the fraction of simultaneously satisfiable constraints in a CSP $\Phi_{V,x}$ over $|\pi|$ variables. Consider the CSP $\Phi_{V,x}$ of 2^r constraints that express the verifier's decision after tossing r random coins and querying its proof oracle in q locations. These are constraints over $|\pi|$ variables, with each constraint depending on the q variables corresponding to locations that the verifier queries (i.e., $\Phi_{V,x}$ is a qCSP). A proof π is an assignment to these variables; completeness means that if $x \in S$ then there is an assignment that satisfies all constraints in $\Phi_{V,x}$, and soundness means that if $x \notin S$ then no assignment satisfies more than half of the constraints in $\Phi_{V,x}$ simultaneously.

Now, what can be said about the CSP $\Phi_{V,x}$ if the PCP verifier V was also strong and smooth?

- A strong PCP verifier with strongness parameter α yields a *stable* CSP $\Phi_{V,x}$, in the sense that an assignment that is δ-far from all satisfying assignments violates an $\alpha \cdot \delta$ fraction of constraints. Like strong soundness, this property holds both when $\Phi_{V,x}$ is satisfiable (i.e. when inputs $x \in S$) and when $\Phi_{V,x}$ is unsatisfiable (i.e. when $x \notin S$). We note that in the latter case, $\Phi_{V,x}$ cannot have more than an α fraction of constraints satisfied simultaneously.
- A smooth PCP verifier yields a CSP such that each variable occurs in the same number of constraints.

Thus, Theorem 1.5 implies that for any set $S \in \mathcal{NP}$ there is an efficient parsimonious reduction from S to α-stable bounded-arity qCSPs with equal variable occurrence. To prove Corollary 1.7, we show a polynomial-time computable parsimonious reduction of α-stable qCSPs with *equal variable occurrence* to $\Omega(\alpha)$-stable qCNFs with *bounded variable occurrence* (Proposition A.1), and then reduce the latter to $\Omega(\alpha)$-stable 3CNFs with bounded variable occurrence (Proposition A.2).

Recall that a formula has *b-bounded-occurrence* if any variable appears in at most b clauses, and that the promise problem (α, b)-stable$q$SAT is distinguishing b-bounded-occurrence qCNF formulas that are α-stable and satisfiable from ones in which any assignment violates at least an α fraction of the clauses.

[27] A CSP Φ of size m over ℓ variables is a set of m functions (called *constraints*) $\Phi = \{C_1, \ldots, C_m\}$ with $C_i : \{0,1\}^\ell \to \{0,1\}$. An assignment $A : [\ell] \to \{0,1\}$ *satisfies* constraints C_i if $C_i(A(1), \ldots, A(\ell)) = 1$. We say that Φ has *arity* q (is a qCSP) if each C_i depends on at most q variables. If constraint C_i depends on variable v then we say that v *occurs* in C_i.

▶ **Proposition A.1.** *For any $q \in \mathbb{N}$ there is $b \in \mathbb{N}$ such that there is a polynomial-time computable parsimonious reduction from α-stable qCSPs with equal variable occurrence to $(\alpha/qb, b)$-stable$q$SAT.*

Proof. We will show a reduction to α/qb'-stable qCSPs with b'-bounded variable occurrence for some constant $b' \in \mathbb{N}$ (independent of q, actually). Then, q-arity of the resulting CSP implies that each of its constraints can be expressed as the conjunction of at most 2^q disjunctive clauses, and conjuncting all these clauses gives a CNF formula that is $\alpha/qb'2^q$-stable and $b'2^q$-bounded variable occurrence. Setting $b \coloneqq b'2^q$ gives the required result.

The reduction, taken from [23], replaces the occurrence of each variable in a constraint with a copy of that variable, and adds consistency (i.e. equality) constraints between copies based on an expander graph of constant degree. We stress that the stability of the obtained CNF crucially relies on the equal variable occurrence of the reduced CSP so that each variable is replaced with the same number of consistency checks, and that the new instance is not necessarily stable without this property (see [12, Appendix D]).

Fix an α-stable CSP Φ consisting of m constraints over ℓ variables, such that each constraint depends on exactly q variables and each variable occurs in exactly mq/ℓ constraints.

Fix a d-regular, explicit[28] graph G over mq/ℓ vertices, with an expansion property guaranteeing that for any set of vertices T that consists of at most a half of the vertices, there are $2|T|$ edges crossing from T to its complement (for example, the expanders of [7]). The variables of Φ' are $\{v_C\}$ for each variable v of Φ and each constraint $C \in \Phi$ in which v occurs. There are two types of constraints in Φ':

- For each constraint $C \in \Phi$, a *primal constraint* C' is added, which is simply C but with variables replaced by their corresponding (alleged) copies. For example, if $C = (x \vee \overline{y} \vee z \vee w) \wedge (\overline{w} \vee x \vee u)$, then $C' = (x_C \vee \overline{y_C} \vee z_C \vee w_C) \wedge (\overline{w_C} \vee x_C \vee u_C)$.
- For each variable v of Φ, $\frac{mq}{\ell} \cdot \frac{d}{2}$ *consistency constraints* are added to Φ': letting $\{C_1, \ldots, C_{mq/\ell}\}$ be the set of constraints in which v occurs, for each edge $\{i, j\}$ in G we add a constraint that is satisfied if and only if v_{C_i} equals v_{C_j}.

All in all, Φ' has $\ell \cdot mq/\ell = mq$ variables and $m + \ell \cdot \frac{mq}{\ell} \cdot \frac{d}{2} = (1 + qd/2)m$ constraints.

Each variable of Φ' occurs in $d+1$ constraints, and the reduction is indeed parsimonious since any assignment that satisfies Φ' must be consistent and therefore gives a satisfying assignment to Φ (and vice-versa). We show that Φ' is $\frac{\alpha}{qd}$-stable. The case that Φ is unsatisfiable follows from [23], so we focus on the case that Φ is satisfiable (and then so is Φ'). Let A' be an assignment to Φ' that violates a γ' fraction of its constraints. Let A_{Maj} be an assignment to Φ that gives each variable a value according to the majority value that A' gives to its (alleged) copies; that is, for each variable v, $A_{\mathrm{Maj}}(v) \coloneqq \mathrm{Maj}_C(A'(v_C))$. Let A'_{Maj} be the extension of A_{Maj} back to the variables of Φ'; that is, $A'_{\mathrm{Maj}}(v_C) \coloneqq A_{\mathrm{Maj}}(v)$ for each variable v and constraint C that depends on v. Let γ_{Maj} denote the fraction of constraints in Φ unsatisfied by A_{Maj}.

▷ Claim A.1.1.

$$\gamma' \geq \frac{\gamma_{\mathrm{Maj}} + \delta(A', A'_{\mathrm{Maj}})}{qd}$$

[28] I.e. there is an algorithm that constructs G in polynomial time when given mq/ℓ as input.

Proof of Claim A.1.1. To prove the claim, we show that whatever primal constraints may be satisfied by A' but not by A'_{Maj} (which works against the lower bound we need) are "made up for" in consistency clauses unsatisfied by A'. It will be nicer to deal with whole numbers rather than fractions, so we let a' (resp. a_{Maj}) denote the *number* of constraints of Φ' (resp. Φ) unsatisfied by A' (resp. A_{Maj}), and notice that the claim follows from showing that

$$a' \geq a_{\text{Maj}} + \left|\left\{v_C : A'(v_C) \neq A'_{\text{Maj}}(v_C)\right\}\right|$$

Fix a variable v, and let k be the number of constraints C such that A' disagrees with A'_{Maj} on the value assigned to v_C. Then v occurs in at most k constraints of Φ that are unsatisfied by A_{Maj} but whose corresponding primal constraint is satisfied by A'. But by the expansion property of the graph G, each of the k inconsistencies result in $2k$ consistency constraints unsatisfied by A'. ◁

Now, let B be the closest satisfying assignment to A_{Maj}. Note that the extension of B to the variables of Φ', denoted B', is a satisfying assignment to Φ', and that $\delta(A_{\text{Maj}}, B) = \delta(A'_{\text{Maj}}, B')$. Using Claim A.1.1 and stability of Φ, we have

$$\gamma' \geq \frac{\gamma_{\text{Maj}} + \delta(A', A'_{\text{Maj}})}{qd} \geq \frac{\alpha\delta(A_{\text{Maj}}, B) + \delta(A'_{\text{Maj}}, B')}{qd} \geq \frac{\alpha}{qd} \cdot \delta(A', B') \qquad \blacktriangleleft$$

▶ **Proposition A.2.** *For any $\alpha \in (0, 1]$ and $b, q \in \mathbb{N}$ there is $\alpha' \in (0, 1]$ such that there is a polynomial-time parsimonious reduction from (α, b)-`stable`q`SAT` to (α', b)-`stable3SAT`.*

Friggstad et al. show a similar reduction in [12][Section 4]. Furthermore, their reduction yields CNFs that have *exactly* three literals per clause.

Proof. We use the standard parsimonious reduction of qSAT to 3SAT to transform a formula φ to a formula φ'. The reduction operates clause-by-clause according to the following recurrence relation: for each clause $C \coloneqq (x_1 \vee \cdots \vee x_k)$ in φ (where $k \leq q$),

- If $k < 4$, add the clause C to φ' and halt.
- Otherwise, introduce a "fresh" *auxiliary variable* z, add the clauses $(x_1 \vee x_2 \vee z)$, $(\overline{x_1} \vee \overline{z})$ and $(\overline{x_2} \vee \overline{z})$ to φ', and then recurse on $C' \coloneqq (\overline{z} \vee x_3 \vee \cdots \vee x_k)$.

If φ is not satisfiable then no assignment satisfies more than an $\alpha/3q$ fraction of the clauses of φ' (by the standard soundness of this reduction), so we focus on the case that φ (and therefore φ') is satisfiable. Let A be some assignment for φ', given as an assignment X to the original variables (i.e. to φ) and an assignment Z to the auxiliary variables. Let X^* be a satisfying assignment to φ closest to X. By parsimony of the reduction, there is a unique assignment Z^* to the auxiliary variables such that $A^* \coloneqq (X^*, Z^*)$ is a satisfying assignment.

Again, it would be more convenient for us to deal with absolute quantities rather than relative quantities, so let V_A denote the number of clauses violated by A, and $\overline{\Delta}(A, A^*)$ denote the number of variables on which A and A^* disagree. Similarly define V_X, $\overline{\Delta}(X, X^*)$ and $\overline{\Delta}(Z, Z^*)$. Since φ' has b-bounded variable occurrence then the ratio between its number of variables and number of clauses is at least $1/b$, therefore it suffices to show that $V_A = \Omega(\alpha \cdot \Delta(A, A^*))$.

Now, on the one hand, α-stability of φ means that $V_X \geq \frac{\alpha}{b}\overline{\Delta}(X, X^*)$ (using the assumption that φ has b-bounded variable occurrence to transition from fractional to absolute quantities, just like in the previous paragraph). Additionally, $V_A \geq V_X/3q$ because if X doesn't satisfy a clause in φ then A doesn't satisfy at least one of the corresponding clauses in φ'. Therefore,

$$V_A \geq \frac{\alpha}{3qb} \cdot \overline{\Delta}(X, X^*) \tag{18}$$

On the other hand, the parsimony of the reduction implies that, for any clause C of φ, if X and X^* agree on all q variables occurring in C, and A satisfies all clauses introduced by C (in φ'), then it must be that Z and Z^* agree on all of the auxiliary variables introduced by C. That is, for any clause C, if Z and Z^* disagree on any of the auxiliary variables introduced by C, then either X and X^* disagree on some of the q variables occurring in C, or A violates some of the clauses introduced by C. Therefore,

$$q \cdot \overline{\Delta}(X, X^*) + V_A \geq \overline{\Delta}(Z, Z^*) \tag{19}$$

Combining Equations (18) and (19), we have that $V_A \geq \frac{\alpha}{4qb} \cdot \max\left(\overline{\Delta}(X, X^*), \overline{\Delta}(Z, Z^*)\right)$, and since $\overline{\Delta}(A, A^*) = \overline{\Delta}(X, X^*) + \overline{\Delta}(Z, Z^*)$ we have $V_A \geq \frac{\alpha}{8qb}\overline{\Delta}(A, A^*)$. ◀

B Tedious notes regarding smoothness

B.1 On uniform sampling

Recall that probabilistic Turing machines (and PCP verifiers in particular) obtain their randomness by *tossing random coins*. It is clear how this ability enables uniform sampling from sets of size that is a power of 2, but throughout this work we allow verifiers to sample uniformly random elements from sets of arbitrary (finite) size. Since the technical implementation of such sampling may affect the smoothness of the verifier, a clarification on this matter is due.

To implement a PCP verifier that uses its randomness only to sample a uniformly random element from a set $[N]$, *rejection sampling* may be employed while losing a constant factor in soundness: the verifier samples a uniformly random element $i \in [2^{\lceil \log N \rceil}]$; if $i > N$ the verifier *immediately accepts*, and otherwise it proceeds as intended. Soundness is halved because the sampled i is in $[N]$ with probability $N/2^{\lceil \log N \rceil} \geq 1/2$.

PCP verifiers in this work use their randomness *only* to sample uniformly random elements from a constant number of sets, and are implemented using rejection sampling on the product of sets from which they sample. Namely, a verifier that uniformly samples from sets $\Omega_1, \ldots, \Omega_k$ is implemented using rejection sampling on the set $\Omega_1 \times \cdots \times \Omega_k$. Still, this (only) halves soundness, and adds at most k random coin tosses (as the randomness complexity $r \coloneqq \sum_{i=1}^{k} \log |\Omega_i|$ grows to $\sum_{i=1}^{k} \lceil \log |\Omega_i| \rceil \leq r + k$).

B.2 Smoothness and uniformity

Without loss of generality, we may require that PCP verifiers query each bit in their proof with some positive probability and never query the same location more than once. Any verifier that doesn't satisfy these simplifying assumptions can be transformed into one that does: we add a single uniformly random query, and then modify the verifier so that whenever it attempts to query the same location twice, it uniformly samples from the remaining unqueried locations in the proof instead.

Now here's a thought: Suppose we have a smooth verifier that issues q queries to a proof oracle of length ℓ. We claim that if we look at a set of query locations $I \subseteq [\ell]$ generated by this verifier (based on its random coins), and subsequently choose a uniformly random element $i \in I$ from this set, then i is uniformly distributed in $[\ell]$. Why? Well, smoothness means that the probability that the verifier queries a certain location in the proof oracle in any of its q queries is equal for any certain location (it's equal to q/ℓ, if you must know). By

the previous paragraph, we can assume all queries of the verifier are distinct, so if a certain location is queried by the verifier it's queried exactly once. Therefore, a uniformly random element from the set of query locations is distributed uniformly in the proof.

Smoothness and marginal uniformity

It may seem natural to define smooth PCPs to be those whose queries are marginally uniform.[29] That is, that the first query of the verifier is distributed uniformly in the proof, and so is the second, third, and so on. We claim that there's almost no difference between this notion and ours (Definition 1.3): by the observation made in the previous paragraph, any nonadaptive PCP satisfying Definition 1.3 can be transformed into one that satisfies marginal uniformity by randomly permuting the order of its queries. Actually, even a random cyclic shift would suffice, and only incurs a $\log q$ additive overhead in randomness complexity – exponentially smaller than the original randomness complexity in the constructions used throughout this work.

C The vector-valued low-degree polynomial test

This section was kindly contributed by Oded Goldreich and Madhu Sudan. It sees the generalization of the *Point-line* low-degree polynomial test ([2, Section 7.2]) to the vector-valued setting.

▶ **Definition C.1** (Definition 5.3 restated). *A function $f\colon \mathbb{F}^m \to \mathbb{F}^k$ is a* vector-valued multivariate polynomial of degree at most d *if for all $i \in [k]$ the projection of f to the ith coordinate is a multivariate polynomial of (total) degree at most d, where the projection of f to the ith coordinate, denoted $f_i : \mathbb{F}^m \to \mathbb{F}$, is defined $f_i(x) \coloneqq f(x)_i$ for all $x \in \mathbb{F}^m$.*

▶ **Algorithm C.2** (Algorithm 5.4 restated). The *point-line vector-valued low-degree test (PL-VLDT)* is given access to an oracle $f\colon \mathbb{F}^m \to \mathbb{F}^k$ and a "lines" proof oracle g that maps line $\mathcal{L}$ to vector-valued univariate polynomial $g_{\mathcal{L}} : \mathcal{L} \to \mathbb{F}^k$ of degree at most d. It samples a uniformly random line $\mathcal{L}$ through $\mathbb{F}^m$ and uniformly random point $x \in \mathcal{L}$, and accepts if and only if $f(x) = g_{\mathcal{L}}(x)$.

▶ **Fact C.3.** *For $k = 1$, PL-VLDT is the low degree test of [2, Theorem 65], therefore if input oracle $f\colon \mathbb{F}^m \to \mathbb{F}$ is δ-far from being a polynomial of degree at most d then PL-VLDT rejects with probability at least $\delta/2$ for any lines oracle g.*

▶ **Proposition C.4** (Proposition 5.5 restated). *Assuming $|\mathbb{F}| > 25k$, if input oracle $f\colon \mathbb{F}^m \to \mathbb{F}^k$ is δ-far from being a vector-valued polynomial of degree at most d then for any lines oracle g, PL-VLDT rejects f and g with probability at least $\delta/40$.*

The proof follows three main cases. If a projection of the input is far from being a (scalar-valued) low-degree polynomial then we are done due to Fact C.3. If $\delta = \Omega(1/|\mathbb{F}|)$ then the distance of the restriction $f[\mathcal{L}]$ from being a univariate low-degree univariate polynomial is proportional to the distance of f from being a low-degree polynomial (i.e. δ) with high probability, and assuming $g_{\mathcal{L}}$ is the closest low-degree polynomial to $f[\mathcal{L}]$, rejection occurs with this very probability. The third case is when $\delta = O(1/|\mathbb{F}|)$, and then we capitalize on δ being very small to show that with probability $\Theta(|\mathbb{F}|\delta)$ the restriction $f[\mathcal{L}]$ is low-degree except for exactly one point, which is sampled with probability $1/|\mathbb{F}|$. A formal proof follows.

[29] For example, as used in [1].

Proof. Throughout this proof, *low-degree* means *of degree at most d*. For each $i \in [k]$ let $f_i : \mathbb{F}^m \to \mathbb{F}$ be the projection of f onto its ith coordinate, and let g_i be a mapping of lines through $\mathbb{F}^m$ to low-degree univariate polynomials given by $g_{\mathcal{L},i}(x) \coloneqq g_{\mathcal{L}}(x)_i \in \mathbb{F}$ for each line $\mathcal{L}$ and point $x \in \mathcal{L}$. Let $\widetilde{f}_i \colon \mathbb{F}^m \to \mathbb{F}$ be a low-degree polynomial closest to f_i, and notice that $\widetilde{f} \coloneqq (\widetilde{f}_1, \ldots, \widetilde{f}_k)$ is a vector-valued low-degree polynomial closest to f. Let δ and δ_i be the distances of f from $\widetilde{f}$ and f_i from $\widetilde{f}_i$ respectively.

First, note that if $f_i(x) \neq g_{\mathcal{L},i}(x)$ for the sampled $\mathcal{L}$ and x then the test rejects, therefore the probability that PL-VLDT rejects f and g is greater than the probability that it rejects input oracle f_i and lines oracle $(g_{\mathcal{L},i})_{\mathcal{L}}$. Hence, if there exists i such that $\delta_i \geq 1/5$ then by Fact C.3 rejection occurs with probability at least $1/10$. Therefore, we may assume that $\delta_i < 1/5$ for all $i \in [k]$. We proceed with a partial analysis that assumes that $\delta \leq 2/5$, and later show how the remaining possibilities follow.

Case 1: $\delta > 5/|\mathbb{F}|$. Points on a randomly sampled line $\mathcal{L}$ are pairwise independent and marginally uniform, so the Chebyshev inequality implies that the relative distance between $f[\mathcal{L}]$ and $\widetilde{f}[\mathcal{L}]$ is at least $\delta/2$ and at most $3\delta/2$ with probability greater than $1 - \frac{\delta(1-\delta)}{(\delta/2)^2|\mathbb{F}|} \geq 1/5$. Conditioned on this event, one of two cases must hold:

- **Case 1.1:** There is i such that $g_{\mathcal{L},i} \neq \widetilde{f}_i[\mathcal{L}]$. Note that $g_{\mathcal{L},i}$ and $\widetilde{f}_i[\mathcal{L}]$ are distinct (univariate) polynomials of degree at most d so they agree on at most d points in $\mathcal{L}$. Since $f_i[\mathcal{L}]$ disagrees with $\widetilde{f}_i[\mathcal{L}]$ on at most $3\delta|\mathbb{F}|/2 \leq 3|\mathbb{F}|/5$ points, it holds that $f[\mathcal{L}]$ and $g_{\mathcal{L}}$ agree on at most $d + 3|\mathbb{F}|/5 \leq 4|\mathbb{F}|/5$ points. Therefore rejection occurs with probability at least $4/5 \geq \delta/2$.
- **Case 1.2:** For all i it holds that $g_{\mathcal{L},i} = \widetilde{f}_i[\mathcal{L}]$. Then rejection occurs if and only if $f[\mathcal{L}]$ and $\widetilde{f}[\mathcal{L}]$ disagree on the sampled $x \in \mathcal{L}$, which occurs with probability at least $\delta/2$.

All in all, rejection occurs with probability at least $\frac{1}{5} \cdot \frac{\delta}{2} = \frac{\delta}{10}$.

Case 2: $\delta < 1/2|\mathbb{F}|$. We claim that $f[\mathcal{L}]$ and $\widetilde{f}[\mathcal{L}]$ agree on exactly 1 point with probability at least $|\mathbb{F}|\delta/2$. Again we use pairwise independence and marginal uniformity of points on a random line, this time served with a side of inclusion-exclusion. Let each line $\mathcal{L}$ have a fixed ordering $\mathcal{L} = \{x_1, \ldots, x_{|\mathbb{F}|}\}$. Then,

$$\begin{aligned}
&\mathbb{P}_{\mathcal{L}}\Big[\exists_! \; x \in \mathcal{L} \; f(x) \neq \widetilde{f}(x)\Big] \\
&\quad \geq \sum_{i \in [|\mathbb{F}|]} \mathbb{P}_{x_i}\Big[f(x_i) \neq \widetilde{f}(x_i)\Big] - \sum_{j \in [|\mathbb{F}| \setminus \{i\}]} \mathbb{P}_{x_i, x_j}\Big[f(x_i) \neq \widetilde{f}(x_i), f(x_j) \neq \widetilde{f}(x_j)\Big] \\
&\quad \geq |\mathbb{F}|\big(\delta - |\mathbb{F}|\delta^2\big) = |\mathbb{F}|\delta(1 - |\mathbb{F}|\delta) \geq |\mathbb{F}|\delta/2
\end{aligned}$$

As in Case 1, conditioned on this event rejection occurs with probability at least $4/5 \geq 1/|\mathbb{F}|$ or $1/|\mathbb{F}|$, depending on whether g and $\widetilde{f}$ agree on the random line $\mathcal{L}$. All in all, rejection occurs with probability at least $\frac{|\mathbb{F}|\delta}{2} \cdot \frac{1}{|\mathbb{F}|} = \frac{\delta}{2}$.

Now, if $\delta > 2/5$ we show that there exists $k' < k$ such that the distance of $f_{[k']} \coloneqq (f_1, \ldots, f_{k'})$ from being a vector-valued low-degree polynomial, denoted $\delta_{[k']}$, is greater than $1/5 \geq 5/|\mathbb{F}|$ and less than $2/5$, and since the rejection probability of f and g is greater than that of $f_{[k']}$ and $g_{[k']} \coloneqq (g_1, \ldots, g_{k'})$ we may then apply Case 1 to $f_{[k']}$. Indeed, $\delta_{[k']} - \delta_{[k'-1]} \leq \delta_{k'} \leq 1/5$ and $\delta_{[k]} = \delta \geq 2/5$, so by a greedy argument there must exist $k' \leq k$ such that $\delta_{[k']} \in [1/5, 2/5]$.

Finally, we tend to the case that $\delta \in [1/2|\mathbb{F}|, 5/|\mathbb{F}|]$. If there exists i such that $\delta_i \geq 1/4|\mathbb{F}|$ then by Fact C.3 rejection occurs with probability at least $1/8|\mathbb{F}| \geq \delta/40$. Otherwise, by a greedy argument there exists $k' \leq k$ such that $\delta_{[k']} \in [1/4|\mathbb{F}|, 1/2|\mathbb{F}|]$. Applying Case 2 to $f_{[k']}$, we have that rejection occurs with probability at least $\delta_{[k']}/2 \geq 1/8|\mathbb{F}| \geq \delta/40$. ◀