On the Impossibility of Probabilistic Proofs in Relativized Worlds

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Siqi Liu
sliu18@berkeley.edu
UC Berkeley

December 8, 2019

Abstract

We initiate the systematic study of probabilistic proofs in relativized worlds. The goal is to understand, for a given oracle, if there exist "non-trivial" probabilistic proofs for checking deterministic or nondeterministic computations that make queries to the oracle.

This question is intimately related to a recent line of work that builds cryptographic primitives (e.g., hash functions) via constructions that are "friendly" to known probabilistic proofs. This improves the efficiency of probabilistic proofs for computations calling these primitives.

We prove that "non-trivial" probabilistic proofs relative to several natural oracles do not exist. Our results provide strong complexity-theoretic evidence that certain functionalities cannot be treated as black boxes, and thus investing effort to instantiate these functionalities via constructions tailored to known probabilistic proofs may be inherent.

Keywords: probabilistically checkable proofs; relativization

Contents

1 Introduction

The study of relativized complexity classes originally aspired to shed light on the structural relationships between *un*relativized complexity classes. However, it was soon realized that many interesting complexity classes have contradictory relativization results. For instance, Baker et al. [BGS75] showed that there exist oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$ and $\mathrm{P}^B \neq \mathrm{NP}^B$.

Subsequent works sought to circumvent this difficulty by considering relativized worlds where the oracle is sampled from a "natural" distribution, and thereby avoid specially-crafted oracles that can force an equality/inequality on the complexity classes being compared. For instance, Bennett and Gill [BG81] proved that, with probability 1 over a random oracle R, it holds that $\mathrm{P}^R \neq \mathrm{NP}^R \neq \text{co-NP}^R$ and $\mathrm{P}^R = \mathrm{BPP}^R$. Since these relativization results agreed with what people believed to be true in the unrelativized case, Bennett and Gill proposed the *Random Oracle Hypothesis*, which states that structural relationships between complexity classes that hold with probability 1 over a random oracle also hold in the unrelativized case. However, this hypothesis was later disproved by Chang et al. [CCG$^+$94], who showed that, with probability 1 over a random oracle R, $\mathrm{IP}^R \neq \mathrm{PSPACE}^R$. (We know that, without oracles, IP = PSPACE [LFKN92, Sha92].)

These works indicate that, in general, relativization results are not helpful for understanding the relationships between unrelativized complexity classes. At best they provide us with relativization barriers, which nowadays are not considered so strong since we know of non-relativizing techniques.

1.1 New motivation: the efficiency of probabilistic proofs

We revisit relativization with a new motivation: the efficiency of probabilistic proofs. Superficially, relativization and probabilistic proofs seem unrelated. Yet they are deeply connected, as we explain.

Probabilistic proofs such as interactive proofs [GMR89] and probabilistically checkable proofs [BFLS91] have played important roles in the study of hardness of approximation since the seminal work of [FGL$^+$91]. In recent years, they became the subject of intense study due to their application to constructing highly-efficient cryptographic proofs (such as succinct arguments), and a major research goal today is to improve the efficiency of probabilistic proofs. We now illustrate, via an example, how relativization results tell us important facts about the efficiency of probabilistic proofs.

Example 1.1. Let $\mathcal{H} = \{H_s \colon \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}_{s \in \{0,1\}^*}$ be a family of "hash functions" (the precise security property in this discussion is unimportant), and consider the following NP language:

$$L_s = \{(n, y) \in \mathbb{N} \times \{0,1\}^{|s|} \mid \exists\, x \in \{0,1\}^{|s|} \text{ s.t. } H_s^n(x) = y\} \ .$$

The efficiency measures (proof length, randomness complexity, query complexity, and others) of a PCP for the language L_s typically depend on the size of an arithmetic circuit that iteratively applies H_s, for n times, to a candidate witness x and checks if the result is y. The size of such a circuit is $\Omega(n\,|H_s|)$, i.e., it depends on the size of a circuit for expressing the computation of H_s. Since there are many NP languages of interest to practitioners that involve cryptographic computations such as hash functions, researchers have been designing specialized families of hash functions that can be represented via small arithmetic circuits [AD18, ACG$^+$19, GKK$^+$19, AABS$^+$19, AGP$^+$19, GLR$^+$19].

We ask: *is optimizing the arithmetic circuit complexity of hash functions necessary?*

We now explain why the answer to this question is connected to relativization statements about probabilistic proofs. Informally, suppose that for any family of hash functions $\mathcal{H}$ it holds that $\mathrm{NP}^{\mathcal{H}} \subseteq \mathrm{PCP}^{\mathcal{H}}$. In other words, every language that can be decided by a nondeterministic polynomial-time machine that makes oracle calls to a hash function has a PCP verifier that may make oracle calls to the same hash function. Now the "oracle language" $\mathcal{L} = \{L_s\}_{s \in \{0,1\}^*}$, which is in the relativized complexity class $\mathrm{NP}^{\mathcal{H}}$, can be decided via a computation that involves n calls to the hash function but does not depend on the complexity of the hash function itself (as this computation happens inside the oracle). Hence, since we assumed that $\mathrm{NP}^{\mathcal{H}} \subseteq \mathrm{PCP}^{\mathcal{H}}$, we can obtain a probabilistic proof for $\mathcal{L}$ whose efficiency does *not* depend on the complexity of the hash function (which is wonderful).

In sum, probabilistic proofs that "relativize with respect to hash functions" obviate the need to design hash functions with small complexity and, conversely, negative results about such relativizations provide strong evidence that efforts to design "PCP-friendly" hash functions are inherent.

The above example illustrates a general connection. On the one hand, constructing probabilisitic proofs in relativized worlds could provide drastic efficiency improvements to probabilistic proofs. On the other hand, ruling out probabilistic proofs in relativized worlds would provide a complexity-theoretic justification for why practitioners may be "stuck" with the task of designing "PCP-friendly" realizations of various functionalities (hash functions, signatures, encryption, and so on).

1.2 Our question: are there PCPs for computations in relativized worlds?

We initiate the systematic study of probabilistic proofs for relativized computations.

In this work an *oracle* is a collection $\mathcal{A} = \{\mathcal{A}_n\}_{n \in \mathbb{N}}$ where each $\mathcal{A}_n$ is a distribution over functions on n-bit inputs. A *sample* from $\mathcal{A}$ is a function $A \colon \{0,1\}^* \to \{0,1\}^*$ obtained by sampling a function f_n from each $\mathcal{A}_n$ and then setting A to equal f_n for n-bit inputs. (See Section 2.1 for definitions.)

We wish to understand for what oracles $\mathcal{A}$ there are probabilistically checkable proofs (PCPs) in a relativized world where all machines have oracle access to a sample from $\mathcal{A}$. Below we make this question more precise, distinguishing between the case of PCPs for relativized *nondeterministic* computations ("do PCPs provide any savings in witness length?") and the case of PCPs for relativized *deterministic* computations ("do PCPs provide any savings in computation length?").

The complexity classes that we study are the natural relativized extensions of DTIME, NTIME, and PCP. Note that complexity classes relative to an oracle $\mathcal{A}$ are sets of *oracle languages* (see Definition 2.5) rather than sets of languages, because the sample from $\mathcal{A}$ affects whether a particular instance is in the language or not. The informal definitions below are made precise in Section 2.4.

$\mathrm{DTIME}(t(n))^{\mathcal{A}}$	oracle languages that are decidable by a *deterministic* machine that runs in time $O(t(n))$ and has oracle access to a sample from $\mathcal{A}$
$\mathrm{NTIME}(t(n))^{\mathcal{A}}$	oracle languages that are decidable by a *nondeterministic* machine that runs in time $O(t(n))$ and has oracle access to a sample from $\mathcal{A}$
$\mathrm{PCP}(t(n), q(n))^{\mathcal{A}}$	oracle languages that are decidable by a *PCP verifier* that runs in time $O(t(n))$, makes $O(q(n))$ queries to a proof string, and has oracle access to a sample from $\mathcal{A}$

We introduce two incomparable questions, which concern the (im)possibility of "non-trivial" PCPs for relativized nondeterministic computations and for relativized deterministic computations.

1. **PCPs for NTIME.** For every oracle $\mathcal{A}$ it holds that $\mathrm{NTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(t(n), t(n))^{\mathcal{A}}$ because a PCP verifier can read in full a witness provided in the PCP proof, and then run the nondeterministic decider on the witness. We ask: for what oracles $\mathcal{A}$ can we have *any* non-trivial

improvement on this trivial inclusion? Namely, we consider PCP verifiers that may make $o(t(n))$ queries to the PCP proof, which in general prevents the PCP verifier from reading a witness from the PCP proof. We additionally allow the PCP verifier to incur a polynomial blow-up in running time: it may run in time $\mathrm{poly}(t(n))$, and in particular can make $\mathrm{poly}(t(n))$ queries to the sample from $\mathcal{A}$. (The queries to the PCP proof are still $o(t(n))$.) This amounts to asking:

Given an oracle $\mathcal{A}$, is it the case that $\mathrm{NTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(\mathrm{poly}(t(n)), o(t(n)))^{\mathcal{A}}$*?*

We will say that an oracle $\mathcal{A}$ *separates* NTIME and PCP if the answer to this question is negative.

2. **PCPs for DTIME.** For every oracle $\mathcal{A}$ it holds that $\mathrm{DTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(t(n), 0)^{\mathcal{A}}$ because a PCP verifier can simply run the deterministic decider. We similarly ask: for what oracles $\mathcal{A}$ can we have *any* non-trivial improvement on this trivial inclusion? Namely, we consider PCP verifiers that run in time $o(t(n))$, which in general prevents a PCP verifier from simply running the deterministic decider. We additionally allow the PCP verifier to ask any number of queries to the proof or to the sample from $\mathcal{A}$ (as bounded by its running time). This amounts to asking:

Given an oracle $\mathcal{A}$, is it the case that $\mathrm{DTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(o(t(n)), o(t(n)))^{\mathcal{A}}$*?*

We will say that an oracle $\mathcal{A}$ *separates* DTIME and PCP if the answer to this question is negative.

What is known? Recall that, for unrelativized complexity classes, we have excellent PCPs. All nondeterministic computations have a constant-query PCP verifier that runs in polylogarithmic time: $\mathrm{NTIME}(t(n)) \subseteq \mathrm{PCP}(\mathrm{poly}(n, \log t(n)), O(1))$ [Din07, BS08, Mie09]. In particular, PCPs simultaneously provide exponential savings in witness length and in computation length.

However, for relativized complexity classes, known relativization results tell us very little. The main relevant prior work is by Hartmanis et al. [HCC+92], who claim that, with probability 1 over a random function $R\colon \{0,1\}^* \to \{0,1\}$, $\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), \log n)^R$. This provides a negative result for the special case where $\mathcal{A}$ is a "random oracle", $t(n)$ is polynomially bounded, and the PCP verifier makes $O(\log n)$ queries to the PCP proof. (In Section 1.4 we discuss other related work.)

However, even for the case of a random oracle, our goal is to rule out *any* non-trivial PCP for *any* nondeterministic computation (ruling out any savings in witness length), and also for any *deterministic* computation (ruling out any savings in computation length, even if there is no witness).

More generally, we are interested to answer these questions for oracles beyond random oracles.

Some intuition. If the PCP verifier could "learn" the oracle in a small number of queries, then we may be able to rely on known techniques to construct PCPs for *un*relativized computations because each oracle call could be replaced by a subroutine that simulates the learned oracle. Conversely, if the oracle is "hard" to learn, then known techniques do not seem to apply because it is not clear how they could deal with oracle calls, and so we may expect that non-trivial PCPs in this case are impossible. Our goal will be to show that, for hard-enough oracles, non-trivial PCPs are indeed impossible (regardless of the techniques that could be used to construct the PCPs).

Beyond PCPs. There are several models of probabilistic proofs beyond PCPs, such as interactive proofs (IPs) [GMR89], interactive PCPs (IPCPs) [KR08], and interactive oracle proofs (IOPs) [BCS16, RRR16]. One may ask: why do we focus only on PCPs in our presentation? The answer is that, for the goals of this paper, the PCP model is equivalent to the IOP model (see Remark 1.6), and the IOP model subsumes the other models as special cases. So, for the goals of this paper, it suffices to study PCPs. All results in Section 1.3 directly translate to IPs, IPCPs, and IOPs.

1.3 Our results

We prove that, for several oracles of cryptographic interest, non-trivial PCPs for relativized computations do not exist — this holds both for deterministic computations (DTIME) and for nondeterministic computations (NTIME). Moreover, we establish several structural results about "hard oracles" for PCPs. These initial results provide us with valuable insights into the efficiency limitations of PCPs, and provide a useful starting point for further investigations into this new direction.

We now summarize our results in more detail.

(1) Random functions. We begin with the oracle that intuition suggests is the "hardest" oracle for PCPs because it is "maximally unlearnable", the *random oracle*. Namely, we consider the oracle $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{R}_n$ is the uniform distribution over all functions $R_n \colon \{0,1\}^n \to \{0,1\}$.[1] Our first result shows that the intuition is correct, i.e., we prove that the oracle $\mathcal{R}$ separates DTIME and PCP and also separates NTIME and PCP.

Theorem 1.2 (informal). *Let $\mathcal{R}$ be the random oracle. For any $t \colon \mathbb{N} \to \mathbb{N}$,*

$$\mathrm{DTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{R}} \quad \textit{and} \quad \mathrm{NTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}} \ .$$

The above theorem tells us that we cannot, in general, expect to construct PCPs for cryptographic computations that involve random oracles, such as Fiat–Shamir signatures [FS86]. The natural alternative would be to somehow instantiate the random oracle, and incur, within the PCP, the cost of the hash function used in place of the random oracle. This is indeed what Valiant [Val08] did in his construction of *incrementally verifiable computation* (IVC): Valiant needed to construct a PCP for the computation of a SNARK verifier that uses random oracles and, lacking suitable PCPs for this relativized computation, considered instead the SNARK verifier obtained by instantiating the random oracle. Our Theorem 1.2 rules out PCPs for computations that use random oracles, and in particular gives strong evidence that Valiant's approach was in some sense justified.

One may argue that, while they give us useful insights, random oracles do not tell us much about other oracles because they are too special in that they have no structure. We now consider two oracles with structure: one with group structure and another with low-degree structure.

(2) Random generic groups. Many group-based cryptographic primitives are stated (and sometimes also analyzed) with respect to a *generic group*. This means that the primitive relies only on the fact that a certain prime-order group is available but does not rely on whether the group is instantiated, say, with a multiplicative subgroup of a finite field or an elliptic curve group. This motivates the question of whether there are PCPs with respect to a *random (generic) group oracle*, which is the oracle $\mathcal{O} = \{\mathcal{O}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{O}_n$ is a random presentation of a group of order n. We prove that the answer is negative, i.e., that the oracle $\mathcal{O}$ separates DTIME and PCP, and also separates NTIME and PCP.

Theorem 1.3 (informal). *Let $\mathcal{O}$ be the random group oracle. For any $t \colon \mathbb{N} \to \mathbb{N}$,*

$$\mathrm{DTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{O}} \quad \textit{and} \quad \mathrm{NTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}} \ .$$

The above theorem tells us that, in general, the representation of a group matters to a PCP. For example, if we return to the iterative hash computation of Example 1.1 and set the hash function to

[1]More generally, we consider the uniform distribution over functions $R_n \colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$ for some $\ell(n)$.

be the Pedersen hash function (a function that is collision resistant over any group where extracting discrete logarithms is hard), we should pick a group tailored to the PCP at hand. This is consistent with the fact that applied cryptographers working with probabilistic proofs have had to carefully design group instantiations for such computations. E.g., `Jubjub` [ZCa17] is an elliptic curve in the Zcash cryptocurrency that is used to instantiate a Pedersen hash function in a way that is "friendly" to probabilistic proofs. Our theorem provides strong evidence that these efforts are necessary.

(3) Random low-degree functions. Probabilistic proofs are typically achieved by relying on low-degree functions that encode information associated to the computation being checked. This fact extends to relativized complexity classes in the sense that results such as IP = PSPACE *algebrize* with respect to every oracle [AW09]. In the language of this paper, this means that for every oracle $\mathcal{A}$, it holds that $\mathrm{PSPACE}^{\mathcal{A}} \subseteq \mathrm{IP}^{\hat{\mathcal{A}}}$ where $\hat{\mathcal{A}}$ is the low-degree extension of $\mathcal{A}$ (each sample in $\mathcal{A}$ is replaced with some low-degree extension of it).

However in this paper we are interested in understanding relativization results, not algebrization results, and the above discussion raises the question of what happens when we compare DTIME/NTIME and PCP in a relativized world where the oracle is a random low-degree function. Namely, we consider the oracle $\mathcal{P} = \{\mathcal{P}_n\}_{n\in\mathbb{N}}$ where $\mathcal{P}_n$ is the uniform distribution over all low-degree polynomials on n variables (for given field and degree parameters). Note that $\mathcal{P}$ can be viewed as a low-degree extension of the random oracle $\mathcal{R}$.

We prove that low-degree structure remains hard for probabilistic proofs, i.e., we prove that $\mathcal{P}$ separates DTIME and PCP, and also separates NTIME and PCP.

Theorem 1.4 (informal)**.** *Let $\mathcal{P}$ be the random low-degree oracle over q-size fields. For any $t\colon \mathbb{N} \to \mathbb{N}$,*

$$\mathrm{DTIME}(t)^{\mathcal{P}} \not\subseteq \mathrm{PCP}(o(t^{\log\frac{q}{q-1}}), o(t^{\log\frac{q}{q-1}}))^{\mathcal{P}} \quad \textit{and} \quad \mathrm{NTIME}(t)^{\mathcal{P}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}} \ .$$

We point out here that it is still open whether $\mathrm{DTIME}(t)^{\mathcal{P}}$ is in $\mathrm{PCP}(o(t), o(t))^{\mathcal{P}}$.

Interlude on separation types. We have so far considered relativized complexity classes in which a single machine is granted oracle access to a sample $A\colon \{0,1\}^* \to \{0,1\}^*$ from the oracle $\mathcal{A}$, and is required to "work" for the language defined by A with probability 1 over the choice of A. For example, $\mathrm{DTIME}(t)^{\mathcal{A}}$ is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which there exists a deterministic machine M, which runs in time $O(t(n))$, such that

$$\Pr_{A\leftarrow\mathcal{A}}\left[M^A \text{ decides the language } L_A \right] = 1 \ .$$

We use analogous definitions for NTIME and PCP, as discussed in Section 2.4. We consider these definitions to be the natural ones to use for the goals of this paper. We sometimes refer to separations between these complexity classes as *uniform separations*, to distinguish them from those below.

We could alternatively study separations where all machines are allowed to *non-uniformly depend* on A, thereby granting all machines more power. In this direction, there are two natural definitions.

- *Somewhere separation.* We say that $\mathcal{A}$ provides a somewhere separation for DTIME and PCP if there exists $A \in \mathcal{A}$ such that $\mathrm{DTIME}(t)^A \not\subseteq \mathrm{PCP}(o(t), o(t))^A$. And similarly for NTIME.

- *Almost-everywhere separation.* We say that $\mathcal{A}$ provides an almost-everywhere separation for DTIME and PCP if $\mathrm{DTIME}(t)^A \not\subseteq \mathrm{PCP}(o(t), o(t))^A$ holds with probability 1 over a random choice of sample $A \leftarrow \mathcal{A}$. (Note that this leaves open the possibility that there is no separation for a set of functions of measure 0 in $\mathcal{A}$.) And similarly for NTIME.

An almost-everywhere separation is, in general, strictly stronger than an a somewhere separation. However, the relation between these and uniform separations is not a priori clear.

We provide clarity on this comparison: in Section 4 we prove that uniform separations are equivalent to somewhere separations, at least when comparing DTIME/NTIME and PCP. Namely, we prove that, for any oracle $\mathcal{A}$, $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{A}}$ if and only if there exists a function A in $\mathcal{A}$ such that $\mathrm{DTIME}(t)^{A} \not\subseteq \mathrm{PCP}(o(t), o(t))^{A}$. And similarly for NTIME.

Almost-everywhere separation for random functions. For the random oracle $\mathcal{R}$, we strengthen the separations in Theorem 1.2 to almost-everywhere separations (via a different, longer proof). We learn that the random oracle $\mathcal{R}$ is particularly "hard" for PCPs.

Theorem 1.5 (informal)**.** *Let $\mathcal{R}$ be the random oracle. For any $t \colon \mathbb{N} \to \mathbb{N}$,*

$$\Pr_{R \leftarrow \mathcal{R}}\left[\, \mathrm{DTIME}(t)^{R} \not\subseteq \mathrm{PCP}(o(t), o(t))^{R} \,\right] = 1 \text{ and } \Pr_{R \leftarrow \mathcal{R}}\left[\, \mathrm{NTIME}(t)^{R} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{R} \,\right] = 1 \ .$$

The above theorem also directly improves on the classical work of Hartmanis et al. [HCC⁺92], who showed that $\Pr_{R \leftarrow \mathcal{R}}[\, \mathrm{NP}^{R} \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), \log n)^{R} \,] = 1$. The improvement is that our result rules out any non-trivial PCP for any nondeterministic computation, and also rules out any non-trivial PCP for any deterministic computation.

We prove Theorem 1.5 by building on techniques of Chang et al. [CCG⁺94] that were used to prove that $\Pr_{R \leftarrow \mathcal{R}}[\, \mathrm{IP}^{R} \neq \mathrm{PSPACE}^{R} \,] = 1$. These techniques rely on the fact that every function R in $\mathcal{R}$ has many other functions in $\mathcal{R}$ that are close to it in all but finitely many points.

We do not know how to extend these techniques to oracles such as the random group oracle $\mathcal{O}$ or the random low-degree oracle $\mathcal{P}$, because in these cases any two samples are far from one another. In this light, we view the techniques that we use to prove Theorems 1.2 to 1.4 as more flexible. Moreover, we consider the separations proved in these theorems as sufficient for our motivations.

Structural results: beyond $\mathcal{R}$, $\mathcal{O}$, $\mathcal{P}$. Our results thus far concern separations for specific oracles of interest. There are other oracles of interest that demand understanding (e.g., pseudorandom functions) and, more generally, the study of separations could benefit from general statements. In Section 3 we prove several useful structural results about oracles that are "hard" for PCPs.

1. *Robustness.* We prove that the separating property is "robust" with respect to small perturbations. In more detail, we prove that for every oracle $\mathcal{A}$ that separates DTIME/NTIME and PCP there exists a distance function ϵ such that any other oracle that is ϵ-close to $\mathcal{A}$ also separates DTIME/NTIME and PCP. This statement can also be viewed as telling us that the set of separating oracles is *open* with respect to statistical distance (see Definition 2.2).

   We can apply the above result to any of the separations that we have proved. For example, if apply it to Theorem 1.2 then we learn that all oracles that are "almost" uniformly random (close enough to the random oracle $\mathcal{R}$) separate DTIME/NTIME and PCP. In fact, in Lemma 9.1, we use additional techniques to quantify (a bound on) this distance threshold, proving that all oracles that are $\frac{1}{3e}$-close to uniformly random separate NTIME and PCP.

2. *Monotonicity.* We prove that the separating property is "monotone" in that, for every oracle $\mathcal{A}$ that separates DTIME/NTIME and PCP, if another oracle $\mathcal{B}$ contains $\mathcal{A}$ as a marginal distribution then $\mathcal{B}$ also separates DTIME/NTIME and PCP. I.e., $\mathcal{B}$ inherits the hardness of $\mathcal{A}$.

   We rely on monotonicity in the proof of Theorem 1.4, where we reduce the problem of showing separation for random *low-degree* polynomials to the problem of showing separation for random *multilinear* polynomials (which we then solve).

3. *Conditioning.* We prove that the separating property is preserved by (finite) conditioning. Namely, given an oracle $\mathcal{A}$ and a function $f\colon D \to \{0,1\}^*$ over a finite domain D, we denote by $\mathcal{A}^{D,f}$ the oracle where samples are conditioned to equal f on D. We prove that if $\mathcal{A}$ separates DTIME/NTIME and PCP then $\mathcal{A}^{D,f}$ also separates DTIME/NTIME and PCP.

   We rely on conditioning to prove the equivalence in Section 4 (between different separation types).

Remark 1.6 (beyond PCPs)**.** Research in the last few years has shown that using known PCPs is not the best choice for constructing efficient succinct arguments. Instead, it is better to construct succinct arguments from IOPs [BCS16, RRR16], which are a multi-round generalization of PCPs that enables significant improvements in asymptotic and concrete efficiency [BCGV16, BCG+17, BCF+17, BBC+17, BBHR18, BBHR19, BCR+19, BCG+19, RR19]. So the reader may rightfully ask: why did we prove all of our results for PCPs instead of IOPs, if these latter are more powerful?

The answer is that *all of our results extend, in a generic way, to IOPs as well.* This is because our results about PCPs only consider the PCP verifier's time complexity and query complexity, and IOPs do *not* provide any benefits over PCPs when only considering these complexity measures. Indeed, any IOP can be "unrolled" into a PCP, possibly of exponentially larger size, while preserving the verifier's time complexity and query complexity, regardless of oracle. In particular, for every oracle $\mathcal{A}$, the complexity class $\mathrm{IOP}(T,q)^{\mathcal{A}}$ (oracle languages for $\mathcal{A}$ decidable by an IOP verifier with time complexity T and query complexity q) equals the complexity class $\mathrm{PCP}(T,q)^{\mathcal{A}}$ (oracle languages for $\mathcal{A}$ decidable by a PCP verifier with time complexity T and query complexity q).

Finally, we additionally obtain analogous impossibility results for interactive proofs (IPs) [GMR89] and interactive PCPs (IPCPs) [KR08], as both are special cases of IOPs.

1.4 Related work

NP vs. PCP in relativized worlds. Fortnow [For94] uses diagonalization to obtain a function $R\colon \{0,1\}^* \to \{0,1\}$ such that, for every $k \in \mathbb{N}$, $\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), n^k)^R$. Hartmanis et al. [HCC+92] report a stronger result: with probability 1 over a random function $R\colon \{0,1\}^* \to \{0,1\}$, $\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), \log n)^R$. We do not know of a version of [HCC+92] that contains a proof of this result, so we cannot comment on the techniques used to prove it. As already discussed, our Theorem 1.5 strengthens this latter result to hold for any non-trivial PCP.

Barriers for relativization and others. PCP constructions involve the use of non-relativizing techniques. A line of works [For94, AIV92, AW09, IKK09, AB18] has developed frameworks that seek to capture the class of such techniques, along with other non-relativizing ones, within formal models, in order to prove barriers for these techniques (e.g., to show that they do not suffice to resolve the P vs. NP question or other difficult questions in complexity theory).

The emphasis and techniques in this work are complementary to the foregoing line of works.

Our emphasis is on establishing impossibility results for PCPs *regardless* of techniques used, as opposed to proving barriers for the PCP techniques that are known today.

Moreover, the axiomatic approaches employed in some of the works cited above cannot be used to even formulate questions that involve PCP verifiers with specific running times or query complexities. E.g., they rely on Cobham's axiomatization of the notion of polynomial time [Cob65], so cannot express exact running times. This means that we would not be able to phrase questions about non-trivial PCPs (as we do in Section 1.2).

1.5 Open problems

The separations that we prove in this paper are for "information-theoretic" oracles $\mathcal{A}$. What can be said about hard "cryptographic" oracles $\mathcal{A}$? E.g., if $\mathcal{A}$ is a pseudo-random function, then must it be the case that $\mathcal{A}$ separates DTIME/NTIME and PCP? What about if $\mathcal{A}$ is a decryption oracle?

More generally, the holy grail in this research direction would be to distill a crisp, and operationally useful, criterion that gives sufficient and necessary condition for an oracle $\mathcal{A}$ that separates DTIME/NTIME and PCP.

2 Definitions

We often consider functions from natural numbers to natural numbers, and we implicitly assume that they are time-constructible. Recall that $f\colon \mathbb{N} \to \mathbb{N}$ is time-constructible if there is a machine M such that, for all $n \in \mathbb{N}$, $M(1^n)$ outputs the binary representation of $f(n) \in \mathbb{N}$ in $O(f(n))$ time. Note that if f is time-constructible then $f(n) \in \Omega(n)$.

2.1 Oracles

An oracle is a collection of distributions over functions, with one distribution per input length.

Definition 2.1. *An* **oracle** *with output length $\ell\colon \mathbb{N} \to \mathbb{N}$ (with $\ell(n) > 0$ for every $n \in \mathbb{N}$) is a collection $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{A}_n$ is a distribution over functions $f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$.*

We can obtain a *sample* $A\colon \{0,1\}^* \to \{0,1\}^*$ from $\mathcal{A}$ by sampling a function f_n from each $\mathcal{A}_n$ and then setting A to equal f_n for inputs of size n. We write "$A \leftarrow \mathcal{A}$" to denote that A is a sample that follows this distribution, and "$A \in \mathcal{A}$" to denote that A is in the support of this distribution. We denote by $\operatorname{supp}(\mathcal{A})$ the support of $\mathcal{A}$. (In particular, the oracles that we consider are product distributions across input sizes. We leave the study of a more general definition to other work.)

An oracle $\mathcal{A}$ induces a corresponding probability measure $\mu_{\mathcal{A}}$ over the space of functions from binary strings to binary strings: given a subset $S \subseteq \{0,1\}^*$ and a set X of functions from S to $\{0,1\}^*$, $\mu_{\mathcal{A}}(X)$ is the probability that the restriction to S of a sample A from $\mathcal{A}$ belongs to X.

Definition 2.2. *Two oracles $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ and $\mathcal{B} = \{\mathcal{B}_n\}_{n\in\mathbb{N}}$ have (statistical) distance $\epsilon\colon \mathbb{N} \to \mathbb{N}$ if, for every $n \in \mathbb{N}$, the statistical distance between the distributions $\mathcal{A}_n$ and $\mathcal{B}_n$ is at most $\epsilon(n)$.*

We write "$g \leftarrow \mathcal{A}_{\le n}$" to denote that g is a function on $\bigcup_{1\le i\le n}\{0,1\}^i$ that is sampled from the distribution $\mathcal{A}_{\le n} := \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$. We write "$g \in \mathcal{A}_{\le n}$" to denote that g is in the support of this distribution, and denote by $\operatorname{supp}(\mathcal{A}_{\le n})$ the support of $\mathcal{A}_{\le n}$.

Definition 2.3. *An oracle $\mathcal{B}$* **contains** *an oracle $\mathcal{A}$ if for all $n \in \mathbb{N}$ it holds that $\mu_{\mathcal{B}}(\operatorname{supp}(\mathcal{A}_{\le n})) > 0$ and, for every $f \in \operatorname{supp}(\mathcal{A}_{\le n})$, $\mu_{\mathcal{B}}(f) = \mu_{\mathcal{B}}(\operatorname{supp}(\mathcal{A}_{\le n})) \cdot \mu_{\mathcal{A}}(f)$.*

The definition below provides an operation to *condition* an oracle to take known values.

Definition 2.4. *Fix a subset $D \subseteq \{0,1\}^*$ and function $f\colon D \to \{0,1\}^*$.*

- *Given a function $A\colon \{0,1\}^* \to \{0,1\}^*$, we define $A^{D,f}\colon \{0,1\}^* \to \{0,1\}^*$ to be the function obtained by setting the values of A on D to f:*

$$A^{D,f} = \begin{cases} f(x) & \text{if } x \in D \\ A(x) & \text{if } x \notin D \end{cases} .$$

- *Given an oracle $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ (such that there exists some $A \in \operatorname{supp}(\mathcal{A})$ agreeing with f on D), we define $\mathcal{A}^{D,f} = \{\mathcal{A}_n^{D,f}\}_{n\in\mathbb{N}}$ to be the oracle where samples are conditioned to equal f on D. In more detail, each distribution $\mathcal{A}_n^{D,f}$ equals the distribution $\mathcal{A}_n$ conditioned on the event that the sampled function agrees with f on $D \cap \{0,1\}^n$.*

2.2 Languages and oracle languages

A language L is a subset of $\{0,1\}^*$. We denote by $L(\mathbb{x})$ the bit that specifies whether a string $\mathbb{x} \in \{0,1\}^*$ is in L ($L(\mathbb{x}) = 1$) or not ($L(\mathbb{x}) = 0$).

We also consider oracle languages because we study relativized complexity classes.

Definition 2.5. *Let $\mathcal{U} := \{F \colon \{0,1\}^* \to \{0,1\}^*\}$ be the set of all functions on binary strings. An* **oracle language** *$\mathcal{L}$ is a collection of languages indexed by functions $F \in \mathcal{U}$, namely, $\mathcal{L} = \{L_F\}_{F \in \mathcal{U}}$ where each L_F is a subset of $\{0,1\}^*$.*

A language L can be viewed as a special case of an oracle language $\{L_F\}_{F \in \mathcal{U}}$ where each $L_F = L$.

Definition 2.6. *For $\ell \colon \mathbb{N} \to \mathbb{N}$, an oracle language $\mathcal{L} = \{L_F\}_{F \in \mathcal{U}}$ is* **ℓ-bounded** *if, for all functions $F \in \mathcal{U}$ and inputs $\mathbb{x} \in \{0,1\}^*$, whether $\mathbb{x} \in L_F$ only depends on F's values at locations of size at most $\ell(|\mathbb{x}|)$. Namely, $L_F(\mathbb{x}) = L_{F'}(\mathbb{x})$ for every F' that agrees with F on the set $\bigcup_{1 \le i \le \ell(|\mathbb{x}|)} \{0,1\}^i$.*

2.3 Machines that query oracles

We consider several notions of (Turing) machines that query oracles, as defined below. Informally, an *oracle machine* is given black-box access to a function $A \colon \{0,1\}^* \to \{0,1\}^*$, which the machine can query, any number of times, at any input of its choice. Each query costs the machine a single computational step, regardless of the function A. In more detail, we consider the following definition.

Definition 2.7. *An* **oracle machine** *M is a machine that has two special tapes called* oracle query tape *and* oracle answer tape*, and two special states called* Query *and* Answer*. The special tapes are in addition to the machine's regular read/write tapes (of which there can be one or multiple) and the special states are in addition to the machine's regular start, accept, reject, and other states. We denote by $M^A(\mathbb{x})$ the output of M on input $\mathbb{x} \in \{0,1\}^*$ and with access to oracle $A \colon \{0,1\}^* \to \{0,1\}^*$, which is computed as follows. The input $\mathbb{x}$ is written in a designated read/write tape, and execution proceeds as normal except if the machine enters the* Query *state. Let $y \in \{0,1\}^*$ be the contents of the oracle query tape when this happens. In the following step, the contents of the oracle answer tape are replaced with $A(y) \in \{0,1\}^*$, and the machine enters the* Answer *state.*

Definition 2.7 considers *deterministic* oracle machines. In Section 2.4 we use these machines to extend the notion of languages decidable in deterministic bounded time to work with oracles.

We also use *nondeterministic* oracle machines, which are defined similarly as above except that they can, in any computational step, choose to make a nondeterministic choice as in the standard definition of a nondeterministic machine. In Section 2.4 we use these machines to extend the notion of languages decidable in nondeterministic bounded time to work with oracles.

We also use *probabilistic* oracle machines that use randomness and can make queries to a proof string $\pi \in \{0,1\}^*$ (in addition to the oracle A). These machines are defined similarly as above except that they can, in any computational step, receive a bit of randomness, or query a location of the proof string π via two dedicated tapes (a *proof query tape* and a *proof answer tape*). In Section 2.4 we use these machines to extend the notion of probabilistic proofs to work with oracles.

Throughout this paper we call oracle machines simply "machines", as it will be clear from context when we are referring to an oracle machine (of one of the foregoing types).

Remark 2.8. Oracle machines are often defined with *one* special tape, instead of *two* as in Definition 2.7. The machine writes its query to the oracle in this one tape, and in the following step the tape's contents are *replaced* with the oracle's answer. Note that, with one oracle tape, the machine has to write each query "from scratch" because the prior query was deleted. This difference is not significant, because writing each query from scratch is at most quadratically slower than not having to do that (due to having two oracle tapes). However, this difference matters when studying questions that are sensitive to such costs. Thus in this paper we use machines with two oracle tapes.

2.4 Complexity classes with oracles

We define, for a given oracle $\mathcal{A}$, the complexity classes $\mathrm{DTIME}(t)^{\mathcal{A}}$, $\mathrm{NTIME}(t)^{\mathcal{A}}$, and $\mathrm{PCP}(t,q)^{\mathcal{A}}$.

Deterministic time. A deterministic machine M is a *D-decider* for a language L if for every $\mathbb{x} \in \{0,1\}^*$ it holds that $M(\mathbb{x}) = L(\mathbb{x})$. The complexity class $\mathrm{DTIME}(t)$ consists of all languages L for which there exists a deterministic machine M that runs in time $O(t(n))$ and is a D-decider for L. We now provide a definition that considers the more general case of oracle languages that are decidable by deterministic machines with access to an oracle.

Definition 2.9. *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be an oracle and let $t\colon \mathbb{N} \to \mathbb{N}$ be a function.* $\mathrm{DTIME}(t)^{\mathcal{A}}$ *is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which there exists a deterministic machine M, which runs in time $O(t(n))$, such that*

$$\Pr_{A\leftarrow\mathcal{A}}\left[M^A \textit{ is a D-decider for } L_A \right] = 1 \ .$$

Nondeterministic time. A nondeterministic machine M is a *ND-decider* for a language L if for every $\mathbb{x} \in \{0,1\}^*$ it holds that $M(\mathbb{x}) = L(\mathbb{x})$. The complexity class $\mathrm{NTIME}(t)$ consists of all languages L for which there exists a nondeterministic machine M that runs in time $O(t(n))$ and is a ND-decider for L. We now provide a definition that considers the more general case of oracle languages that are decidable by nondeterministic machines with access to an oracle.

Definition 2.10. *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be an oracle and let $t\colon \mathbb{N} \to \mathbb{N}$ be a function.* $\mathrm{NTIME}(t)^{\mathcal{A}}$ *is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which there exists a nondeterministic machine M, which runs in time $O(t(n))$, such that*

$$\Pr_{A\leftarrow\mathcal{A}}\left[M^A \textit{ is a ND-decider for } L_A \right] = 1 \ .$$

Probabilistic proofs. A probabilistic machine M is a *PCP-verifier* for a language L if: for every $\mathbb{x} \in L$ there exists $\pi \in \{0,1\}^*$ such that $\Pr[M^\pi(\mathbb{x}) = 1] \geq 2/3$; for every $\mathbb{x} \notin L$ and $\pi \in \{0,1\}^*$ it holds that $\Pr[M^\pi(\mathbb{x}) = 0] \geq 2/3$. The complexity class $\mathrm{PCP}(t,q)$ consists of all languages L for which there exists a probabilistic machine M that runs in time $O(t(n))$, makes $O(q(n))$ queries to the proof string, and is a PCP-verifier for L. Below we consider the more general case of oracle languages that are decidable by probabilistic machines with access to an oracle.

Definition 2.11. *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be an oracle and let $t, q\colon \mathbb{N} \to \mathbb{N}$ be functions.* $\mathrm{PCP}(t,q)^{\mathcal{A}}$ *is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which the there exists a probabilistic machine M, which runs in time $O(t(n))$ and makes $O(q(n))$ queries to the proof string, such that*

$$\Pr_{A\leftarrow\mathcal{A}}\left[M^A \textit{ is a PCP-verifier for } L_A \right] = 1 \ .$$

Given an oracle language $\mathcal{L} = \{L_A\}_{A \in \mathcal{A}}$ and machine M, we say that M **succeeds** on input $\mathbb{x} \in \{0,1\}^*$ and function $A \in \mathcal{A}$ for $\mathcal{L}$ if the following holds: if $\mathbb{x} \in L_A$ then $\Pr[M^{A,\pi}(\mathbb{x}) = 1] \geq \frac{2}{3}$ for some proof π; else if $\mathbb{x} \notin L_A$ then $\Pr[M^{A,\pi}(\mathbb{x}) = 1] \leq \frac{1}{3}$ for every proof π. Otherwise, we say that M **fails** on input $\mathbb{x} \in \{0,1\}^*$ and function $A \in \mathcal{A}$ for $\mathcal{L}$ (that is, if $\mathbb{x} \in L_A$ then $\Pr[M^{A,\pi}(\mathbb{x}) = 1] < \frac{2}{3}$ for every proof π; else if $\mathbb{x} \notin L_A$ then $\Pr[M^{A,\pi}(\mathbb{x}) = 1] > \frac{1}{3}$ for some proof π).

We conclude this section with several technical remarks.

Remark 2.12 (bounded oracle languages)**.** For every oracle $\mathcal{A}$ and oracle language $\mathcal{L}$ in the complexity class $\mathrm{DTIME}(t)^{\mathcal{A}}$, $\mathrm{NTIME}(t)^{\mathcal{A}}$, or $\mathrm{PCP}(t,q)^{\mathcal{A}}$, there exists a $O(t)$-bounded oracle language $\mathcal{L}^*$ such that for every oracle $A \in \mathcal{A}$ it holds that $L_A = L^*_A$. The language $\mathcal{L}^*$ is naturally defined by $\mathcal{L}$'s $\mathrm{DTIME}^{\mathcal{A}}$ decider, $\mathrm{NTIME}^{\mathcal{A}}$ decider, or $\mathrm{PCP}^{\mathcal{A}}$ verifier. In particular, we can assume without loss of generality that oracle languages in these complexity classes are $O(t)$-bounded.

Remark 2.13 (index over $\mathcal{A}$ instead of $\mathcal{U}$)**.** We sometimes define an oracle language $\mathcal{L}$ in $\mathrm{DTIME}(t)^{\mathcal{A}}$ or in $\mathrm{NTIME}(t)^{\mathcal{A}}$ only for functions in (the support of) an oracle $\mathcal{A}$. In this case it is understood that L_F is defined by the $\mathrm{DTIME}^{\mathcal{A}}$ or $\mathrm{NTIME}^{\mathcal{A}}$ decider of $\{L_A\}_{A \in \mathcal{A}}$ for all functions $F \in \mathcal{U} \setminus \mathcal{A}$.

Remark 2.14 (relativized classes for a single function)**.** Definitions 2.9 to 2.11 capture, as a special case, relativized classes where the oracle is a single function $A \colon \{0,1\}^* \to \{0,1\}^*$ rather than a distribution over functions (let $\mathcal{A}$ be the oracle that puts all the probability mass on A). In this case the relativized classes can be "collapsed" to sets of languages rather than sets of oracle languages.

3 Structural properties of separation

We prove structural properties about the non-containments $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$ (Theorem 3.1) and $\mathrm{NTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$ (Theorem 3.2). We use these in later sections.

Theorem 3.1 (DTIME & PCP)**.** *Let $t, T\colon \mathbb{N} \to \mathbb{N}$ be time bound functions and $q\colon \mathbb{N} \to \mathbb{N}$ a query bound function. Let $\mathcal{A}$ be an oracle such that $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$. Then the following holds.*

(1) Robustness: *there exists a positive function $\epsilon\colon \mathbb{N} \to \mathbb{R}$ ($\epsilon(n) > 0$ for every $n \in \mathbb{N}$) such that, for every oracle $\mathcal{B}$ that is ϵ-close to $\mathcal{A}$, it also holds that $\mathrm{DTIME}(t)^{\mathcal{B}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{B}}$.*

(2) Monotonicity: *for every oracle $\mathcal{B}$ that contains $\mathcal{A}$, it also holds that $\mathrm{DTIME}(t)^{\mathcal{B}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{B}}$.*

(3) Conditioning: *for every function $f\colon D \to \{0,1\}^*$, it also holds that $\mathrm{DTIME}(t)^{\mathcal{A}^{D,f}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}^{D,f}}$.*

Theorem 3.2 (NTIME & PCP)**.** *Theorem 3.1 also holds with $\mathrm{NTIME}(t)$ in place of $\mathrm{DTIME}(t)$.*

The above theorems are direct corollaries of general properties that we prove, as we now explain.

For the rest of this section we fix: (a) an oracle language $\mathcal{L}$ that is ℓ-bounded for some $\ell\colon \mathbb{N} \to \mathbb{N}$; (b) a time bound function $T\colon \mathbb{N} \to \mathbb{N}$; (c) a query bound function $q\colon \mathbb{N} \to \mathbb{N}$; (d) an oracle $\mathcal{A}$.

We shall provide an equivalent formulation for the condition "$\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$" (Claim 3.5), and then use it to derive several general properties about this condition: robustness (Lemma 3.6), monotonicity (Lemma 3.7), and conditioning (Lemma 3.8).

By taking $\mathcal{L}$ to be an oracle language in either $\mathrm{DTIME}(t)^{\mathcal{A}}$ or $\mathrm{NTIME}(t)^{\mathcal{A}}$ (which implies that the oracle language is $O(t)$-bounded), we can then derive the corresponding property in Theorem 3.1 or Theorem 3.2 respectively. We are left to state and prove the claim and lemmas mentioned above.

Definition 3.3. *We denote by $\mathbf{M}_{T,q}$ the set of probabilistic oracle machines that, on inputs of length n, read $O(q(n))$ proof bits and run in $O(T(n))$ time.*

Definition 3.4. *For every $n \in \mathbb{N}$, we define $s(n) := \max\{\ell(n), 2^{T(n)}\}$ and $S_n := \bigcup_{1 \le i \le s(n)} \{0,1\}^i$.*

Claim 3.5. *The following two conditions are equivalent:*

(1) $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$.

(2) For every machine $M \in \mathbf{M}_{T,q}$ there exist an input $\mathbb{x} \in \{0,1\}^*$ and function $f\colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}}(f) > 0$ such that, for every function $F\colon \{0,1\}^* \to \{0,1\}^*$ that agrees with f on $S_{|\mathbb{x}|}$, M fails on input $\mathbb{x}$ and function F for $\mathcal{L}$. (The function F need not be in $\mathrm{supp}(\mathcal{A})$.)

Proof. We separately consider the two directions.

(1) ⇒ (2). If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then for every $M \in \mathbf{M}_{T,q}$ there exists a function $F \in \mathrm{supp}(\mathcal{A})$ such that M^F fails to verify L_F on some input $\mathbb{x}$. Consider the function $f\colon S_{|\mathbb{x}|} \to \{0,1\}^*$ obtained by restricting F to $S_{|\mathbb{x}|}$. Since the running time of M^F on input $\mathbb{x}$ is at most $O(T(|\mathbb{x}|))$, it cannot distinguish between having access to the function F and access to any other function F' that agrees with f. Moreover, since $\mathcal{L}$ is ℓ-bounded, $L_F(\mathbb{x}) = L_{F'}(\mathbb{x})$ for every F' that agrees with f. Therefore, $M^{F'}(\mathbb{x})$ fails for every F' that agrees with f. The set of all such oracles has positive measure in $\mathcal{A}$ (i.e., $\mu_{\mathcal{A}}(f) > 0$), because any finite prefix of any function in $\mathcal{A}$ has positive measure.

(2) ⇒ (1). The condition directly implies that, for every $M \in \mathbf{M}_{T,q}$, M^A is not a PCP-verifier for L_A for a set of functions A with positive measure in $\mathcal{A}$. Therefore $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$. □

Lemma 3.6 (robustness)**.** *If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then there exists a positive function $\epsilon \colon \mathbb{N} \to \mathbb{R}$ ($\epsilon(n) > 0$ for every $n \in \mathbb{N}$) such that, for every oracle $\mathcal{B}$ that is ϵ-close to $\mathcal{A}$, $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$.*

Proof. For every $n \in \mathbb{N}$, define $X_n := \{f \colon S_n \to \{0,1\}^* \mid \mu_{\mathcal{A}}(f) > 0\}$ to be the set of all functions over S_n that have positive measure in $\mathcal{A}$. We define the distance function as $\epsilon(n) := \min_{f \in X_n} \mu_{\mathcal{A}}(f)$.

By Claim 3.5, from $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$ we deduce that for any $M \in \mathbf{M}_{T,q}$ there exists an input $\mathbb{x}$ and function $f \colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}}(f) > 0$ such that, for every function $F \colon \{0,1\}^* \to \{0,1\}^*$ that agrees with f on $S_{|\mathbb{x}|}$, M fails on input $\mathbb{x}$ and function F for $\mathcal{L}$. Since the oracle $\mathcal{B}$ is ϵ-close to $\mathcal{A}$ we deduce, from the definition of ϵ, that $\mu_{\mathcal{B}}(f) > 0$ as well.

Since the above holds for every $M \in \mathbf{M}_{T,q}$, by Claim 3.5 we conclude that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$. $\square$

Lemma 3.7 (monotonicity)**.** *If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then, for every oracle $\mathcal{B}$ that contains $\mathcal{A}$, it holds that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$.*

Proof. From Claim 3.5, since $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, we know that for every $M \in \mathbf{M}_{T,q}$ there exists an input $\mathbb{x}$ and function $f \colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}}(f) > 0$ such that, for every function $F \colon \{0,1\}^* \to \{0,1\}^*$ that agrees with f on $S_{|\mathbb{x}|}$, M fails on input $\mathbb{x}$ and function F for $\mathcal{L}$. Due to containment (Definition 2.3), since $f \in \mathrm{supp}(\mathcal{A}_{\le s(|\mathbb{x}|)})$, we deduce that

$$\mu_{\mathcal{B}}(f) = \mu_{\mathcal{B}}\left(\mathrm{supp}(\mathcal{A}_{\le s(|\mathbb{x}|)})\right) \cdot \mu_{\mathcal{A}}(f) > 0 \ .$$

Since the above holds for every $M \in \mathbf{M}_{T,q}$, by Claim 3.5 we conclude that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$. $\square$

Lemma 3.8 (conditioning)**.** *Let $I \subseteq \mathbb{N}$ be finite, and set $D := \bigcup_{i \in I} \{0,1\}^i$. If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then there exists a function $g \colon D \to \{0,1\}^*$ such that, letting $\mathcal{L}^g := \{L^g_F\}_{F \in \mathcal{U}}$ where each $L^g_F := L_{F^{D,g}}$, for every $f \colon D \to \{0,1\}^*$ in the support of $\mathcal{A}$ it holds that $\mathcal{L}^g \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,f}}$.*

Proof. Consider the following set of functions over D:

$$G_{D,\mathcal{A}} := \{g \colon D \to \{0,1\}^* \mid \exists A \in \mathrm{supp}(\mathcal{A}) \text{ that agrees with } g \text{ on } D\} \ .$$

First, we argue that, since $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, there exists $g \in G_{D,\mathcal{A}}$ such that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,g}}$. Suppose by way of contradiction that $\mathcal{L} \in \mathrm{PCP}(T,q)^{\mathcal{A}^{D,g}}$ for every $g \in G_{D,\mathcal{A}}$. Then every oracle $\mathcal{A}^{D,g}$ has a PCP-verifier $M_g \in \mathbf{M}_{T,q}$ for $\mathcal{L}$. We use the PCP-verifiers $\{M_g\}_{g \in G_{D,\mathcal{A}}}$ to construct a PCP-verifier M that shows that $\mathcal{L} \in \mathrm{PCP}(T,q)^{\mathcal{A}}$ (a contradiction): $M^{A,\pi}(\mathbb{x})$ first queries all locations in D to identify which $g \in G_{D,\mathcal{A}}$ is consistent with A; then it rules according to $M_g^{A,\pi}(\mathbb{x})$. By construction, the machine M is in $\mathbf{M}_{T,q}$ because querying all locations in D takes a constant amount of time and involves a constant number of queries. (The size of D is a finite constant.)

Next, we use $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,g}}$ to argue that $\mathcal{L}^g \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,f}}$. By definition of $\mathcal{L}^g$, for every $F \in \mathrm{supp}(\mathcal{A}^{D,g})$ we have $L_F = L^g_{F^{D,f}}$. Moreover, since D is the union of binary strings of certain lengths, there is a bijection between functions $F \in \mathrm{supp}(\mathcal{A}^{D,g})$ and functions $F^{D,f} \in \mathrm{supp}(\mathcal{A}^{D,f})$, and $\mu_{\mathcal{A}^{D,g}}(F) = \mu_{\mathcal{A}^{D,f}}(F^{D,f})$. This means that if the oracle $\mathcal{A}^{D,f}$ has a PCP-verifier $M_f \in \mathbf{M}_{T,q}$ for $\mathcal{L}^g$, then we can construct a machine M_g that, relative to the oracle $\mathcal{A}^{D,g}$, is a PCP-verifier for $\mathcal{L}$: $M_g^{A,\pi}(\mathbb{x})$ runs $M_f(\mathbb{x})$ except that M_g answers any query $y \in D$ from M_f with $f(y)$ instead of $A(y)$. One can verify that $M_g \in \mathbf{M}_{T,q}$, which means that $\mathcal{L} \in \mathrm{PCP}(T,q)^{\mathcal{A}^{D,g}}$, a contradiction. $\square$

4 Uniform separation is equivalent to somewhere separation

We prove that uniform separations are equivalent to somewhere separations, both when comparing DTIME and PCP and also when comparing NTIME and PCP. This equivalence relies on structural properties described in Section 3.

Theorem 4.1 (equivalence of uniform and somewhere)**.** *Let* $t, T \colon \mathbb{N} \to \mathbb{N}$ *be time bound functions and* $q \colon \mathbb{N} \to \mathbb{N}$ *a query bound function. For every oracle* $\mathcal{A}$ *the following equivalences hold.*

- $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$ *if and only if* $\exists\, A \in \mathrm{supp}(\mathcal{A})$ *such that* $\mathrm{DTIME}(t)^{A} \not\subseteq \mathrm{PCP}(T,q)^{A}$*.*
- $\mathrm{NTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$ *if and only if* $\exists\, A \in \mathrm{supp}(\mathcal{A})$ *such that* $\mathrm{NTIME}(t)^{A} \not\subseteq \mathrm{PCP}(T,q)^{A}$*.*

The "if directions" are proved in Lemma 4.2, and the "only if directions" are proved in Lemma 4.3.

Lemma 4.2 (somewhere to uniform)**.**
- *If* $\exists\, A \in \mathrm{supp}(\mathcal{A})$ *such that* $\mathrm{DTIME}(t)^{A} \not\subseteq \mathrm{PCP}(T,q)^{A}$*, then* $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$*.*
- *If* $\exists\, A \in \mathrm{supp}(\mathcal{A})$ *such that* $\mathrm{NTIME}(t)^{A} \not\subseteq \mathrm{PCP}(T,q)^{A}$*, then* $\mathrm{NTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$*.*

Proof. We argue the first bullet point. Let L_A be a language in $\mathrm{DTIME}(t)^{A}$ but not in $\mathrm{PCP}(T,q)^{A}$. Let M_A be a deterministic machine that decides L_A, when given oracle access to A, in time $O(t(n))$.

We use the deterministic machine M_A to define an *oracle* language $\mathcal{L} = \{L_F\}_{F \in \mathcal{U}}$ where

$$L_F := \left\{ \mathbb{x} \;\middle|\; M_A^F(\mathbb{x}) = 1 \right\} \ .$$

By definition, $\mathcal{L}$ is decidable by the deterministic machine M_A, and in particular is in $\mathrm{DTIME}(t)^{\mathcal{A}}$. Note that $\mathcal{L}$ is $O(t)$-bounded (see Remark 2.12).

Next, we argue that $\mathcal{L}$ is not in $\mathrm{PCP}(T,q)^{\mathcal{A}}$. Since $\mathcal{L} \notin \mathrm{PCP}(T,q)^{A}$, by Claim 3.5 (invoked for A) we know that for every $M \in \mathbf{M}_{T,q}$ there exists an input $\mathbb{x}$ and function $f \colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_A(f) > 0$ such that, for every function $F \colon \{0,1\}^* \to \{0,1\}^*$ that agrees with f on $S_{|\mathbb{x}|}$, M^F fails to decide the input $\mathbb{x}$ for the language L_F (i.e., $M^F(\mathbb{x}) \neq L_F(\mathbb{x})$). The set of all oracles that agree with f has positive measure in $\mathcal{A}$, because $A \in \mathrm{supp}(\mathcal{A})$ and the finite prefix of a function in $\mathcal{A}$ has positive measure in the distribution. Thus, again relying on Claim 3.5 (this time invoked for $\mathcal{A}$), we conclude that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$.

For the second bullet point, an analogous argument holds if we replace $\mathrm{DTIME}(t)$ with $\mathrm{NTIME}(t)$. The only difference is that the machine is nondeterministic rather than deterministic. □

Lemma 4.3 (uniform to somewhere)**.**
- *If* $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$*, then* $\exists\, A \in \mathrm{supp}(\mathcal{A})$ *such that* $\mathrm{DTIME}(t)^{A} \not\subseteq \mathrm{PCP}(T,q)^{A}$*.*
- *If* $\mathrm{NTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$*, then* $\exists\, A \in \mathrm{supp}(\mathcal{A})$ *such that* $\mathrm{NTIME}(t)^{A} \not\subseteq \mathrm{PCP}(T,q)^{A}$*.*

Proof. We argue the first bullet point. Let $\mathcal{L} = \{L_F\}_{F \in \mathcal{U}}$ be an oracle language that is complete for $\mathrm{DTIME}(t)^{\mathcal{A}}$. We use diagonalization to find a function $A \in \mathrm{supp}(\mathcal{A})$ such that $L_A \notin \mathrm{PCP}(T,q)^{A}$.

Consider an ordering of the machines $\mathbf{M}_{T,q} = \{M_1, M_2, \dots\}$ (this set is countable). We initialize A to be an arbitrary function in $\mathrm{supp}(\mathcal{A})$. At each step i, we modify the function A to agree with some function $f_i \colon D_i \to \{0,1\}^*$, where D_i is the union of binary strings of some bounded length. We prove inductively that after step i, none of the machines $M_1, \dots, M_i$ is a PCP-verifier for L_A.

- *Step $i = 1$.* Since $\mathcal{L} \notin \mathrm{PCP}(T, q)^{\mathcal{A}}$, by Claim 3.5, for M_1 there exist an input $\mathbb{x}_1$ and function $f_1 \colon S_{|\mathbb{x}_1|} \to \{0,1\}^*$ such that, for every function $F \colon \{0,1\}^* \to \{0,1\}^*$ that agrees with f_1 on $S_{|\mathbb{x}_1|}$, M_1^F is not a PCP-verifier for L_F. Define $D_1 := S_{|\mathbb{x}_1|}$, and modify A by setting $A|_{D_1} := f_1$.

  After step 1, we know that M_1^A is not a PCP-verifier for L_A.

- *Step $i = 2$.* The oracle language $\mathcal{L}$ is not in $\mathrm{PCP}(T, q)^{\mathcal{A}^{D_1,f_1}}$ because: (a) $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T, q)^{\mathcal{A}}$ implies, via Theorem 3.1, that $\mathrm{DTIME}(t)^{\mathcal{A}^{D_1,f_1}} \not\subseteq \mathrm{PCP}(T, q)^{\mathcal{A}^{D_1,f_1}}$; and moreover (b) $\mathcal{L}$ is complete for $\mathrm{DTIME}(t)^{\mathcal{A}}$, and thus also for $\mathrm{DTIME}(t)^{\mathcal{A}^{D_1,f_1}}$ (see Claim 4.4 below).

  Next, since $\mathcal{L} \notin \mathrm{PCP}(T, q)^{\mathcal{A}^{D_1,f_1}}$, by Claim 3.5, we deduce that for M_2 there exists an input $\mathbb{x}_2$ and function $f_2 \colon S_{|\mathbb{x}_2|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}^{D_1,f_1}}(f_2) > 0$ such that for every function $F \colon \{0,1\}^* \to \{0,1\}^*$ that agrees with f_2 on $S_{|\mathbb{x}_2|}$, M_2^F is not a PCP-verifier for L_F. Define $D_2 := S_{|\mathbb{x}_2|}$, and modify A by setting $A|_{D_2} := f_2$. By construction, f_1 and f_2 agree on $D_1 \cap D_2 = D_1$, and A agrees with both f_1 and f_2 on $D_1 \cap D_2$.

  After step 2, neither M_1^A nor M_2^A is a PCP-verifier for L_A.

- *General case.* At any step $i > 2$, we repeat the same as above for the oracle $\mathcal{A}^{\cup_{k=1}^{i-1} D_k, \cup_{k=1}^{i-1} f_k}$.

The function A obtained via the above (infinite) procedure is in $\mathrm{supp}(\mathcal{A})$ because, for all $n \in \mathbb{N}$, $A|_{\{0,1\}^n} \in \mathrm{supp}(\mathcal{A}_n)$. By construction of A, no machine in $\mathbf{M}_{T,q}$ is a PCP-verifier for the language L_A, which means that $L_A \notin \mathrm{PCP}(T, q)^A$. Moreover, $L_A \in \mathrm{DTIME}(t)^A$, as we argue next.

Suppose by way of contradiction that $L_A \notin \mathrm{DTIME}(t)^A$, which means that for every $\mathrm{DTIME}(t)$ decider M there exists an input $\mathbb{x}$ such that $M^A(\mathbb{x}) \neq L_A(\mathbb{x})$. Since $\mathcal{L}$ is ℓ-bounded and M runs in time $c \cdot t(n)$ for some $c > 0$, we know that $L_{A'}(\mathbb{x}) \neq M^{A'}(\mathbb{x})$ for every A' that agrees with A on the set $\bigcup_{1 \le i \le \max\{\ell(|\mathbb{x}|), c \cdot t(|\mathbb{x}|)\}} \{0,1\}^i$. This implies (via an argument analogous to the proof of Claim 3.5) that M is not a $\mathrm{DTIME}(t)$ decider of $\mathcal{L}$ for $\mathcal{A}$. Since M was an arbitrary $\mathrm{DTIME}(t)$ decider, we deduce that $\mathcal{L} \notin \mathrm{DTIME}(t)^{\mathcal{A}}$, a contradiction. So it must be that $L_A \in \mathrm{DTIME}(t)^A$.

We conclude that, for the function $A \in \mathrm{supp}(\mathcal{A})$, it holds that $\mathrm{DTIME}(t)^A \not\subseteq \mathrm{PCP}(T, q)^A$.

For the second bullet point, an analogous argument holds if we replace $\mathrm{DTIME}(t)$ with $\mathrm{NTIME}(t)$. The only difference is that machines are nondeterministic rather than deterministic. □

Claim 4.4. *For every finite $D \subseteq \{0,1\}^*$ and function $f \colon D \to \{0,1\}^*$ (such that there exists some $A \in \mathrm{supp}(\mathcal{A})$ agreeing with f on D), if $\mathcal{L}$ is complete for $\mathrm{DTIME}(t)^{\mathcal{A}}$, then $\mathcal{L}$ is complete for $\mathrm{DTIME}(t)^{\mathcal{A}^{D,f}}$. The analogous statement holds for $\mathrm{NTIME}(t)$.*

Proof. First, for every oracle language $\mathcal{L}' = \{L'_F\}_{F \in \mathcal{U}}$ in $\mathrm{DTIME}(t)^{\mathcal{A}^{D,f}}$ there exists an oracle language $\mathcal{L}'' = \{L''_F\}_{F \in \mathcal{U}}$ in $\mathrm{DTIME}(t)^{\mathcal{A}}$ such that for every $A \in \mathcal{A}^{D,f}$ it holds that $L'_A = L''_A$. Indeed, letting M be a decider that puts $\mathcal{L}'$ in $\mathrm{DTIME}(t)^{\mathcal{A}^{D,f}}$, we define $L''_F := \{\mathbb{x} \mid M^F(\mathbb{x}) = 1\}$.

Next, since $\mathcal{L}''$ can be reduced to $\mathcal{L}$ for $A \in \mathcal{A}$, $\mathcal{L}'$ can be reduced to $\mathcal{L}$ for $A \in \mathcal{A}^{D,f}$. Since $\mathcal{L}'$ was arbitrary in $\mathrm{DTIME}(t)^{\mathcal{A}^{D,f}}$, we conclude that $\mathcal{L}$ is also complete for $\mathrm{DTIME}(t)^{\mathcal{A}^{D,f}}$.

An analogous argument holds if we replace $\mathrm{DTIME}(t)$ with $\mathrm{NTIME}(t)$. □

5 Separations for random functions

We define the notion of a random oracle and then state our separation results for it.

Definition 5.1. *A* **random oracle** *with output length* $\ell \colon \mathbb{N} \to \mathbb{N}$ *is the oracle* $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ *where each* $\mathcal{R}_n$ *is the uniform distribution over functions* $f \colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$.

The probability measure of $\mathcal{R}$ is uniform, in the sense that, for any choice of distinct binary strings $x_1,\dots,x_m$ of lengths $n_1,\dots,n_m$ and choice of binary strings $b_1,\dots,b_m$ of lengths $\ell(n_1),\dots,\ell(n_m)$, the set $S := \{A \mid A(x_1) = b_1,\dots,A(x_m) = b_m\}$ has measure $\mu_{\mathcal{R}}(S) = 1/(\Pi_{i=1}^{m} 2^{\ell(n_i)})$.

Theorem 5.2. *Let* $\mathcal{R}$ *be the random oracle with output length* $\ell \colon \mathbb{N} \to \mathbb{N}$.

1. *For any function* $t \colon \mathbb{N} \to \mathbb{N}$,

$$\mathrm{NTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}} \ .$$

2. *For any function* $t \colon \mathbb{N} \to \mathbb{N}$,

$$\mathrm{DTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{R}} \ .$$

Remark 5.3. In Section 8 we prove a stronger (almost-everywhere) separation result. We provide a standalone proof of Theorem 5.2 because the techniques used to prove it can be modified to apply to other oracles. We do not know how to prove stronger separations for other oracles.

5.1 Proof of Part 1 of Theorem 5.2

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{NTIME}(t)^{\mathcal{R}}$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}}$.

Oracle language. Let $u_{k,i}$ denote the $\lceil \log k \rceil$-bit string that represents the index $i \in [k]$. The oracle language $\mathcal{L} = \{L_R\}_{R\in\mathcal{R}}$ is defined as follows:

$$L_R := \left\{ 0^n \;\middle|\; \exists\, w \in \{0,1\}^{t(n)} \text{ s.t. } \begin{array}{c} R(w \| u_{t(n),1})_1 = 0 \\ R(w \| u_{t(n),2})_1 = 0 \\ \vdots \\ R(w \| u_{t(n),t(n)})_1 = 0 \end{array} \right\} \ .$$

Note that L_R does not contain any string that is not all-zeros. Strings of the form 0^n may or may not be in L_R, depending on the answers from R.

In NTIME. We argue that $\mathcal{L}$ is in $\mathrm{NTIME}(t)^{\mathcal{R}}$. Consider the nondeterministic machine that, for inputs of the form 0^n, expects as nondeterministic witness a string $w \in \{0,1\}^{t(n)}$ and checks, via $t(n)$ calls to R, if R returns a string whose first bit is zero on input $w \| u_{t(n),i}$ for every $i \in \{1,\dots,t(n)\}$. The machine rejects any input not of the form 0^n. This machine, for any given R, decides the language L_R on every input. The machine's running time is $O(t(n))$: writing the first query costs $O(t(n))$ steps and updating the query tape with each new subsequent query costs $O(1)$ steps.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}}$. Suppose by way of contradiction that $\mathcal{L}$ has a PCP-verifier $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$, and denote by $T(n)$ the running time of M on input of size n. For every $n \in \mathbb{N}$, define the domain $S_n := \bigcup_{1 \le i \le T(n)} \{0,1\}^i$ and the following set X_n of functions over S_n:

$$X_n = \left\{ f \colon S_n \to \{0,1\}^* \;\middle|\; \begin{array}{l} \exists !\, w \in \{0,1\}^{t(n)} \text{ such that both conditions below hold} \\ \bullet \forall i \in [t], f(x \| u_{t,i}) = 01^{\ell(t+\lceil \log t \rceil)-1} \\ \bullet \forall y \notin \{w \| u_{t,j}\}_{j \in [t]}, f(y) = 1^{\ell(|y|)} \end{array} \right\} .$$

Note that, for every $n \in \mathbb{N}$, every function $f \in X_n$ has measure $\mu_{\mathcal{R}}(\{f\}) > 0$.

We also use $\mathbf{1}$ to denote the all-one function that on an input $z \in \{0,1\}^*$ returns $1^{\ell(|z|)}$.

We derive a contradiction from the following two (contradicting) statements.

- Lemma 5.4: For every $n \in \mathbb{N}$ and function R agreeing with some $f \in X_n$ there exists a proof π such that $M^{\mathbf{1},\pi}(0^n)$ queries $\mathbf{1}$ at some "witness location" $w \| u_{t(n),i}$ of R with probability at least $\frac{1}{3}$.
- Lemma 5.5: For every $n \in \mathbb{N}$ there exists a function R agreeing with some $f \in X_n$ such that for any proof π it holds that $M^{\mathbf{1},\pi}(0^n)$ queries $\mathbf{1}$ at some "witness location" $w \| u_{t(n),i}$ of R with probability only $o(1)$.

We are left to prove the lemmas. We abbreviate $t(n)$ with t as the choice of n is clear from context.

Lemma 5.4. *If $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ is a PCP-verifier for $\mathcal{L}$, then for every $n \in \mathbb{N}$ and function R agreeing with some $f \in X_n$ there exists a proof π s.t.*

$$\Pr_r \left[M^{\mathbf{1},\pi}(0^n; r) \text{ queries } \mathbf{1} \text{ at some } w \| u_{t,i} \text{ s.t. } R(w \| u_{t,i})_1 = 0 \right] > \frac{1}{3} .$$

Proof. By definition of X_n, $0^n \in L_R$ for any R agreeing with some $f \in X_n$. Therefore, for every such R, there exists a proof π such that $M^{R,\pi}(0^n)$ accepts with probability at least $\frac{2}{3}$. We also note that since $M^{R,\pi}(0^n)$ has running time $T(n)$, it cannot make oracle queries outside the set S_n.

We now argue that for every R agreeing with some $f \in X_n$ there exists a proof π such that

$$\Pr_r \left[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at some } w \| u_{t,i} \text{ s.t. } R(w \| u_{t,i}) = 0 \right] > \frac{1}{3} . \tag{1}$$

Suppose Equation (1) does not hold. Then more than $\frac{2}{3}$ of the time $M^{R,\pi}(0^n; r)$ does not query the witness bits. If we change R slightly by flipping the first bit of $R(w \| u_{t,i})$ from 0 to 1, and denote the new oracle by R_i, then $0^n \notin L_{R_i}$. However, the machine $M^{R_i,\pi}(0^n; r)$ cannot detect the change in more than $\frac{2}{3}$ of the time. So $M^{R_i,\pi}(0^n; r)$ accepts with probability at least $\frac{1}{3}$. Moreover, $M^{R'_i,\pi}(0^n; r)$ makes the same mistake for any function R'_i that agrees with R_i on S_n. This conclusion contradicts the fact that M verifies $L_{R'_i}$ on 0^n for all such functions R'_i. So Equation (1) holds.

Furthermore, if $w \| u_{t,i}$ is the first query made by $M^{R,\pi}(0^n; r)$ such that $R(w \| u_{t,i}) = 01^{\ell(t+\lceil \log t \rceil)-1}$, then $M^{\mathbf{1},\pi}(0^n; r)$ would also make the query $w \| u_{t,i}$. This is because M has the same view in the two cases at the time it makes the query $w \| u_{t,i}$. The lemma follows. □

Lemma 5.5. *If $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ is a PCP-verifier for $\mathcal{L}$, then for every $n \in \mathbb{N}$ there exists R agreeing with some $f \in X_n$ such that for every proof $\pi \in \{0,1\}^*$,*

$$\Pr_r \left[M^{\mathbf{1},\pi}(0^n; r) \text{ queries } \mathbf{1} \text{ at some } w \| u_{t,i} \text{ s.t. } R(w \| u_{t,i})_1 = 0 \right] \in o(1) .$$

Proof. For the sake of contradiction, suppose there exists some $n \in \mathbb{N}$ such that for every R agreeing with some $f \in X_n$ there exists a proof π for which the above probability is $\Omega(1)$. Then, by averaging, there exists some randomness r^* such that, for a $\Omega(1)$-fraction of the functions R agreeing with some $f \in X_n$, there exists a proof π such that $M^{\mathbf{1},\pi}(0^n; r^*)$ queries some $w \| u_{t,i}$ s.t. $R(w\|u_{t,i})_1 = 0$. So across all possible proofs, $M^{\mathbf{1},\cdot}(0^n; r^*)$ need to make at least $\Omega(|X_n|) = \Omega(2^t)$ distinct queries.

However, since the randomness is fixed and M is in $\mathbf{M}_{\mathrm{poly}(t),o(t)}$, $M^{\mathbf{1},\cdot}(0^n; r^*)$ can only make at most $2^{o(t)} \cdot \mathrm{poly}(t)$ distinct queries, which leads to a contradiction because $\Omega(2^t) \gg 2^{o(t)} \cdot \mathrm{poly}(t)$. $\square$

5.2 Proof of Part 2 of Theorem 5.2

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{DTIME}(t)^{\mathcal{R}}$ but not in $\mathrm{PCP}(o(t), o(t))^{\mathcal{R}}$.

Oracle language. Let $u_{k,i}$ denote the $\lceil \log k \rceil$-bit string that represents the index $i \in [k]$. For convenience we use $t^*(n)$ to denote $t(n)/n$. The language $\mathcal{L} = \{L_R\}_{R \in \mathcal{R}}$ is then defined as follows:

$$L_R := \{(x, y) \in \{0,1\}^n \times \{0,1\}^n \mid F_{R,n}(x) = y\} \quad ,$$

where $F_{R,n}(x)$ is the n-bit string whose i-th bit is $\bigoplus_{j \in \{(i-1)\cdot t^*(n)+1, \dots, i \cdot t^*(n)\}} R(x \| u_{t(n),j})_1$, for $i \in [n]$.

In DTIME. We argue that the language L_R is in $\mathrm{DTIME}(t)^R$ for every $R \in \mathcal{R}$. Consider the deterministic machine that on input (x, y): (a) copies x to the query tape; (b) for every $i \in [n]$, calls R on inputs $\{x \| u_{t(n),j}\}_{j \in \{(i-1)\cdot t^*(n)+1, \dots, i \cdot t^*(n)\}}$ to get $z_i := F_{R,n}(x)_i$; (c) accepts if $y = z$. Copying x takes time $O(n)$, querying all bits of the form $x \| u_{t(n),j}$ takes $O(t(n))$ time, computing z takes time $O(t(n))$, and comparing z and y takes time $O(n)$. So the running time of the machine is $O(t(n))$.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(o(t), o(t))^{\mathcal{R}}$.

- Consider any $R \in \mathcal{R}$ and input $x \in \{0,1\}^n$. We know that by definition $(x, F_{R,n}(x)) \in L_R$.
- Let $e_{k,j}$ denote the k-bit string that has 1 in the j-th coordinate and 0 eeverywhere else. For every $x \in \{0,1\}^n$ and $j \in \{1, \dots, t(n)\}$, define the oracle $R^{(x,j)}$ to be

  $$R^{(x,j)}(z) := \begin{cases} R(z) \oplus e_{|z|,1} & \text{if } z = x \| u_{t(n),j} \\ R(z) & \text{otherwise} \end{cases} \quad .$$

  Therefore for every $x \in \{0,1\}^n$ and $j \in \{1, \dots, t(n)\}$ $(x, F_{R,n}(x)) \notin L_{R^{(x,j)}}$.

We argue that any PCP-verifier M for $\mathcal{L}$ must have running time $\Omega(t(n))$.

- Since M is a PCP-verifier for $\mathcal{L}$, for every $x \in \{0,1\}^n$ there exists a proof π such that $M^{R,\pi}(x, F_{R,n}(x))$ accepts with probability at least $\frac{2}{3}$. Then, via Lemma 5.6 below, we deduce that there exists a randomness r such that $M^{R,\pi}(x, F_{R,n}(x); r)$ makes at least $\frac{t(n)}{3}$ queries of the form $x \| u_{t(n),j}$ for some $j \in \{1, \dots, t(n)\}$.
- We conclude that the running time of $M^{R,\pi}(x, F_{R,n}(x); r)$ is at least $\frac{t(n)}{3} \in \Omega(t(n))$.

We have shown that any PCP-verifier for $\mathcal{L}$ has running time in $\Omega(t(n))$, and so $\mathcal{L} \notin \mathrm{PCP}(o(t), o(t))^{\mathcal{R}}$.

Lemma 5.6. *If M is a PCP-verifier for $\mathcal{L}$ then, for every oracle $R \in \mathcal{R}$, $n \in \mathbb{N}$, and $x \in \{0,1\}^n$, there exists a proof π and randomness r such that $M^{R,\pi}(x, F_{R,n}(x); r)$ makes at least $\frac{t(n)}{3}$ queries of the form $x \| u_{t(n),j}$ where $j \in \{1, \dots, t(n)\}$.*

Proof. Since M is a PCP-verifier for $\mathcal{L}$ so for every $R \in \mathcal{R}$, $n \in \mathbb{N}$, and $x \in \{0,1\}^n$ there exists a proof π such that $M^{R,\pi}(x, F_{R,n}(x))$ accepts with probability at least $\frac{2}{3}$. Additionally we know that for any $j \in \{1, \ldots, t(n)\}$ $(x, F_{R,n}(x)) \notin L_{R^{(x,j)}}$. So $M^{R^{(x,j)},\pi}(x, F_{R,n}(x))$ should accept with probability at most $\frac{1}{3}$. This implies that for at least $\frac{1}{3}$ fraction of randomness r, $M^{R,\pi}(x, F_{R,n}(x); r)$ queries $x \| u_{t(n),j}$. By averaging over all randomness r, there exists r^* such that $M^{R,\pi}(x, F_{R,n}(x); r^*)$ makes $\frac{t(n)}{3}$ distinct queries of the form $x \| u_{t(n),j}$ where $j \in \{1, \ldots, t(n)\}$. □

6 Separation for random low-degree functions

We define the notion of a random *low-degree* oracle and then state our separation result for it.

Definition 6.1. *Let q be a prime power, $\mathbb{F}_q$ the finite field of size q, and $d \in \mathbb{N}$ a degree bound. The* **random oracle over $\mathbb{F}_q$ with degree** *d is the oracle $\mathcal{P}[q,d] = \{\mathcal{P}_n\}_{n \in \mathbb{N}}$ where each $\mathcal{P}_n$ is the uniform distribution over polynomials $f \colon \mathbb{F}_q^n \to \mathbb{F}_q$ of degree at most d in each variable. (In particular, $\mathcal{P}[2,1]$ equals the random oracle $\mathcal{R}$ from Definition 5.1.)*

Theorem 6.2. *Let $q \in \mathbb{N}$ be a prime power and $d \in \mathbb{N}$ a degree bound.*

1. For any function $t \colon \mathbb{N} \to \mathbb{N}$,

$$\mathrm{NTIME}(t)^{\mathcal{P}[q,d]} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}[q,d]} .$$

2. For any function $t \colon \mathbb{N} \to \mathbb{N}$,

$$\mathrm{DTIME}(t)^{\mathcal{P}[q,d]} \not\subseteq \mathrm{PCP}(o(t^{\log_2 \frac{q}{q-1}}), o(t^{\log_2 \frac{q}{q-1}}))^{\mathcal{P}[q,d]} .$$

Proof. We first focus on the special case of $d = 1$, i.e., the case of random multilinear polynomials. We show in Lemma 6.3 that $\mathcal{P}[q,1]$ separates NTIME/DTIME and PCP. Next, we simply observe that for any prime power q and degree d, the oracle $\mathcal{P}[q,d]$ contains the multilinear oracle $\mathcal{P}[q,1]$. Therefore, by Theorems 3.1 and 3.2, $\mathcal{P}[q,d]$ also separates NTIME/DTIME and PCP. $\square$

Lemma 6.3. *Let $q \in \mathbb{N}$ be a prime power.*

1. For any function $t \colon \mathbb{N} \to \mathbb{N}$,

$$\mathrm{NTIME}(t)^{\mathcal{P}[q,1]} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}[q,1]} .$$

2. For any function $t \colon \mathbb{N} \to \mathbb{N}$,

$$\mathrm{DTIME}(t)^{\mathcal{P}[q,1]} \not\subseteq \mathrm{PCP}(o(t^{\log_2 \frac{q}{q-1}}), o(t^{\log_2 \frac{q}{q-1}}))^{\mathcal{P}[q,1]} .$$

6.1 Proof of Part 1 of Lemma 6.3

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{NTIME}(t)^{\mathcal{P}[q,1]}$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}[q,1]}$.

Oracle language. Let $e_{k,i}$ denote the vector in $\mathbb{F}_q^k$ that has 1 in the i-th coordinate and 0 everywhere else. Let $\mathcal{L} = \{L_P\}_{P \in \mathcal{P}[q,1]}$ be the oracle language where

$$L_P := \left\{ 0^n \;\middle|\; \exists\, w \in \mathbb{F}_q^{t(n)} \text{ s.t. } P(w) = 1 \text{ and } \begin{array}{c} P(w + e_{t(n),1}) = 0 \\ P(w + e_{t(n),2}) = 0 \\ \vdots \\ P(w + e_{t(n),t(n)}) = 0 \end{array} \right\} .$$

In NTIME. We argue that $\mathcal{L}$ is in $\mathrm{NTIME}(t)^{\mathcal{P}[q,1]}$. Consider the nondeterministic machine that, for inputs of the form 0^n, expects as nondeterministic witness a string $w \in \{0,1\}^{t(n)}$ and checks, via $t(n)+1$ calls to P, if P returns 1 on input w and zero on input $w + e_{t(n),i}$ for every $i \in [t(n)]$.

The machine rejects any input not of the form 0^n. This machine, for any given P, decides the language L_P on every input. The machine's running time is $O(t(n))$: writing the first query costs time $O(t(n))$ and updating the query tape with each new subsequent query costs time $O(1)$.

Not in PCP. Suppose by way of contradiction that $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ is a PCP-verifier for $\mathcal{L}$. Use $T(n)$ to denote the running time of M on input of size n. For every $n \in \mathbb{N}$, we define the following set of functions over the domain $S_{T(n)} = \{\mathbb{F}_q^i \mid i \leq T(n)\}$.

$$X_{T(n)} = \left\{ f\colon S_{T(n)} \to \mathbb{F}_q \;\middle|\; \begin{array}{l} \forall x \in \mathbb{F}_q^{t(n)},\ f(x) = \prod_{i=1}^{t(n)} (b_i - x_i) \text{ where } b \in \mathbb{F}_q^{t(n)} \\ \forall x \in S_{T(n)} \setminus \mathbb{F}_q^{t(n)},\ f(x) = 0 \end{array} \right\} .$$

Note for every $n \in \mathbb{N}$, every function $f \in X_{T(n)}$ has its measure $\mu_{\mathcal{P}[q,1]}(\{f\}) > 0$.

We also use $\mathbf{0}$ to denote the all zero function.

We derive a contradiction from the following two steps. First in Claim 6.4, we show that for every $n \in \mathbb{N}$ and oracle polynomial P agreeing with some $f \in X_{T(n)}$ there exists some proof π such that the PCP-verifier $M^{\mathbf{0},\pi}$ queries $\mathbf{0}$ at some x satisfying $P(x) \neq 0$ with probability at least $\frac{1}{3}$. Next, in Claim 6.5, we show that there exists some oracle polynomial P agreeing with some $f \in X_{T(n)}$ such that for any proof $\pi \in \{0,1\}^*$, the PCP-verifier $M^{\mathbf{0},\pi}$ queries P at some x satisfying $P(x) \neq 0$ with probability only $o(1)$. These two statements are in contradiction. Thus $\mathcal{L}$ does not have a PCP-verifier. Therefore $\mathcal{P}[q,1]$ separates NTIME and PCP.

In the proofs of the two claims we abbreviate $t(n)$ with t whenever the choice of n is clear from the context.

Claim 6.4. *If M is a PCP-verifier for $\mathcal{L}$ with running time $T(n)$, then for every $n \in \mathbb{N}$ and oracle polynomial P agreeing with some $f \in X_{T(n)}$, there exists π s.t.*

$$\Pr_r[M^{\mathbf{0},\pi}(0^n; r) \textit{ queries } \mathbf{0} \textit{ at some } y \in \mathbb{F}_q^{t(n)} \textit{ s.t. } P(y) \neq 0] > \frac{1}{3} .$$

Proof. We first observe that for every function $f(x) = \prod_{i=1}^{t(n)} (b_i - x_i)$ in $X_{T(n)}$, the element $y = (b_1 - 1) \| \ldots \| (b_{t(n)} - 1) \in \mathbb{F}_q^{t(n)}$ satisfies $f(y) = 1$ and $f(y + e_{t(n),i}) = 0$ for every $i \in [t(n)]$. Therefore for every P agreeing with f over $\mathbb{F}_q^{t(n)}$, $0^n \in L_P$. As a result, for every such P, there exists some proof $\pi \in \{0,1\}^*$ such that $M^{P,\pi}(0^n)$ accepts with probability at least $\frac{2}{3}$. We also note that since $M^{P,\pi}(0^n)$ has running time $T(n)$, M cannot make oracle queries outside the set $S_{T(n)}$.

For every P agreeing with some $f \in X_{T(n)}$, use π_P to denote the accepting proof for P. We additionally note that for every oracle P' agreeing with $\mathbf{0}$ over $S_{T(n)}$, it holds that $0^n \notin L_{P'}$. So for any π_P, $M^{P',\pi_P}(0^n)$ accepts with probability at most $\frac{1}{3}$.

This implies that

$$\Pr_r[M^{P,\pi_P}(0^n; r) \text{ queries } P \text{ at } x \text{ s.t. } P(x) \neq \mathbf{0}(x) = 0] \geq \frac{1}{3} . \tag{2}$$

Let y be the first oracle query made by $M^{P,\pi_P}(0^n; r)$ such that $P(y) \neq \mathbf{0}(y) = 0$. If we replace P with $\mathbf{0}$, $M^{\mathbf{0},\pi_P}(0^n; r)$ would still make the oracle query y. We deduce that

$$\Pr_r[M^{\mathbf{0},\pi_P}(0^n; r) \text{ queries } \mathbf{0} \text{ at } x \text{ s.t. } P(x) \neq \mathbf{0}(x) = 0] \geq \frac{1}{3} .$$

□

Claim 6.5. *If $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ is a PCP-verifier for $\mathcal{L}$ with running time $T(n)$, there exists P agreeing with some $f \in X_{T(n)}$ s.t. for all $\pi \in \{0,1\}^*$,*

$$\Pr_r[M^{\mathbf{0},\pi}(0^n;r) \text{ queries } \mathbf{0} \text{ at } x \text{ s.t. } P(x) \neq 0] \in o(1) \quad .$$

Proof. Suppose for every P agreeing with some $f \in X_{T(n)}$ there exists a proof π that the aforementioned probability is $\Omega(1)$. Then, by an averaging argument, there exists some randomness r^* such that for $\Omega(1)$ fraction of oracles P agreeing with some $f \in X_n$, there exists π s.t. $M^{\mathbf{0},\pi}(0^n;r^*)$ queries some x s.t. $P(x) \neq 0$. Additionally, for any $x \in \mathbb{F}_q^{t(n)}$, there are exactly $(q-1)^{t(n)}$ multilinear functions $f \in X_n$ such that $f(x) \neq 0$. So across all possible proofs, $M^{\mathbf{0},\cdot}(0^n;r^*)$ need to make at least $\Omega(|X_n|/(q-1)^t) = \Omega((q/(q-1))^t)$ distinct queries.

However, since the randomness is fixed to be r^* and M is in $\mathbf{M}_{\mathrm{poly}(t),o(t)}$, $M^{\mathbf{0},\cdot}(0^n;r^*)$ can make at most $2^{o(t)} \cdot \mathrm{poly}(t)$ distinct queries. However, $\Omega((q/(q-1))^t) \gg 2^{o(t)} \cdot \mathrm{poly}(t)$, so we derive a contradiction. ☐

6.2 Proof of Part 2 of Lemma 6.3

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{DTIME}(t)^{\mathcal{P}[q,1]}$ but not in $\mathrm{PCP}(o(t^{\log_2 \frac{q}{q-1}}), o(t^{\log_2 \frac{q}{q-1}}))^{\mathcal{P}[q,1]}$.

Oracle language. Let $\mathcal{L} = \{L_P\}_{P \in \mathcal{P}[q,1]}$ be the oracle language where

$$L_P := \left\{ 0^n \;\middle|\; \forall\, y \in \mathbb{F}_q^{\log_2 t(n)},\ P(y) = P(y\|0) \right\} \quad .$$

In DTIME. We argue that $\mathcal{L}$ is in $\mathrm{DTIME}(t)^{\mathcal{P}[q,1]}$. Consider the deterministic machine that on input 0^n: (a) for each $y \in \{0,1\}^{\log_2 t(n)}$, queries the oracle P at y and $y\|0$, and reject if $P(y) \neq P(y\|0)$ (here 0 and 1 are elements in $\mathbb{F}_q$); (b) in the end accepts if has not rejected. Making all the oracle queries takes amortized $O_q(t(n))$ time. So the running time of the machine is in $O_q(t(n))$.

Not in PCP. We argue that any PCP-verifier for $\mathcal{L}$ must have running time $\Omega(t(n)^{\log_2 \frac{q}{q-1}})$, and thus prove that $\mathcal{L}$ is not in $\mathrm{PCP}(o(t^{\log_2 \frac{q}{q-1}}), o(t^{\log_2 \frac{q}{q-1}}))^{\mathcal{P}[q,1]}$. Let M be a PCP-verifer for $\mathcal{L}$.

- For every $n \in \mathbb{N}$, we define the following set of functions.

$$X_n = \left\{ f\colon \mathbb{F}_q^* \to \mathbb{F}_q \;\middle|\; \begin{array}{l} \forall x \in \mathbb{F}_q^{\log_2 t(n)},\ f(x) = \prod_{i=1}^{\log_2 t(n)} (b_i - x_i) \text{ where } b \in \mathbb{F}_q^{\log_2 t(n)} \\ \forall z \notin \mathbb{F}_q^{\log_2 t(n)},\ f(z) = 0 \end{array} \right\} \quad .$$

  For every P such that $0^n \in L_P$, consider the modified oracle $P^{(f)} := P + f$ for some $f \in X_n$. Then $0^n \notin L_{P^{(f)}}$ because there exists $x \in \mathbb{F}_q^{\log_2 t(n)}$ such that $P^{(f)}(x) = P(x) + f(x) \neq P(x\|0) = P^{(f)}(x\|0)$.

- Since M is a PCP-verifier for $\mathcal{L}$, for every $n \in \mathbb{N}$ and every P such that $0^n \in L_P$ there exists some proof π such that $M^{P,\pi}(0^n)$ accepts with probability at least $\frac{2}{3}$. Then, via Lemma 6.6 below, we deduce that for every n there exists a randomness r such that $M^{P,\pi}(0^n)$ queries P at at least $\Omega((q/(q-1))^{\log_2 t(n)})$ different $y \in \mathbb{F}_q^{\log_2 t(n)}$.

- Writing down $\Omega((q/(q-1))^{\log_2 t(n)})$ distinct $y \in \mathbb{F}_q^{\log_2 t(n)}$ takes amortized $\Omega_q((q/(q-1))^{\log_2 t(n)})$ time. So the running time of $M^{P,\pi}(0^n;r)$ is at least $\Omega_q((q/(q-1))^{\log_2 t(n)}) = \Omega_q\left(t(n)^{\log_2(q/(q-1))}\right)$.

Lemma 6.6. *If M is a PCP-verifier for $\mathcal{L}$ then, for every large enough $n \in \mathbb{N}$ and every P such that $0^n \in L_P$, there exists a proof π and randomness r such that $M^{P,\pi}(0^n; r)$ queries P at at least $\Omega((q/(q-1))^{\log_2 t(n)})$ different $y \in \mathbb{F}_q^{\log_2 t(n)}$.*

Proof. Since M is a PCP-verifier for $\mathcal{L}$, for every such P satisfying $0^n \in L_P$, there exists a proof π such that $M^{P,\pi}(0^n)$ accepts with probability at least $\frac{2}{3}$. If we change P to some $P^{(f)}$ where $f \in X_n$, then $0^n \notin L_{P^{(f)}}$. So $M^{P^{(f)},\pi}(0^n)$ should accept with probability at most $\frac{1}{3}$. This implies that for at least $\frac{1}{3}$ fraction of randomness r, $M^{P,\pi}(0^n; r)$ queries P at some y such that $P^{(f)}(y) \neq P(y)$ (in other words $f(y) \neq 0$).

By considering all $f \in X_n$ and averaging over all randomness r, there exists r^* such that for $\frac{1}{3}$ fraction of $f \in X_n$ $M^{P,\pi}(0^n; r^*)$ queries P at some y such that $f(y) \neq 0$. Additionally, for any $y \in \mathbb{F}_q^{\log_2 t(n)}$, there are exactly $(q-1)^{\log_2 t(n)}$ functions $f \in X_n$ such that $f(y) \neq 0$. So $M^{P,\pi}(0^n; r^*)$ need to make at least $\frac{1}{3}|X_n|/(q-1)^{\log_2 t(n)} = \Omega((q/(q-1))^{\log_2 t(n)})$ different queries in $\mathbb{F}_q^{\log_2 t(n)}$. □

7 Separation for random groups

The notion of a random group oracle is modeled after the generic group model [Nec94, Sho97, BL96, MW98, Mau05]. We provide definitions, and then state our separation result for random groups.

Definition 7.1 (groups and their representations). *An abelian group of order p is a pair $\mathbb{G} = (S, +)$ where S is a set of size p and $+ \colon S \times S \to S$ is a function that satisfies the axioms of a group operation. We denote by $\mathbf{0}$ the identity of $\mathbb{G}$. A representation of $\mathbb{G}$ is an injective function $\sigma \colon S \to \{0,1\}^{\lceil \log_2 p \rceil}$, and its inverse $\sigma^{-1} \colon \{0,1\}^{\lceil \log_2 p \rceil} \to S$ maps each image $\sigma(g) \in \{0,1\}^{\lceil \log_2 p \rceil}$ to its pre-image $g \in S$ and each string $s \in \{0,1\}^{\lceil \log_2 p \rceil} \setminus \sigma(S)$ to the identity $\mathbf{0} \in S$.*

Definition 7.2 (group oracles). *Let $\mathbb{G}$ be a group of order p, and σ a representation of $\mathbb{G}$. The group oracle corresponding to $(\mathbb{G}, \sigma)$ is the function $O \colon \{0,1\}^{4\lceil \log_2 p \rceil} \to \{0,1\}^{\lceil \log_2 p \rceil}$ such that $O(\gamma, \delta, a, b) = \sigma(\gamma \cdot \sigma^{-1}(a) + \delta \cdot \sigma^{-1}(b))$. Note that $O(0^{\lceil \log_2 p \rceil}, 0^{\lceil \log_2 p \rceil}, a, b) = \sigma(\mathbf{0})$.*

Definition 7.3. *The* **random (generic) group oracle** *is the oracle $\mathcal{O} = \{\mathcal{O}_p\}_{p \in \mathsf{PRIMES}}$ where each $\mathcal{O}_p$ is the uniform distribution over all group oracles for groups of size p. Namely, a sample from $\mathcal{O}_p$ is obtained as follows: sample a random representation σ_p of $\mathbb{G}_p$ (the group of prime order p), and output the group oracle $O \colon \{0,1\}^{4\lceil \log_2 p \rceil} \to \{0,1\}^{\lceil \log_2 p \rceil}$ corresponding to $(\mathbb{G}_p, \sigma_p)$.*[2]

Theorem 7.4. *Let $\mathcal{O}$ be the random group oracle.*

1. *For any function $t \colon \mathbb{N} \to \mathbb{N}$,*
$$\mathrm{NTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}} \enspace .$$

2. *For any function $t \colon \mathbb{N} \to \mathbb{N}$,*
$$\mathrm{DTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{O}} \enspace .$$

7.1 Proof of Part 1 of Theorem 7.4

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{NTIME}(t)^{\mathcal{O}}$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}}$.

Oracle language. Let $p(n)$ be the largest prime number smaller than $2^{t(n)}$, so we know that $p(n) \in [2^{t(n)-1}, 2^{t(n)})$. Let $e_{k,i}$ denote the k-bit string that has 1 in the i-th entry and 0 everywhere else. Define the set $S_n := \{x \in \{0,1\}^{t(n)} \,|\, 0 \le x < p(n) - 1\}$. Let $\mathcal{L} = \{L_O\}_{O \in \mathcal{O}}$ be the oracle language where

$$L_O := \left\{ 0^n \;\middle|\; \exists\, w \in S_n \text{ s.t. } \sigma_{p(n)}^{-1}(w) + \sigma_{p(n)}^{-1}(w \oplus e_{t(n),t(n)}) = \mathbf{0} \right\} \enspace .$$

In NTIME. We argue that $\mathcal{L}$ is in $\mathrm{NTIME}(t)^{\mathcal{O}}$. Consider the nondeterministic machine that, for inputs of the form 0^n, expects as nondeterministic witness a string $w \in S_n$ and checks, via $O(1)$ calls

[2]The definition of an oracle given here is convenient for this section but does not syntactically match the one in Definition 2.1. This is an unimportant technicality because we are implicitly using the oracle $\mathcal{O}^* = \{\mathcal{O}^*_{5n}\}_{n \in \mathbb{N}}$ defined as follows: $\mathcal{O}^*_n$ is the distribution over functions $O^* \colon \{0,1\}^{5n} \to \{0,1\}^n$ such that, for every $p \in \{0, \dots, 2^n - 1\} \cong \{0,1\}^n$, if p is a prime in $[2^{n-1}, 2^n)$ then $O^*(p, \gamma, \delta, a, b)$ is distributed identically to $O_p(\gamma, \delta, a, b)$ for a sample $O_p \leftarrow \mathcal{O}_p$, and otherwise ($p$ is not a prime or is not in $[2^{n-1}, 2^n)$) then $O^*(p, \gamma, \delta, a, b)$ always equals 0^n.

to O, if $\sigma_{p(n)}^{-1}(w) + \sigma_{p(n)}^{-1}(w \oplus e_{t(n),t(n)}) = \mathbf{0}$. This machine, for any given O, decides the language L_O on every input. The running time is $O(t(n))$.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}}$. Consider an oracle O such that $0^n \notin L_O$. We show that, for any PCP-verifier $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$, if M^O succeeds on 0^n then we can construct another oracle O' for which $M^{O'}$ fails on 0^n (see Lemma 7.5 below). Therefore there exists some $\tilde{O}$ for which $M^{\tilde{O}}$ fails on 0^n. Additionally, the oracle language $\mathcal{L}$ is by definition $O(t)$-bounded and the running time of M is $T(n) \in \mathrm{poly}(t(n))$. Since $M^{\tilde{O}}$ fails on 0^n, for every function F that agrees with $\tilde{O}$ on $\bigcup_{1\le i\le T(n)}\{0,1\}^i$, M^F also fails on 0^n. So by Claim 3.5 we conclude that $\mathcal{L} \notin \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}}$. We are left to prove the lemma below.

Lemma 7.5. *For every PCP-verifier $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ and function $O \in \mathcal{O}$ such that $0^n \notin L_O$, if M^O succeeds on 0^n then there exists a function $O' \in \mathcal{O}$ for which $M^{O'}$ fails on 0^n.*

Proof. Define the set of pairs of strings

$$I(O) := \{(u,v) \in S_n \times S_n \mid \sigma_{p(n)}^{-1}(u) + \sigma_{p(n)}^{-1}(v \oplus e_{t(n),t(n)}) = \mathbf{0}\} \ .$$

Note that if we use $O^{(u,v)}$ to denote the oracle identical to O except that its permutation function for the group of order $p(n)$ is the following one:

$$\sigma_{p(n)}^{(u,v)}(g) := \begin{cases} u & \text{if } \sigma_{p(n)}(g) = v \\ v & \text{if } \sigma_{p(n)}(g) = u \\ \sigma_{p(n)}(g) & \text{otherwise} \end{cases} \ .$$

For every $(u,v) \in I(O)$, we have $(\sigma_{p(n)}^{(u,v)})^{-1}(v) + (\sigma_{p(n)}^{(u,v)})^{-1}(v \oplus e_{t(n),t(n)}) = \mathbf{0}$. So $0^n \in L_{O^{(u,v)}}$.

We say that a machine queries $O \in \mathcal{O}$ at some string $a \in \{0,1\}^{t(n)}$ if the machine queries $O_{p(n)}(\gamma,\delta,a,b)$ or $O_{p(n)}(\gamma,\delta,b,a)$ for some integers γ,δ and $b \in \{0,1\}^{t(n)}$.

Consider the pairs of strings $(u,v) \in I(O)$ for which M^O queries all of (u,v) with "low" probability:

$$Q_\star(O) := \left\{ (u,v) \in I(O) \;\middle|\; \text{for } a \in \{u,v\},\ \sum_{\pi\in\{0,1\}^*} \Pr_r\left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } a \\ \text{or gets back } a \text{ from } O \end{array}\right] < \frac{1}{6} \right\} \ .$$

We argue that $|Q_\star(O)| > 0$, and for every $(u,v) \in Q_\star(O)$ it holds that $M^{O^{(u,v)}}$ fails on 0^n.

Consider the set of strings $a \in S_n$ that M^O queries with "high" probability:

$$Q_c(O) := \left\{ a \in S_n \;\middle|\; \sum_{\pi\in\{0,1\}^*} \Pr_r\left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } a \\ \text{or gets back } a \text{ from } O \end{array}\right] \ge \frac{1}{6} \right\} \ .$$

Observe that

$$|Q_c(O)| \le \Big(c \cdot t(n)\Big) \cdot \Big(2^{o(t(n))} \cdot \mathrm{poly}(t(n))\Big) = 2^{o(t(n))} \ . \tag{3}$$

This is because, in any given execution, M can make at most $\mathrm{poly}(t(n))$ queries to O (also can get at most $\mathrm{poly}(t(n))$ symbols from O) and $o(t(n))$ queries to the given proof string, which means that

$$\sum_{a\in S_n} \sum_{\pi\in\{0,1\}^*} \Pr_r\left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } a \\ \text{or gets back } a \text{ from } O \end{array}\right] \le 2^{o(t(n))}\mathrm{poly}(t(n)) \ .$$

We deduce that $|Q_\star(O)|$ is large:

$$|Q_\star(O)| \geq (p(n)-3) - 2\cdot 2|Q_c(O)| = 2^{t(n)} - 2^{o(t(n))} \ .$$

Next, for every $(u,v) \in Q_\star(O)$, it holds that

$$\forall\, \pi \in \{0,1\}^*, \text{ for } a \in \{u,v\}, \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n; r) \text{ queries } O \text{ at } a \\ \text{or gets back } a \text{ from } O \end{array} \right] < \frac{1}{6} \ .$$

Therefore,

$$\begin{aligned} \forall\, \pi \in \{0,1\}^*, \quad & \Pr_r[M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } u \text{ or } v] \\ & \leq \sum_{a\in\{u,v\}} \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n; r) \text{ queries } O \text{ at } a \\ \text{or gets back } a \text{ from } O \end{array} \right] \\ & < 2 \cdot \frac{1}{6} = \frac{1}{3} \ . \end{aligned}$$

This means that for every $(u,v) \in Q_\star(O)$ it holds that M cannot distinguish between O and $O^{(u,v)}$ with probability greater than or equal to $\frac{1}{3}$.

We know that M^O succeeds on 0^n, namely, for every proof string π it holds that $M^{O,\pi}(0^n)$ accepts with probability no more than 1/3. We also know that for every $(u,v) \in I(O)$ it holds that 0^n is in $L_{O^{(u,v)}}$. But the foregoing argument tells us that for every $(u,v) \in Q_\star(O)$ it holds that for every proof string π we have that $M^{O^{(u,v)},\pi}(0^n)$ accepts with probability less than $1/3 + 1/3 = 2/3$. We deduce that, for every $(u,v) \in Q_\star(O)$, $M^{O^{(u,v)}}$ fails on 0^n. □

7.2 Proof of Part 2 of Theorem 7.4

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{DTIME}(t)^{\mathcal{O}}$ but not in $\mathrm{PCP}(o(t), o(t))^{\mathcal{O}}$.

Oracle language. Let $u_{k,i}$ denote the $\lceil \log k \rceil$-bit string that represents the index $i \in [k]$. Let $t^*(n) := t(n)/n$. Let $p(n)$ denote the largest prime in the interval $[2^{n-1}t(n), 2^n t(n))$. The oracle language $\mathcal{L} = \{L_O\}_{O\in\mathcal{O}}$ is defined as follows:

$$L_O := \{(x,y) \in \{0,1\}^n \times \{0,1\}^n \mid F_{O,n}(x) = y\} \ ,$$

where $F_{O,n}(x)$ is the n-bit string whose i-th bit is

$$\bigoplus_{j\in\{(i-1)\cdot t^*(n)+1,\ldots,i\cdot t^*(n)\}} O\left(1,1,x\|u_{t(n),j}, x\|u_{t(n),j}\right)_1 \ ,$$

for $i \in \{1,\ldots,n\}$.

In DTIME. We argue that $\mathcal{L}$ is in $\mathrm{DTIME}(t)^{\mathcal{O}}$. Consider the deterministic machine that on input (x,y): (a) copies x to the query tape; (b) for every $i \in \{1,\ldots,n\}$, calls O on inputs

$$\{(1,1,x\|u_{t(n),j}, x\|u_{t(n),j})\}_{j\in\{(i-1)\cdot t^*(n)+1,\ldots,i\cdot t^*(n)\}}$$

to get $z_j := F_O(x)_i$; (c) accepts if $y = z$. Making the oracle queries takes $O(t(n))$ time, computing z takes $O(t(n))$ time, and comparing y and z takes $O(t(n))$ time. So the running time of the machine is $O(t(n))$.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(o(t), o(t))^{\mathcal{O}}$. We begin with some notations and definitions.

- Let $e_{k,j}$ denote the k-bit string that has 1 in the j-th coordinate and 0 everywhere else. Use $\sigma_{p(n)}$ to denote the representation of O's order $p(n)$ group. For every $j \in \{1, \ldots, t(n)\}$, define two group elements

$$g^{(j)} := \sigma_{p(n)}^{-1}(0^n \| u_{t(n),j}) + \sigma_{p(n)}^{-1}(0^n \| u_{t(n),j}), \text{ and}$$
$$h^{(j)} := \sigma_{p(n)}^{-1}\left(\sigma_{p(n)}(g^{(j)}) \oplus e_{\lceil \log p(n) \rceil,1}\right) .$$

- For every $n \in \mathbb{N}$, define the set of oracles

$$X_n = \left\{ O \in \mathcal{O} \,\middle|\, \forall \text{ distinct } j, j' \in [t(n)],\ \sigma_{p(n)}(g^{(j)}) \neq \sigma_{p(n)}(h^{(j')}) \right\} .$$

- Define the oracle $O^{(j)}$ to be identical to O except the representation $\sigma_{p(n)}$ of O is changed to

$$\sigma_{p(n)}^{(j)}(g) = \begin{cases} \sigma_{p(n)}(g^{(j)}) & \text{if } g = h^{(j)} \\ \sigma_{p(n)}(h^{(j)}) & \text{if } g = g^{(j)} \\ \sigma_{p(n)}(g) & \text{otherwise} \end{cases} .$$

Therefore for every $x \in \{0,1\}^n$ and $j \in \{1, \ldots, t(n)\}$ $(x, F_{O,n}(x)) \notin L_{O^{(j)}}$.

We argue that any PCP-verifier M for $\mathcal{L}$ must have running time $\Omega(t(n))$.

- Since M is a PCP-verifier for $\mathcal{L}$, for every $n \in \mathbb{N}$ and every $O \in X_n$ there exists a proof π such that $M^{O,\pi}(0^n, F_{O,n}(0^n))$ accepts with probability at least $\frac{2}{3}$. Then, via Lemma 7.6 below, we deduce that there exists a randomness r such that $M^{O,\pi}(0^n, F_{O,n}(0^n); r)$ queries O at at least $\frac{t(n)}{3}$ different strings.

- Therefore the running time of $M^{O,\pi}(0^n, F_{O,n}(0^n); r)$ is at least $\frac{t(n)}{3} \in \Omega(t(n))$.

We have shown that any PCP-verifier for $\mathcal{L}$ has running time in $\Omega(t(n))$, and so $\mathcal{L} \notin \mathrm{PCP}(o(t), o(t))^{\mathcal{O}}$.

Lemma 7.6. *If M is a PCP-verifier for $\mathcal{L}$ then, for every $n \in \mathbb{N}$ and every $O \in X_n$, there exists a proof π and randomness r such that $M^{O,\pi}(0^n, F_{O,n}(0^n); r)$ queries O at at least $\frac{t(n)}{3}$ different strings.*

Proof. Since M is a PCP-verifier for $\mathcal{L}$, for every n, $(0^n, F_{O,n}(0^n)) \in L_O$ for every $O \in X_n$. So for every $O \in X_n$ there exists a proof π such that $M^{O,\pi}(0^n, F_{O,n}(0^n))$ accepts with probability at least $\frac{2}{3}$. If we change O to obtain $O^{(j)}$, then $F_{O^{(j)}}(0^n) \neq F_O(0^n)$. So $M^{O^{(j)},\pi}(0^n, F_{O,n}(0^n))$ should accept with probability at most $\frac{1}{3}$. This implies that for at least $\frac{1}{3}$ fraction of randomness r, $M^{O,\pi}(0^n, F_{O,n}(0^n); r)$ queries $\sigma_{p(n)}(g^{(j)})$ or $\sigma_{p(n)}(h^{(j)})$. By averaging over all $j \in [t(n)]$, there exists r^* such that $M^{O,\pi}(0^n, F_{O,n}(0^n); r^*)$ queries O at $\sigma_{p(n)}(g^{(j)})$ or $\sigma_{p(n)}(h^{(j)})$ for at least $\frac{t(n)}{3}$ different $j \in [t(n)]$. Further more, by definition of X_n, for any two distinct indices $j, j' \in [t(n)]$ the four group elements $\sigma_{p(n)}(g^{(j)})$, $\sigma_{p(n)}(h^{(j)})$, $\sigma_{p(n)}(g^{(j')})$, and $\sigma_{p(n)}(h^{(j')})$ are distinct. So $M^{O,\pi}(0^n, F_{O,n}(0^n); r^*)$ queries O at at least $\frac{t(n)}{3}$ distinct strings. □

8 Almost-everywhere separation for random functions

We strengthen the separation for random functions in Theorem 5.2. The difference is that now the choice of machine M, which is the candidate PCP-verifier for L_R, *depends* on the sample R.

Theorem 8.1. *Let $\mathcal{R}$ be a random oracle with output length $\ell\colon \mathbb{N} \to \mathbb{N}$.*

1. For any function $t\colon \mathbb{N} \to \mathbb{N}$,

$$\Pr_{R \leftarrow \mathcal{R}}\left[\mathrm{NTIME}(t)^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^R\right] = 1 \ .$$

2. For any function $t\colon \mathbb{N} \to \mathbb{N}$,

$$\Pr_{R \leftarrow \mathcal{R}}\left[\mathrm{DTIME}(t)^R \not\subseteq \mathrm{PCP}(o(t), o(t))^R\right] = 1 \ .$$

8.1 Proof of Part 1 of Theorem 8.1

We prove the statement by arguing that, for every oracle R in a certain set of measure 1 (derived below), there exists a language L_R that is in $\mathrm{NTIME}(t)^R$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^R$.

We first define the language $L_R \subseteq \{0,1\}^*$ for any $R \in \mathcal{R}$. Let $u_{k,i}$ denote the $\lceil \log k \rceil$-bit string that represents the index $i \in [k]$. The language L_R is defined as follows:

$$L_R := \left\{ 0^n \;\middle|\; \exists\, w \in \{0,1\}^{t(n)} \text{ s.t. } \begin{array}{c} R(w \| u_{t(n),1})_1 = 0 \\ R(w \| u_{t(n),2})_1 = 0 \\ \vdots \\ R(w \| u_{t(n),t(n)})_1 = 0 \end{array} \right\} \ .$$

The language L_R is in $\mathrm{NTIME}(t)^R$ for every $R \in \mathcal{R}$ (via the same argument as in Section 5.1). We are left to argue that L_R is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^R$ for R in a certain set of measure 1. For this, we state a lemma (which we prove later on below), and then conclude the proof of the theorem.

Lemma 8.2. *For every $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$, $\Pr_{R \leftarrow \mathcal{R}}\left[M^R \text{ is a PCP-verifier for } L_R\right] = 0$.*

Let S_M be the set of oracles $R \in \mathcal{R}$ for which M^R is a PCP-verifier for L_R. Lemma 8.2 tells us that S_M has measure zero, that is, $\mu_{\mathcal{R}}(S_M) = 0$. Since the set $\mathbf{M}_{\mathrm{poly}(t),o(t)}$ is countable (it is a subset of the countable set of all machines) and measures are countably sub-additive, we deduce that

$$\mu_{\mathcal{R}}\left(\bigcup_{M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}} S_M\right) \leq \sum_{M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}} \mu_{\mathcal{R}}(S_M) = 0 \ .$$

We conclude that

$$\Pr_{R \leftarrow \mathcal{R}}\left[\exists\, M \in \mathbf{M}_{\mathrm{poly}(t),o(t)} \text{ s.t. } M^R \text{ is a PCP-verifier for } L_R\right] = 0 \ ,$$

which shows that L_R is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^R$ for all R in a set of measure 1.

This completes the proof, and so we are only left with proving Lemma 8.2.

Before proving Lemma 8.2, we define two disjoint sets of oracles, $S_{n,0}$ and $S_{n,1}$, and then prove certain properties about them (see Lemmas 8.5 to 8.7 below).

Definition 8.3. *For every $n \in \mathbb{N}$, function $R \in \mathcal{R}_n$, and string $w \in \{0,1\}^{t(n)}$, we define the function $\mathbf{F}[R,w] \colon \{0,1\}^* \to \{0,1\}^*$ to be*

$$\mathbf{F}[R,w](z)_j := \begin{cases} 0 & \text{if } j = 1 \text{ and } z = w \| u_{t(n),i} \text{ for some } i \in [t(n)] \\ R(z)_j & \text{otherwise} \end{cases} .$$

Moreover, given $S \subseteq \mathcal{R}_n$, we define $\mathbf{F}[S, \{0,1\}^{t(n)}]$ to be the set $\{\mathbf{F}[R,w] \mid R \in S \, , \, w \in \{0,1\}^{t(n)}\}$.

Definition 8.4. *For every $n \in \mathbb{N}$, $S_{n,0}$ is the set of functions $R \in \mathcal{R}_n$ for which $0^n \notin L_R$, that is, for which for every $w \in \{0,1\}^{t(n)}$ there exists an index $i \in [t(n)]$ such that $R(w \| u_{t(n),i})_1 \neq 0$. Also, $S_{n,1}$ equals the set $\mathbf{F}[S_{n,0}, \{0,1\}^{t(n)}]$, which is disjoint from $S_{n,0}$ (since $0^n \in L_R$ for every $R \in S_{n,1}$).*

Lemma 8.5. *For every subset $S \subseteq S_{n,0}$, we have $\mu_{\mathcal{R}}(S) \leq \mu_{\mathcal{R}}(\mathbf{F}[S, \{0,1\}^{t(n)}])$.*

Proof. Each $R \in S$ yields $2^{t(n)}$ distinct functions $\mathbf{F}[R,w]$ as w ranges over $\{0,1\}^{t(n)}$. On the other hand, each $R' \in \mathbf{F}[S, \{0,1\}^{t(n)}]$ has at most $2^{t(n)} - 1$ "pre-images" in S: there exists precisely one w such that $R(w \| u_{t(n),i})_1 = 0$ for all $i \in \{1, \ldots, t(n)\}$. So if $R' = \mathbf{F}[R,w]$, R and R' can only be different in the first bit at locations of the form $w \| u_{t(n),i}$ for $i \in \{1, \ldots, t(n)\}$. There are $2^{t(n)} - 1$ different assignments to the first bits at $w \| u_{t(n),i}$ each of which gives rise to a preimage of R' in S (we exclude the all-zero assignment). We deduce that $2^{t(n)} \mu_{\mathcal{R}}(S) \leq \left(2^{t(n)} - 1\right) \mu_{\mathcal{R}}(\mathbf{F}[S, \{0,1\}^{t(n)}])$, and so $\mu_{\mathcal{R}}(S) \leq \mu_{\mathcal{R}}(\mathbf{F}[S, \{0,1\}^{t(n)}])$. □

Lemma 8.6. $\lim_{n \to \infty} \mu_{\mathcal{R}}(S_{n,0}) = 1/e$.

Proof. For any $n \in \mathbb{N}$ the measure of $S_{n,0}$ in $\mathcal{R}$ is $\mu_{\mathcal{R}}(S_{n,0}) = (1 - \frac{1}{2^{t(n)}})^{2^{t(n)}}$. Therefore:

$$\lim_{n \to \infty} \mu_{\mathcal{R}}(S_{n,0}) = \lim_{N \to \infty} \left(1 - \frac{1}{N}\right)^N = \lim_{N \to \infty} e^{N\left(1 - \frac{1}{N}\right)} = \lim_{N \to \infty} e^{\left(1 - \frac{1}{N}\right)' / \left(\frac{1}{N}\right)'} = 1/e \ . \qquad \square$$

Lemma 8.7. *For every function $R \in S_{n,0}$, if M^R succeeds on 0^n then there are at least $2^{t(n)} - 2^{o(t(n))}$ strings $w \in \{0,1\}^{t(n)}$ for which $M^{\mathbf{F}[R,w]}$ fails on 0^n. (Note that $\mathbf{F}[R,w] \in S_{n,1}$.)*

Proof. Fix a constant $c > 0$ to be determined later. Consider the set of strings $w \in \{0,1\}^{t(n)}$ for which M^R queries all of $\{w \| u_{t(n),1}, \ldots, w \| u_{t(n),t(n)}\}$ with "low" probability:

$$Q_\star(R) := \left\{ w \in \{0,1\}^{t(n)} \;\middle|\; \forall\, i \in [t(n)] \, , \sum_{\pi \in \{0,1\}^*} \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w \| u_{t(n),i}] < \frac{1}{c \cdot t(n)} \right\} .$$

We argue that $|Q_\star(R)|$ is large, and for every $w \in Q_\star(R)$ it holds that $M^{\mathbf{F}[R,w]}$ fails on 0^n.

Consider the set of strings $w \in \{0,1\}^{t(n)}$ that M^R queries with "high" probability:

$$Q_c(R) := \left\{ w \| u \in \{0,1\}^{t(n) + \lceil \log t(n) \rceil} \;\middle|\; \sum_{\pi \in \{0,1\}^*} \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w \| u \,] \geq \frac{1}{c \cdot t(n)} \right\} .$$

Observe that

$$|Q_c(R)| \leq \Big(c \cdot t(n)\Big) \cdot \Big(2^{o(t(n))} \cdot \text{poly}(t(n))\Big) = 2^{o(t(n))} \ . \qquad (4)$$

This is because, in any given execution, M can make at most $\mathrm{poly}(t(n))$ queries to R and $o(t(n))$ queries to the given proof string, which means that

$$\sum_{w\|u \in \{0,1\}^{t(n)+\lceil \log t(n) \rceil}} \sum_{\pi \in \{0,1\}^*} \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w\|u] \leq 2^{o(t(n))}\mathrm{poly}(t(n)) \ .$$

We deduce, via Equation (5), that $|Q_\star(R)|$ is large:

$$|Q_\star(R)| \geq 2^{t(n)} - t(n)|Q_c(R)| = 2^{t(n)} - 2^{o(t(n))} \ .$$

Next, for every $w \in Q_\star(R)$ it holds that

$$\forall\, \pi \in \{0,1\}^*,\ \forall\, i \in [t(n)],\ \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w\|u_{t(n),i}] < \frac{1}{c \cdot t(n)} \ .$$

Therefore,

$$\begin{aligned}\forall\, \pi \in \{0,1\}^*, \quad & \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at any } w\|u_{t(n),i}] \\ & \leq \sum_{i\in[t(n)]} \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w\|u_{t(n),i}] \\ & < t(n) \cdot \frac{1}{c \cdot t(n)} = \frac{1}{c} \ .\end{aligned}$$

This means that for every $w \in Q_\star(R)$ it holds that M cannot distinguish between R and $R' := \mathbf{F}[R,w]$ with probability greater than $1/c$ (the probability is over M's randomness r).

We know that M^R succeeds on 0^n, namely, for every proof string π it holds that $M^{R,\pi}(0^n)$ accepts with probability less than $1/3$. We also know that for every $w \in \{0,1\}^{t(n)}$ it holds that 0^n is in $L_{\mathbf{F}[R,w]}$. But the foregoing argument tells us that for every $w \in Q_\star(R)$ it holds that for every proof string π we have that $M^{\mathbf{F}[R,w],\pi}(0^n)$ accepts with probability less than $1/3 + 1/c$.

Choosing $c \geq 3$, we deduce that, for every $w \in Q_\star(R)$, $M^{\mathbf{F}[R,w]}$ fails on 0^n. □

Proof of Lemma 8.2. Fix $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$. It suffices to show that, for some $\epsilon \in [0,1)$ and every $n \in \mathbb{N}$, M^R succeeds on input 0^n for at most an ϵ-fraction of oracles R. Indeed, this fact would imply that:

$$\begin{aligned}& \Pr_{R\leftarrow\mathcal{R}}\left[\, M^R \text{ is a PCP-verifier for } L_R \,\right] \\ & \leq \Pr_{R\leftarrow\mathcal{R}}\left[\, \forall n \in \mathbb{N},\ M^R \text{ succeeds on } 0^n \,\right] \\ & = \prod_{n\in\mathbb{N}} \Pr_{R\leftarrow\mathcal{R}}\left[\, M^R \text{ succeeds on } 0^n \;\middle|\; M^R \text{ succeeds on } 0^i \text{ for all } i < n \,\right] \\ & = \prod_{n\in\mathbb{N}} \Pr_{R\leftarrow\mathcal{R}}\left[\, M^R \text{ succeeds on } 0^n \,\right] \\ & \leq \lim_{n\to\infty} \epsilon^n = 0 \ ,\end{aligned}$$

as claimed. Above "succeeds" on input 0^n means that if $0^n \in L_R$ then there exists a proof string π such that $M^{R,\pi}(0^n)$ accepts with probability at least $2/3$, and if $0^n \notin L_R$ then for every proof string π it holds that $M^{R,\pi}(0^n)$ rejects with probability at least $2/3$.

We are left to argue that M^R succeeds on input 0^n for at most an ϵ-fraction of oracles R. Consider the following sets of oracles:

$$\begin{aligned}U_{n,\mathsf{all}} &:= \left\{ R \in \mathcal{R} \,|\, M^R \text{ fails on } 0^n \right\} \ , \\ U_{n,0} &:= \left\{ R \in S_{n,0} \,|\, M^R \text{ fails on } 0^n \right\} \ , \\ U_{n,1} &:= \left\{ R' \in S_{n,1} \,|\, M^{R'} \text{ fails on } 0^n \right\} \ .\end{aligned}$$

Note that $U_{n,0} \cup U_{n,1} \subseteq U_{n,\mathsf{all}}$. Also, $U_{n,0}$ and $U_{n,1}$ are disjoint, because $S_{n,0}$ and $S_{n,1}$ are disjoint.

We want to prove that $\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) > 1 - \epsilon$. We do so as follows:

$$\begin{aligned}\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) &\geq \mu_{\mathcal{R}}(U_{n,0}) + \mu_{\mathcal{R}}(U_{n,1}) \\ &\geq \Big(\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0})\Big) + \mu_{\mathcal{R}}(U_{n,1}) \\ &\geq_{[a]} \Big(\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0})\Big) + \frac{2^{t(n)} - 2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}\left(\mathbf{F}[S_{n,0} \setminus U_{n,0}, \{0,1\}^{t(n)}]\right) \\ &\geq_{[b]} \Big(\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0})\Big) + \frac{2^{t(n)} - 2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0}) \\ &= \mu_{\mathcal{R}}(S_{n,0}) - \frac{2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0}) \\ &\geq \mu_{\mathcal{R}}(S_{n,0}) - \frac{2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}(S_{n,0}) \\ &= \left(1 - \frac{2^{o(t(n))}}{2^{t(n)}}\right) \cdot \mu_{\mathcal{R}}(S_{n,0}) \ .\end{aligned}$$

Above, the third inequality (labeled [a]) follows from Lemma 8.7; the fourth inequality (labeled [b]) follows from Lemma 8.5 applied to the set $S_{n,0} \setminus U_{n,0}$.

Finally, by Lemma 8.6 we know that $\lim_{n\to\infty} \mu_{\mathcal{R}}(S_{n,0}) = 1/e > 1/3$, so if we set $\epsilon := 2/3$ then the above expression is greater than $1 - \epsilon$ for large enough n. □

8.2 Proof of Part 2 of Theorem 8.1

We prove the statement by arguing that, for every oracle R in a certain set of measure 1 (derived below), there exists a language L_R that is in $\mathrm{DTIME}(t)^R$ but not in $\mathrm{PCP}(o(t), o(t))^R$.

We first define the language $L_R \subseteq \{0,1\}^*$ for any $R \in \mathcal{R}$.

Let $u_{k,i}$ denote the $\lceil \log k \rceil$-bit string that represents the index $i \in [k]$. For convenience we use $t^*(n)$ to denote $t(n)/n$. For every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, we define $F_{R,n}(x)$ to be the n-bit string whose i-th bit is $\bigoplus_{j\in\{(i-1)\cdot t^*(n)+1,\ldots,i\cdot t^*(n)\}} R(x\|u_{t(n),j})_1$, for $i \in [n]$. The language L_R is then defined as follows:

$$L_R := \{(x,y) \in \{0,1\}^n \times \{0,1\}^n \,|\, F_{R,n}(x) = y\} \ .$$

We already showed that the language L_R is in $\mathrm{DTIME}(t)^R$ in Section 5.2.

We are left to argue that L_R is not in $\mathrm{PCP}(o(t), o(t))^R$ for R in a certain set of measure 1. For this, we state a lemma (which we prove later on below), and then conclude the proof of the theorem.

Lemma 8.8. *For every $M \in \mathbf{M}_{o(t),o(t)}$, $\Pr_{R\leftarrow\mathcal{R}}\left[M^R \text{ is a PCP-verifier for } L_R\right] = 0$.*

Using the same arguement as in the proof of Theorem 8.1, we conclude that

$$\Pr_{R \leftarrow \mathcal{R}} \left[\exists\, M \in \mathbf{M}_{o(t),o(t)} \text{ s.t. } M^R \text{ is a PCP-verifier for } L_R \right] = 0 \ ,$$

which shows that L_R is not in $\mathrm{PCP}(o(t), o(t))^R$ for all R in a set of measure 1.

This completes the proof, and so we are only left with proving Lemma 8.8.

Before proving Lemma 8.8, we define 2^n pairwise disjoint subsets of oracles $\{S_{n,y}\}_{y \in \{0,1\}^n}$ and then prove certain properties about them (see Lemmas 8.11 and 8.12 below).

Definition 8.9. *For every $n \in \mathbb{N}$ and $y \in \{0,1\}^n$, $S_{n,y}$ is the set of functions $R \in \mathcal{R}_n$ for which $(0^n, y) \in L_R$, that is, for which $F_{R,n}(0^n) = y$. Note that the sets $\{S_{n,y}\}_{y \in \{0,1\}^n}$ are pairwise disjoint.*

Definition 8.10. *For every $n \in \mathbb{N}$, function $R \in \mathcal{R}_n$, and indices $i \in [n]$ and $j \in [t^*(n)]$, we define the function $\mathbf{F}[R,i,j] \colon \{0,1\}^* \to \{0,1\}^*$ to be*

$$\mathbf{F}[R,i,j](z)_k := \begin{cases} 1 - R(z)_k & \text{if } k = 1 \text{ and } z = 0^n \| u_{t(n),(i-1)\cdot t^*(n)+j} \\ R(z)_k & \text{otherwise} \end{cases} \ .$$

Moreover, given $S \subseteq \mathcal{R}_n$, we define $\mathbf{F}[S,i,j]$ to be the set $\{\mathbf{F}[R,i,j] \mid R \in S\}$.

Lemma 8.11. *For every $R \in S_{n,y}$, $i \in [n]$, and $j \in [t^*(n)]$: (a) $\mathbf{F}[R,i,j] \notin S_{n,y}$; and (b) the number of preimages of R under the maps $\{\mathbf{F}[\cdot,i,j]\}_{i \in [n], j \in [t^*(n)]}$ is $t(n)$.*

Proof. For the first part, flipping the first bit of $R(0^n \| u_{t(n),(i-1)t^*(n)+j})$ results in $F_{R,n}(0^n)_i \neq F_{\mathbf{F}[R,i,j],n}(0^n)_i$. For the second part, for each choice of $i \in [n], j \in [t^*(n)]$, we flip the first bit of $R(0^n \| u_{t(n),(i-1)\cdot t^*(n)+j})$ to obtain $\mathbf{F}[R,i,j]$. We further observe that by definition $\mathbf{F}[\mathbf{F}[R,i,j],i,j] = R$, so $\mathbf{F}[R,i,j]$ is the preimage of R under the map $\mathbf{F}[\cdot,i,j]$. Therefore $\{\mathbf{F}[R,i,j]\}_{i \in [n], j \in [t^*(n)]}$ are exactly the $n \cdot t^*(n) = t(n)$ preimages of R under the maps $\{\mathbf{F}[\cdot,i,j]\}_{i \in [n], j \in [t^*(n)]}$. □

Lemma 8.12. *For every function $R \in S_{n,y}$, if M^R succeeds on $(0^n, y)$ then there are at least $t(n) - o(t(n))$ pairs $(i,j)_{i \in [n], j \in [t^*(n)]}$ for which $M^{\mathbf{F}[R,i,j]}$ fails on $(0^n, y)$.*

Proof. Fix a constant $c > 0$. Let π be the proof such that $\Pr_r[M^{R,\pi}(0^n, y; r)] \geq \frac{2}{3}$. Consider the set of $t(n)$ queries

$$Q := \left\{ 0^n \| u_{t(n),(i-1)t^*(n)+j} \right\}_{i \in [n],\, j \in [t^*(n)]} \ .$$

Define the subset of Q which $M^{R,\pi}$ queries with "low" probability:

$$Q_\star(R) := \left\{ s \in Q \;\middle|\; \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } s] < \frac{1}{c} \right\} \ .$$

We argue that $|Q_\star(R)|$ is large, and for every $s \in Q_\star(R)$, letting $s = 0^n \| u_{t(n),(i-1)t^*(n)+j}$, it holds that $M^{\mathbf{F}[R,i,j],\pi}$ fails on $(0^n, y)$.

Consider the set of strings $s \in Q$ that M^R queries with "high" probability:

$$Q_c(R) := \left\{ s \in Q \;\middle|\; \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } s\ ] \geq \frac{1}{c} \right\} \ .$$

Observe that

$$|Q_c(R)| \leq c \cdot o(t(n)) \in o(t(n)) \ . \tag{5}$$

This is because, in any given execution, M can make at most $o(t(n))$ queries to R which means that

$$\sum_{s \in Q} \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } s] \leq o(t(n)) \ .$$

We deduce, via Equation (5), that $|Q_\star(R)|$ is large:

$$|Q_\star(R)| \geq t(n) - |Q_c(R)| = t(n) - o(t(n)) \ .$$

Next, for every $s \in Q_\star(R)$ it holds that

$$\Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } s] < \frac{1}{c} \ .$$

This means that for every $s \in Q_\star(R)$, letting $s = 0^n \| u_{t(n),(i-1)t^*(n)+j}$, it holds that M cannot distinguish between R and $R' := \mathbf{F}[R, i, j]$ with probability greater than $1/c$ (the probability is over M's randomness r).

We know that $M^{R,\pi}$ succeeds on $(0^n, y)$, namely, $M^{R,\pi}(0^n)$ accepts with probability at least $1/3$. We also know that for every $(i, j) \in [n] \times [t^*(n)]$ it holds that $(0^n, y) \notin L_{\mathbf{F}[R,i,j]}$ (Part a of Lemma 8.11). But the foregoing argument tells us that for every $0^n \| u_{t(n),(i-1)t^*(n)+j} \in Q_\star(R)$ it holds that $M^{\mathbf{F}[R,i,j],\pi}(0^n)$ accepts with probability greater than $1/3 + 1/c$.

Choosing $c \geq 3$, we deduce that, for every $0^n \| u_{t(n),(i-1)t^*(n)+j} \in Q_\star(R)$, $M^{\mathbf{F}[R,i,j]}$ fails on $(0^n, y)$. $\square$

Proof of Lemma 8.8. Fix $M \in \mathbf{M}_{o(t),o(t)}$. It suffices to show that, for some $\epsilon \in [0, 1)$ and every $n \in \mathbb{N}$, M^R succeeds on inputs $\{(0^n, y)\}_{y \in \{0,1\}^n}$ for at most an ϵ-fraction of oracles R. Indeed, this fact would imply that:

$$\begin{aligned}
&\Pr_{R \leftarrow \mathcal{R}}\left[\, M^R \text{ is a PCP-verifier for } L_R \,\right] \\
&\leq \Pr_{R \leftarrow \mathcal{R}}\left[\, \forall n \in \mathbb{N},\ M^R \text{ succeeds on } \{(0^n, y)\}_{y \in \{0,1\}^n} \,\right] \\
&= \prod_{n \in \mathbb{N}} \Pr_{R \leftarrow \mathcal{R}}\left[\, M^R \text{ succeeds on } \{(0^n, y)\}_{y \in \{0,1\}^n} \;\middle|\; M^R \text{ succeeds on } \{(0^i, y)\}_{y \in \{0,1\}^i} \text{ for all } i < n \,\right] \\
&= \prod_{n \in \mathbb{N}} \Pr_{R \leftarrow \mathcal{R}}\left[\, M^R \text{ succeeds on } \{(0^n, y)\}_{y \in \{0,1\}^n} \,\right] \\
&\leq \lim_{n \to \infty} \epsilon^n = 0 \ ,
\end{aligned}$$

as claimed. Above "succeeds" on input $(0^n, y)$ means that if $(0^n, y) \in L_R$ then there exists a proof string π such that $M^{R,\pi}(0^n, y)$ accepts with probability at least $2/3$, and if $(0^n, y) \notin L_R$ then for every proof string π it holds that $M^{R,\pi}(0^n, y)$ rejects with probability at least $2/3$.

We are left to argue that M^R succeeds on inputs $\{(0^n, y)\}_{y \in \{0,1\}^n}$ for at most an ϵ-fraction of oracles R. Consider the following sets of oracles:

$$\begin{aligned}
U_{n,\mathsf{all}} &:= \left\{ R \in \mathcal{R} \,|\, M^R \text{ fails on } (0^n, y) \text{ for some } y \in \{0,1\}^n \right\} \ , \\
U_{y,\mathsf{all}} &:= \left\{ R \in S_{n,y} \,|\, M^R \text{ fails on } (0^n, y') \text{ for some } y' \in \{0,1\}^n \right\} \ , \\
U_{y,y} &:= \left\{ R \in S_{n,y} \,|\, M^R \text{ fails on } (0^n, y) \right\} \ .
\end{aligned}$$

We want to prove that $\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) > 1 - \epsilon = 1/3$. We do so as follows:

$$\begin{aligned}
\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) &= \sum_{y \in \{0,1\}^n} \mu_{\mathcal{R}}(U_{y,\mathsf{all}}) \\
&\geq_{[a]} \frac{1}{t(n)} \cdot \sum_{y \in \{0,1\}^n} \mu_{\mathcal{R}}(S_{n,y} \setminus U_{y,y}) \cdot (t(n) - o(t(n))) \\
&\geq \sum_{y \in \{0,1\}^n} \mu_{\mathcal{R}}(S_{n,y} \setminus U_{y,\mathsf{all}}) \cdot \frac{t(n) - o(t(n))}{t(n)} \\
&= (1 - \mu_{\mathcal{R}}(U_{n,\mathsf{all}})) \cdot \frac{t(n) - o(t(n))}{t(n)} \\
&\geq \frac{t(n) - o(t(n))}{2t(n)} > \frac{1}{3} \ .
\end{aligned}$$

In the inequality labeled $[a]$, the $\frac{1}{t(n)}$ term comes from Part b of Lemma 8.11 that each $R \in \mathcal{R}_n$ has $t(n)$ preimages under the maps $\{\mathbf{F}[\cdot, i, j]\}_{i \in [n], j \in [t^*(n)]}$; the term $\mu_{\mathcal{R}}(S_{n,y} \setminus U_{y,y}) \cdot (t(n) - o(t(n)))$ is the measure of all $R' \in \cup_{i,j} \mathbf{F}[S_{n,y}, i, j]$ for which $M^{R',\pi}(0^n, y)$ fails (Lemma 8.12). □

9 Separation for almost random functions

We strengthen the robustness of separation (see Theorem 3.2) for the random oracle $\mathcal{R}$, by using the stronger separation result from Section 8. Namely, we derive an explicit distance threshold such that all oracles that are closer to $\mathcal{R}$ than this threshold separate NTIME and PCP.

Lemma 9.1. *For any $\nu \in [0, \frac{1}{3e})$, if an oracle $\mathcal{A}$ is ν-close to the random oracle $\mathcal{R}$, then*

$$\mathrm{NTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{A}} \ .$$

Proof. Define $S_{n,0}$ and $S_{n,1}$ with respect to $\mathcal{R}$ (see Definition 8.4). Let V denote the set of all values that $\mathcal{A}$ and $\mathcal{R}$ can take. If the statistical distance between $\mathcal{A}$ and $\mathcal{R}$ is at most ν, then by definition

$$\max_{W \subseteq V} |\mu_{\mathcal{R}}(W) - \mu_{\mathcal{A}}(W)| \leq \nu \ .$$

Setting $W := S_{n,0}$, we obtain $|\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{A}}(S_{n,0})| \leq \nu$, and so $\mu_{\mathcal{A}}(S_{n,0}) \geq \mu_{\mathcal{R}}(S_{n,0}) - \nu$. Define $S'_{n,0} = \{f \in S_{n,0} \mid \mu_{\mathcal{A}}(f) > 0\}$. By definition of $S'_{n,0}$, $\mu_{\mathcal{A}}(S'_{n,0}) = \mu_{\mathcal{A}}(S_{n,0})$. Apply the definition of statistical distance to $W := S'_{n,0}$ to get $\left|\mu_{\mathcal{R}}(S'_{n,0}) - \mu_{\mathcal{A}}(S'_{n,0})\right| \leq \nu$. Hence $\mu_{\mathcal{R}}(S'_{n,0}) \geq \mu_{\mathcal{R}}(S_{n,0}) - 2\nu$.

We deduce that

$$\Pr_{A \leftarrow S_{n,0}}\left[\forall \pi \ \Pr_r[M^{A,\pi}(0^n; r) = 1] \leq \frac{1}{3}\right] = 1 \Rightarrow \Pr_{R \leftarrow S_{n,0}}\left[\forall \pi \ \Pr_r[M^{R,\pi}(0^n; r) = 1] \leq \frac{1}{3}\right] \geq \frac{\mu_{\mathcal{R}}(S_{n,0}) - 2\nu}{\mu_{\mathcal{R}}(S_{n,0})} \ .$$

In this case, by Lemma 8.7,

$$\Pr_{R \leftarrow S_{n,1}}\left[\forall \pi \ \Pr_r[M^{R,\pi}(0^n; r) = 1] < \frac{2}{3}\right] \geq \frac{\mu_{\mathcal{R}}(S_{n,0}) - 2\nu}{\mu_{\mathcal{R}}(S_{n,0})} \cdot \left(1 - \frac{2^{o(t)}}{2^t}\right) \ .$$

Define $S'_{n,1} := \{f \in S_{n,1} \mid \forall \pi \ \Pr_r[M^{f,\pi}(0^n; r) = 1] < \frac{2}{3}\}$ and observe that

$$\mu_{\mathcal{R}}(S'_{n,1}) \geq \frac{\mu_{\mathcal{R}}(S_{n,1})}{\mu_{\mathcal{R}}(S_{n,0})} \left(\mu_{\mathcal{R}}(S_{n,0}) - 2\nu\right) \left(1 - \frac{2^{o(t)}}{2^t}\right) \ . \tag{6}$$

If the right-hand side of Equation (6) is greater than ν, we can deduce that $\mu_{\mathcal{A}}(S'_{n,1}) > 0$ and thus

$$\Pr_{A \leftarrow S_{n,1}}\left[\exists \pi \ \Pr_r[M^{A,\pi}(0^n; r) = 1] \geq \frac{2}{3}\right] < 1 \ .$$

We now argue that this is the case for any $\nu \in [0, \frac{1}{3e})$ and for large enough n. Write ν as $\nu = \frac{1}{3e} - \delta$.

By Lemma 8.6, for large enough n, $\mu_{\mathcal{R}}(S_{n,0}) \geq \frac{1}{e} - \frac{\delta}{2}$. Therefore,

$$\begin{aligned}
&\frac{\mu_{\mathcal{R}}(S_{n,1})}{\mu_{\mathcal{R}}(S_{n,0})} \left(\mu_{\mathcal{R}}(S_{n,0}) - 2\nu\right) \left(1 - \frac{2^{o(t)}}{2^t}\right) - \nu \\
&> \left(\mu_{\mathcal{R}}(S_{n,0}) - 2\nu\right) \left(1 - \frac{2^{o(t)}}{2^t}\right) - \nu \quad \text{(by Lemma 8.5)} \\
&= \mu_{\mathcal{R}}(S_{n,0}) \left(1 - \frac{2^{o(t)}}{2^t}\right) - \left(3 - 2\frac{2^{o(t)}}{2^t}\right) \nu \\
&\geq \left(\frac{1}{e} - \frac{\delta}{2}\right) \left(1 - \frac{2^{o(t)}}{2^t}\right) - \left(3 - 2\frac{2^{o(t)}}{2^t}\right) \left(\frac{1}{3e} - \delta\right) \\
&= \left(\frac{1}{6e} - \frac{3}{2}\delta\right) \frac{2^{o(t)}}{2^t} + \frac{5}{2}\delta \ .
\end{aligned}$$

If $\delta \leq \frac{1}{9e}$, the last expression is strictly positive. If instead $\delta > \frac{1}{9e}$, then for n large enough it holds that $\frac{2^{o(t)}}{2^t} \leq \frac{5\delta/2}{3\delta/2 - 1/6e}$, and in this case the last expression is non-negative.

We have shown that for any $\nu \in [0, \frac{1}{3e})$, if $\mathcal{A}$ and $\mathcal{R}$ are ν-close, then

$$\Pr_{A \leftarrow S_{n,0}} \left[\forall \pi \ \Pr_r[M^{A,\pi}(0^n; r) = 1] \leq \frac{1}{3}\right] = 1 \Rightarrow \Pr_{A \leftarrow S_{n,1}} \left[\exists \pi \ \Pr_r[M^{A,\pi}(0^n; r) = 1] \geq \frac{2}{3}\right] < 1 \ .$$

Since $W_{0^n,0} = S_{n,0}$ and $W_{0^n,1} \supseteq S_{n,1}$, we deduce that

$$\Pr_{A \leftarrow W_{0^n,0}} \left[\forall \pi \ \Pr_r[M^{A,\pi}(0^n; r) = 1] \leq \frac{1}{3}\right] = 1 \Rightarrow \Pr_{A \leftarrow W_{0^n,1}} \left[\exists \pi \ \Pr_r[M^{A,\pi}(0^n; r) = 1] \geq \frac{2}{3}\right] < 1 \ .$$

We conclude that $\mathcal{A}$ fools M_t. $\square$

Acknowledgments

This research was supported in part by: the Berkeley Haas Blockchain Initiative, and donations from the Ethereum Foundation and the Interchain Foundation.

References

[AABS$^+$19] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Efficient symmetric primitives for advanced cryptographic protocols. IACR Cryptology ePrint Archive, Report 2019/426, 2019.

[AB18] Bariş Aydınlıoğlu and Eric Bach. Affine relativization: Unifying the algebrization and relativization barriers. *ACM Transactions on Computation Theory*, 10(1):1:1–1:67, 2018.

[ACG$^+$19] Martin R Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic cryptanalysis of STARK-friendly designs: Application to MARVELlous and MiMC. IACR Cryptology ePrint Archive, Report 2019/419, 2019.

[AD18] Tomer Ashur and Siemen Dhooghe. MARVELlous: a STARK-friendly family of cryptographic primitives. IACR Cryptology ePrint Archive, Report 2018/1098, 2018.

[AGP$^+$19] Martin R Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for MPC, and more. IACR Cryptology ePrint Archive, Report 2019/397, 2019.

[AIV92] Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: The role of local checkability. Manuscript, 1992.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, 2009.

[BBC$^+$17] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear pcps. In *Proceedings of the 36th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '17, pages 551–579, 2017.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Ynon Horesh, and Michael Riabzev. Fast Reed–Solomon interactive oracle proofs of proximity. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*, ICALP '18, pages 14:1–14:17, 2018.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *Proceedings of the 39th Annual International Cryptology Conference*, CRYPTO '19, pages 733–764, 2019.

[BCF$^+$17] Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Zero knowledge protocols from succinct constraint detection. In

Proceedings of the 15th Theory of Cryptography Conference, TCC '17, pages 172–206, 2017.

[BCG$^+$17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Interactive oracle proofs with constant rate and query complexity. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*, ICALP '17, pages 40:1–40:15, 2017.

[BCG$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. Linear-size constant-query IOPs for delegating computation. In *Proceedings of the 17th Theory of Cryptography Conference*, TCC '19, 2019.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasilinear-size zero knowledge from linear-algebraic PCPs. In *Proceedings of the 13th Theory of Cryptography Conference*, TCC '16-A, pages 33–64, 2016.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '19, pages 103–128, 2019. Full version available at `https://eprint.iacr.org/2018/828`.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, TCC '16-B, pages 31–60, 2016.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991.

[BG81] Charles H Bennett and John Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography. In *Proceedings of the 16th Annual International Cryptology Conference*, CRYPTO '96, pages 283–297, 1996.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC '05.

[CCG$^+$94] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

[Cob65] Alan Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress in Logic, Methodology and Philosophy of Science*, pages 24–30, 1965.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[FGL$^+$91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual Symposium of Foundations of Computer Science*, pages 2–12, 1991.

[For94] Lance Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–243, 1994.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO '86, pages 186–194, 1986.

[GKK$^+$19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Starkad and Poseidon: New hash functions for zero knowledge proof systems. IACR Cryptology ePrint Archive, Report 2019/458, 2019.

[GLR$^+$19] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The HADES design strategy. Cryptology ePrint Archive, Report 2019/1107, 2019.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC '85.

[HCC$^+$92] Juris Hartmanis, Richard Chang, Suresh Chari, Desh Ranjan, and Pankaj Rohatgi. Relativization: A revisionistic retrospective. In *Bulletin of the EATCS*, 1992.

[IKK09] Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 695–704, 2009.

[KR08] Yael Kalai and Ran Raz. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP '08, pages 536–547, 2008.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography. In *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, IMA '05, pages 1–12, 2005.

[Mie09] Thilo Mie. Short PCPPs verifiable in polylogarithmic time with o(1) queries. *Annals of Mathematics and Artificial Intelligence*, 56:313–338, 2009.

[MW98] Ueli Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In *Proceedings of the 17th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '98, pages 72–84, 1998.

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55:165–172, 1994.

[RR19] Noga Ron-Zewi and Ron D. Rothblum. Local proofs approaching the witness length. Cryptology ePrint Archive, Report 2019/1062, 2019.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC '16, pages 49–62, 2016.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptographic Techniques*, EUROCRYPT '97, pages 256–266, 1997.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC '08, pages 1–18, 2008.

[ZCa17] ZCash. What is Jubjub?, 2017. `https://z.cash/technology/jubjub.html`.