Electronic Colloquium on Computational Complexity, Report No. 182 (2024)

Maximum Circuit Lower Bounds for Exponential-time Arthur Merlin

Lijie Chen*
UC Berkeley
lijiechen@berkeley.edu

Jiatu Li†
MIT
jiatuli@mit.edu

Jingxun Liang
CMU
jingxunl@andrew.cmu.edu

November 17, 2024

Abstract

We show that the complexity class of exponential-time Arthur Merlin with sub-exponential advice ($\mathsf{AMEXP}_{/2^{n^\varepsilon}}$) requires circuit complexity at least $2^n/n$. Previously, the best known such near-maximum lower bounds were for symmetric exponential time by Chen, Hirahara, and Ren (STOC'24) and Li (STOC'24), or randomized exponential time with MCSP oracle and sub-exponential advice by Hirahara, Lu, and Ren (CCC'23).

Our result is proved by combining the recent iterative win-win paradigm of Chen, Lu, Oliveira, Ren, and Santhanam (FOCS'23) together with the uniform hardness-vs-randomness connection for Arthur-Merlin protocols by Shaltiel-Umans (STOC'07) and van Melkebeek-Sdroievski (CCC'23). We also provide a conceptually different proof using a novel "critical win-win" argument that extends a technique of Lu, Oliveira, and Santhanam (STOC'21).

Indeed, our circuit lower bound is a corollary of a new explicit construction for properties in coAM. We show that for every dense property $P \in \mathsf{coAM}$, there is a quasi-polynomial-time Arthur-Merlin protocol with short advice such that the following holds for infinitely many n: There exists a canonical string $w_n \in P \cap \{0,1\}^n$ so that (1) there is a strategy of Merlin such that Arthur outputs w_n with probability 1 and (2) for any strategy of Merlin, with probability 2/3, Arthur outputs either w_n or a failure symbol $\perp$. As a direct consequence of this new explicit construction, our circuit lower bound also generalizes to circuits with an $\mathsf{AM} \cap \mathsf{coAM}$ oracle. To our knowledge, this is the first unconditional lower bound against a strong non-uniform class using a hard language that is only "quantitatively harder".

*Lijie Chen is supported by a Miller Research Fellowship.
†Jiatu Li is supported by MIT Akamai Presidential Fellowship and the National Science Foundation under Grant CCF-2127597.

ISSN 1433-8092

Contents

1 Introduction

Proving circuit lower bounds for uniform complexity classes is one of the central problems in complexity theory. Despite that (following a simple counting argument) almost all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ require $2^n/n$-size circuit to compute [Sha49], the progress on proving explicit circuit lower bounds has been relatively slow.

The progress on proving *exponential* lower bounds (thereby matching Shannon's counting argument) is even more limited. Kannon [Kan82] proved that $\Sigma_3\mathsf{E} \cap \Pi_3\mathsf{E}$ requires maximum $(2^n/n)$ size circuits, the complexity of the hard function was later improved to $\Delta_3\mathsf{E} = \mathsf{E}^{\Sigma_2\mathsf{P}}$ by Miltersen, Vinodchandran, and Watanabe [MVW99], via a simple binary search argument. This was essentially all we knew before until last year.

The limited progress was due to the lack of techniques for proving exponential-size circuit lower bounds. There has been much progress on proving super-polynomial-size circuit lower bounds (see Section 2.1 for details), which all follow the famous "win-win" paradigm. However, it has been observed [MVW99] that this "win-win" paradigm could not give exponential-size lower bounds.[1]

A recent work by Chen, Hirahara, and Ren [CHR24], following a new technique called "iterative win-win paradigm" (originally developed by [CLO+23] for pseudo-deterministic construction of primes), proved that $\Sigma_2\mathsf{E}$ (as well as $\mathsf{S}_2\mathsf{E}$ with one-bit advice) requires $2^n/n$-size circuits. Their results were later simplified and strengthened by Li [Li24], showing that $\mathsf{S}_2\mathsf{E}$ (with no advice) requires maximum circuit complexity on all but finitely many input lengths. With a different approach, Hirahara, Lu, and Ren [HLR23] also proved a maximum circuit lower bound for $\mathsf{BPE}^{\mathsf{MCSP}}$ with $2^{\varepsilon n}$ bits of advice.

One surprising feature of the recent work [CHR24, Li24] is that their proofs *relativizes*. Given the limitations of relativizing proofs (for example, no relativizing proofs can prove the super-polynomial-size lower bound for MAEXP [BFT98]), a natural question is whether we can combine the techniques behinds [CHR24, Li24] (e.g., the iterative win-win paradigm) with *non-relativizing* proof techniques to make further progress on proving exponential-size circuit lower bounds.

1.1 Our Results

1.1.1 Maximum Circuit Lower Bound for AMEXP

In this work, we make progress on the question above by combining the non-relativizing techniques of *arithmetization* (specifically, the uniform hardness vs. randomness trade-off for AM [SU07, vS23]) and the iterative win-win paradigm [CLO+23, CHR24]. We show that $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$ with a sub-exponential amount of advice requires maximum circuit complexity.

Theorem 1.1. $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}} \not\subset \mathsf{SIZE}[2^n/n]$ *for any constant* $\varepsilon \in (0,1)$.

Compared with previous works [CHR24, Li24] where the same maximum circuit lower bound was proved for $\mathsf{S}_2\mathsf{E}$, our lower bound is proved for the smaller class $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$. Indeed, $\mathsf{S}_2\mathsf{E}$ is a randomized version of E^{NP}, while $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$ is a randomized version of $\mathsf{NEXP} \cap \mathsf{coNEXP}$. So in a sense, our result is much closer to NEXP than the previous one. On the other hand, our lower bound for $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$ requires a sub-exponential amount of advice, while the lower bound in [Li24] requires no advice.

[1] We note that exponential-size circuit lower bounds have more applications compared to super-polynomial-size circuit lower bounds: $2^{\Omega(n)}$-size lower bounds for E imply that $\mathsf{P} = \mathsf{BPP}$ [NW94, IW97], while super-polynomial lower bounds for E only give that BPP can be derandomized in sub-exponential time.

Moreover, our circuit lower bound not only holds for Boolean circuits, but also generalizes to circuits with an $\mathsf{AM} \cap \mathsf{coAM}$ oracle[2].

Theorem 1.2. *For any language $L \in \mathsf{AM} \cap \mathsf{coAM}$, $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}} \not\subseteq \mathsf{SIZE}^L[2^n/n]$.*

To the best of our knowledge, this is the first unconditional lower bound against a strong non-uniform class with a hard language that is only quantitatively harder (in terms of time complexity) than the non-uniform class.[3] In comparison, most of the existing unconditional lower bounds require qualitatively stronger hard languages; for instance, $\Sigma_2\mathsf{E} \not\subseteq \mathsf{SIZE}[2^n/n]$ [CHR24, Li24] requires a hard language in a high level of the exponential-time hierarchy.

This lower bound can also be interpreted as a trade-off between time and non-uniformity. It means that it is impossible to speed up an arbitrary $(\mathsf{AM} \cap \mathsf{coAM})$-style algorithm with relatively short non-uniform advice using even near-maximum non-uniform advice.

Arthur-Merlin classes. An Arthur-Merlin protocol for a language L [BM88, GS89] is a two-party constant-round interactive proof system where a computationally unbounded prover (called Merlin) aims to convince a probabilistic polynomial-time verifier (called Arthur) that $x \in L$ for a string x owned by both parties. A *strategy* of Merlin is a function that given a partial transcript of the protocol, outputs the next message to send to Arthur. The protocol should be *sound* in the sense that the verifier rejects any strategy of Merlin with high probability if $x \notin L$, and *complete* in the sense that there is a strategy of Merlin that could convince Arthur with high probability if $x \in L$ (see Section 3.2 for a formal definition).

The class $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/\alpha(n)}$ consists of languages L such that both L and $\overline{L}$ are decidable by $2^{\mathrm{poly}(n)}$-time Arthur-Merlin protocols where both parties receive an $\alpha(n)$-bit non-uniform advice on input length n.

1.1.2 Hitting Dense coAM Properties

Recent developments on maximum circuit lower bounds highlight a folklore view that proving a circuit lower bound for exponential-time classes is equivalent to designing an algorithm that explicitly constructs hard truth table [Kor22, CHR24, Li24].

More formally, consider the property Π_{hard} defined as the set of strings that are not truth tables of circuits of size at most $2^n/n$. If (for instance) there is a deterministic polynomial-time algorithm that given 1^{2^n} outputs a string $tt_n \in \Pi_{\mathsf{hard}} \cap \{0,1\}^{2^n}$ for infinitely many n, we can define L_{hard} as:

$$x \in L_{\mathsf{hard}} \iff \text{the } x\text{-th bit of } tt_{|x|} \text{ is } 1,$$

so that $L_{\mathsf{hard}} \in \mathsf{E} := \mathsf{DTIME}[2^n]$ (by calling the deterministic algorithm) and $L_{\mathsf{hard}} \notin \mathsf{SIZE}[2^n/n]$ (by the definition of Π_{hard}).

This connection can be adapted to AMEXP lower bounds with suitable technical definitions: If there is a *single-valued* Arthur-Merlin protocol (with short non-uniform advice) that given 1^{2^n} outputs a string in $\Pi_{\mathsf{hard}} \cap \{0,1\}^{2^n}$ for infinitely many n, we can obtain the lower bound in

[2] Note that $\mathsf{SIZE}^L[s(n)]$ refers to languages that admit a family of size-$s(n)$ circuits with L-oracle gates. Since L oracle gates could have unbounded fan-in, the size of the circuits is defined as the number of wires. For a concrete example, one may think of a factoring oracle, i.e., given positive integers N and k encoded in binary, it decides whether there is a divisor d of N such that $2 \le d \le k$.

[3] Here we only consider lower bounds against non-uniform classes that are at least as strong as general Boolean circuits. In restricted circuit settings, it is known (for instance) that exponential-size uniform-AC^0 requires sub-exponential-size non-uniform AC^0 circuits, which follows from the AC^0 upper and lower bound for the parity function [Ajt83, FSS84, Yao85, Has86].

Theorem 1.1. Similarly, we can obtain the lower bound in Theorem 1.2 if we replace Π_{hard} with the property Π^{L}_{hard} that contains maximally hard truth tables against L-oracle circuits. Here, a single-valued Arthur-Merlin protocol outputs a *canonical* string with high probability if Arthur does not reject during the interaction (see Section 3.2 for a formal definition).

Note that Π_{hard} and Π^{L}_{hard} are decidable in coAM. Moreover, by Shannon's counting argument [Sha49] (also see Appendix A), Π_{hard} and Π^{L}_{hard} are both *dense* properties. Indeed, both our lower bounds follow from the following general result: We show that for every dense coAM property P, there is a single-valued Arthur-Merlin protocol with short non-uniform advice that given 1^n outputs a canonical $x_n \in P \cap \{0,1\}^n$ for infinitely many $n \in \mathbb{N}$. Formally:

Theorem 1.3 (Main Theorem)**.** *Let $k > 1$ be an arbitrary constant and $P \in$ coAM be a language such that $|P_n| \geq 2^{n-1}$ for every $n \in \mathbb{N}$. There is a sequence of strings $\{x_n \in \{0,1\}^n\}_{n\in\mathbb{N}}$ and an Arthur-Merlin algorithm A that runs in time $2^{\log^{O(k)} n}$ and takes $2^{\log^{1/k} n}$ bits of advice $\{\alpha_n\}_{n\in\mathbb{N}}$ such that the following properties hold:*

- (Conformity). *For every $n \in \mathbb{N}$, there is a strategy of Merlin such that $\Pr[A(1^n, \alpha_n) = x_n] = 1$.*
- (Resiliency). *For every $n \in \mathbb{N}$ and any string $\zeta_n \in \{0,1\}^{2^{\log^{1/k} n}}$, there is a string $y_n \in \{0,1\}^n$ such that for any strategy of Merlin, $\Pr[A(1^n, \zeta_n) \in \{y_n, \bot\}] \geq 2/3$.*
- (Hitting). *For infinitely many $n \in \mathbb{N}$, $x_n \in P$.*

Here, conformity and resiliency formalize the intuition of a non-uniform single-valued Arthur-Merlin algorithm with arbitrary (i.e. possibly non-Boolean) output; see Section 3.2 for a formal definition. We also note that besides being single-valued, our Arthur-Merlin protocol enjoys an additional nice property that could be useful for other applications: The AM protocol is partially single-valued (i.e. either rejects or outputs a canonical string) even if it is given incorrect advice.

We will formally prove in Section 4 that our circuit lower bounds (see Theorems 1.1 and 1.2) follow from the main theorem. In Section 5 and Section 6, we will provide two proofs of the main theorem that are different both conceptually and technically (also see Section 2 for an overview and related discussion).

1.2 Related Works on Explicit Construction Algorithms

Our main theorem (see Theorem 1.3) is also interesting in its own right as an unconditional explicit construction algorithm for any dense property in coAM, contributing to a recent program of solving explicit construction problems using techniques from complexity theory. We provide a summary of related works from the perspective of explicit construction problems for dense properties in P, BPP, and stronger classes.

- Dense property in P: Chen et al. [CLO+23] (built on an earlier result [OS17]) proved that for any dense property Π decidable in P, there is a randomized polynomial-time algorithm that for infinitely many n, it outputs a canonical string in $\Pi \cap \{0,1\}^n$ with high probability given 1^n.[4] In particular, there is an efficient algorithm that constructs a canonical n-bit prime given 1^n for infinitely many n.[5]

[4] This is also known as a pseudodeterministic algorithm [GG11], i.e., a randomized algorithm that outputs canonical solutions with high probability.

[5] Note that primality is a dense property by the prime number theorem, and is decidable in P by the AKS primality test [AKS04].

- Dense property in BPP: Oliveira and Santhanam [OS17] proved similar pseudodeterministic algorithms exist for any dense properties decidable in BPP, but only achieves subexponential running time. Subsequently, Lu, Oliveira, and Santhanam [LOS21] constructed a polynomial-time pseudodeterministic algorithm that takes an $O(n^{\varepsilon})$-bit advice for the same problem.

- Range avoidance: Range avoidance problem [KKMP21, Kor22, RSW22] refers to the search problem that given a multi-output circuit $C : \{0,1\}^n \to \{0,1\}^m$ satisfying $m > n$, outputs a string $y \in \{0,1\}^m$ outside of the range of C, i.e., $C^{-1}(y) = \varnothing$. Deterministic (and pseudodeterministic) algorithms for range avoidance are known to imply circuit lower bounds. Chen, Hirahara, and Ren [CHR24] proved that there is a single-valued search-$\mathsf{S_2P}_{/1}$ algorithm[6] for range avoidance that works on infinitely many input lengths, which was improved by Li [Li24] to a fully uniform search-$\mathsf{S_2P}$ algorithm (i.e. avoiding the 1-bit advice) that works on all input lengths.[7]

Note that the range avoidance problem is a non-unary explicit construction problem, i.e., the input is not of form 1^n. One can also consider the unary version of it, i.e., the input circuit C is restricted to uniform family $\{C_n\}_{n\in\mathbb{N}}$ of circuits.[8] Solving unary range avoidance is just to hit the dense coNP property Π_{avoid} defined as

$$\Pi_{\mathsf{avoid}} \coloneqq \left\{ y \in \{0,1\}^n \mid n \in \mathbb{N}, C_n^{-1}(y) = \varnothing \right\}.$$

Therefore, our main theorem extends the sequence of works to the explicit construction of properties beyond (unary) range avoidance to arbitrary dense coNP properties.[9]

2 Technical Overview

In this section, we will first revisit the important conceptual ideas and technical ingredients leading to recent breakthroughs in pseudodeterministic constructions [CLO+23] and exponential circuit lower bounds [CHR24, Li24]. We will explain the *iterative win-win paradigm* in Section 2.1 introduced in [CLO+23, CHR24], which will be adapted to our setting and sketch the *first* proof of Theorem 1.3 (see Sections 2.2 and 2.3). We will then introduce an alternative technique called the *criticial win-win argument* that sketches the *second* proof of Theorem 1.3 in Section 2.4. Readers who are already familiar with the iterative win-win paradigm can skip directly to Section 2.2.

We stress that our two proofs of Theorem 1.3 are technically incomparable, and they highlight two conceptually different approaches to bypassing the half-exponential barrier (see Section 2.1 and [MVW99, CHR24]). We believe that these two techniques (or combined in some way) will lead to stronger results in circuit lower bounds and explicit construction problems.

[6] $\mathsf{S_2P}$ is a subclass of $\mathsf{ZPP}^{\mathsf{NP}} \subseteq \Sigma_2\mathsf{P}$; interested readers are referred to [CHR24] and references therein.

[7] There have also been many works on solving special cases of the range avoidance problem [RSW22, GLW22, GGNS23, CHLR23], as well matrix rigidity [AC22, BHPT24] (which is reducible to range avoidance, see [Kor22]).

[8] That is, there is a polynomial-time Turing machine outputting C_n given 1^n. Note that (pseudo-)deterministic algorithms for the unary range avoidance problem suffice to imply circuit lower bounds (see, e.g., [RSW22, CHR24]).

[9] We also note that the (non-unary) range avoidance problem is unlikely to be solvable by even non-uniform nondeterministic search algorithms [ILW23, CL24]; for this reason, it is unlikely to improve the search-$\mathsf{S_2P}$ algorithm of [CHR24, Li24] to a single-valued AM algorithm (even with advice) as otherwise one can derandomize AM using non-uniformity to obtain a non-uniform non-deterministic search algorithm for the range avoidance problem.

2.1 Bypassing the Half-exponential Barrier: the Iterative Win-win Paradigm

Before delving into our techniques of proving Theorem 1.3, we first review why a vanilla win-win argument is unable to prove exponential circuit lower bounds, and how a recent *iterative win-win paradigm* bypasses this barrier.

Win-win arguments. Win-win arguments are widely used in complexity theory to prove unconditional circuit lower bounds against general circuits. The idea goes as follows. Given an (inefficient) brute-force algorithm BF for finding a hard truth table (e.g. via diagonalization), we find a suitable problem Q and ask whether Q has large circuit complexity. If so, we obtain a circuit lower bound for Q; otherwise, we *speedup* the brute-force algorithm BF using the efficient circuit for Q and thus obtain a non-trivial algorithm for finding hard truth tables, which also leads to a circuit lower bound.

A standard example is Kannan's theorem [Kan82]. The brute-force algorithm BF is a language $L_{\mathsf{diag}} \in \Sigma_3\mathsf{E}$ that requires super-polynomial size circuits via diagonalization, and the problem Q is SAT. If $\mathsf{SAT} \notin \mathsf{P}_{/\,\mathrm{poly}}$ we already obtain a circuit lower bound for $\mathsf{NP} \subseteq \Sigma_2\mathsf{E}$; otherwise, Karp-Lipton theorem [KL80] shows that the polynomial hierarchy collapses to its second level (in particular, $\Sigma_2\mathsf{E} = \Sigma_3\mathsf{E}$), and thus $L_{\mathsf{diag}} \in \Sigma_2\mathsf{E}$. In both cases, we obtain a super-polynomial circuit lower bound for $\Sigma_2\mathsf{E}$. The lower bound can be improved to (for example) $\mathsf{S}_2\mathsf{E}$ [Cai07] using the same approach by invoking a stronger Karp-Lipton style collapse.

Indeed, a similar argument can be adapted to solve other explicit construction problems beyond circuit lower bounds, with an interesting example of pseudodeterministic constructions of large primes [OS17, LOS21].[10] Recall that a pseudodeterministic construction refers to a randomized algorithm that (given 1^n) outputs a *canonical* n-bit primes with high probability.

Instead of the Karp-Lipton collapse, [OS17] uses a reconstructive pseudorandom generator from the hardness-vs-randomness paradigm [IW01, TV07]. The brute-force algorithm BF enumerates all n-bit strings and outputs the first prime, which can be implemented in PSPACE. It then asks whether a PSPACE-complete problem L_{TV} is in BPP. If so, $\mathsf{PSPACE} = \mathsf{BPP}$ and thus BF can be implemented by a polynomial-time randomized algorithm. Otherwise, we can obtain, from the hardness of L_{TV}, a pseudorandom generator with seed length n^ε that fools uniform algorithms via the uniform hardness-vs-randomness paradigm [IW01, TV07]. Since primes are dense and the primality test is in P [AKS04], the pseudorandom generator must hit an n-bit prime, and thus we can output a canonical prime in sub-exponential time by enumerating all the seeds and outputting the first prime from the outputs of the pseudorandom generator.

The half-exponential barrier. The win-win arguments we mentioned above all run into a "half-exponential barrier", as pointed out in [MVW99] (also see [CHR24]). Intuitively, it means that the two cases in the win-win argument are competing with each other, which prevents us from proving exponential lower bounds (or polynomial-time explicit construction) in both cases.

Take Kannan's theorem as an example. Suppose that we want to prove an exponential circuit lower bound for $\Sigma_2\mathsf{E}$. If we perform a win-win argument on whether $\mathsf{SAT} \notin \mathsf{SIZE}[s(n)]$ for (say) $s(n) = 2^{n^\varepsilon}$ rather than $s(n) = n^{\omega(1)}$ to improve the lower bound when $\mathsf{SAT} \notin \mathsf{SIZE}[s(n)]$, we will encounter a sub-exponential overhead for the Karp-Lipton collapse in the case that $\mathsf{SAT} \in \mathsf{SIZE}[s(n)]$, which prevents us from proving an exponential lower bound for $\Sigma_2\mathsf{E}$. By

[10] Recall that the result of [OS17] (and the subsequent improvement from [CLO+23]) can be used to hit any dense property in P, and the construction of primes follows from the AKS primality test [AKS04] and the prime number theorem. For concreteness, we stick to the construction of large primes in the introduction.

a careful calculation of parameters, it turns out that the best we can hope is to set $s(n)$ such that $s(s(\text{poly}(n))) \le 2^n$, leading to a so-called *half-exponential* lower bound. Similarly, [OS17, LOS21] can only construct large primes psueodeterministically in half-exponential time.

Perspective: Construction of dense property and an input-length-pair-wise win-win argument. It turns out that a new interpretation of the win-win argument in [OS17, LOS21] serves as the key idea for bypassing the half-exponential barrier [CLO+23, CHR24, Li24]. Concretely:

- Both tasks above (i.e. proving circuit lower bounds and generating large primes) can be viewed as designing an efficient single-valued algorithm to hit a uniform dense property P. For the construction of primes, P is the set of primes and is decidable in P [AKS04]; for exponential circuit lower bounds, P is the set of strings that are not the truth tables of $2^n/n$-size circuits, which is known to be in coNP. This view is highlighted in recent works on the range avoidance problem, see, e.g., [Kor22, RSW22, CHLR23].[11]

- Instead of identifying a problem Q and designing two algorithms for two possible outcomes of whether Q is hard or easy (the win-win argument), we can indeed interpret the standard win-win argument as designing *one* algorithm unifying the two algorithms so that it always "wins". That is, the new algorithm always correctly outputs a canonical string in P on infinitely many input lengths.

  In more detail, let $A_{\mathsf{n}}, A_{\mathsf{m}}$ be two different algorithms, the unified algorithm considers two disjoint infinite sets of input lengths $\{n_0, n_1, \dots\}$ and $\{m_0, m_1, \dots\}$. It simulates $A_{\mathsf{n}}(1^{n_i})$ on each input length n_i ($i \in \mathbb{N}$), and simulates $A_{\mathsf{m}}(1^{m_i})$ on each input length m_i.[12] These two algorithms are designed so that for each $i \in \mathbb{N}$, either it (simulating A_{n}) is correct on the input length n_i, or it (simulating A_{m}) is correct on the input length m_i. In other words, it performs an "input-length-pair-wise" win-win argument between each pair of input lengths n_i and m_i. Indeed, this view has been found useful in proving circuit lower bounds against ACC^0 [Che24] (following [MW20]).

For concreteness, consider $n_{i+1} = 2^{n_i}$ and $m_i = 2^{n_i^{0.1}}$. Let BF be a brute-force algorithm for hitting the property P that runs in (say) exponential time.[13] Intuitively, the unified algorithm performs a win-win argument on each pair (n_i, m_i) of input lengths by considering whether the "computation history" of $\mathsf{BF}(1^{n_i})$ is "hard". It works differently according to the input length:

- On the input length m_i ($i \in \mathbb{N}$), it assumes that the "computation history" of $\mathsf{BF}(1^{n_i})$ is "hard", and utilizes this hardness to hit the dense property P (say in time $2^{O(n_i)} = 2^{m_i^{o(1)}}$) with a suitable hardness-vs-randomness framework (e.g. [TV07, NW94, IW01]).

- On the input length n_i ($i \in \mathbb{N}$), it assumes that the "computation history" of $\mathsf{BF}(1^{n_i})$ is not "hard", and tries to speed up the brute-force algorithm $\mathsf{BF}(1^{n_i})$.

If we can figure out suitable definitions of "computation history" and "hardness", and apply a hardness-vs-randomness framework accordingly, this algorithm will work on either the input length n_i or m_i for each $i \in \mathbb{N}$, unifying the two cases of the win-win argument. Indeed, one can verify that [OS17] can be interpreted as such an algorithm using the PRG in [TV07].

[11] In particular, hitting the set of strings that are not the truth tables of small circuits is reducible to the range avoidance problem [Kor22], which serves as the main technical ingredient leading to [CHR24, Li24].

[12] Recall that the explicit construction problem has a unary input. Also, note that we should define these two sets so that the algorithm can decide uniformly whether the input length is one of n_i or one of m_i.

[13] The complexity measure may not be time complexity, but (for instance) alternation in Kannan's theorem [Kan82]. We use time complexity only for illustrative purposes.

Insight: amortizing the cost of win-win. Once adapted to the input-length-pair-wise view on the win-win argument, it is natural to ask whether it is beneficial to perform a "win-win-win argument" on 3-tuples of input lengths (rather than pairs), or even play with more "wins". The rationale is that the half-exponential barrier comes with the "self-competing" nature of two possible outcomes of the win-win argument, and if we introduce more outcomes and amortize the overhead among different input lengths, we may achieve better bounds.

Iterative win-win paradigm. Indeed, the answer is positive. A framework called the *iterative win-win paradigm* was introduced in [CLO+23], which improved the half-exponential time pseudodeterministic algorithm in [OS17, LOS21] to a *polynomial-time* algorithm. Subsequently, [CHR24] proved maximum circuit lower bounds for $\Sigma_2\mathsf{E}$ and $\mathsf{S}_2\mathsf{E}_{/1}$ following the same paradigm, improving the half-exponential lower bounds of Kannan [Kan82].

As the name indicates, the iterative win-win approach performs a win-win argument iteratively with *super-constantly* many cases instead of only two cases.[14] The basic idea is as follows. Let $n_0, n_1, \dots, n_\ell$ be an increasing sequence of input lengths where n_0 is sufficiently large. We start with the brute-force algorithm BF that runs in (say) exponential-time, and ask whether its "computation history" on input length n_0 is "moderately hard".

- (*Win*). If the computation history on input length n_0 is not even "moderately hard", we can improve the brute-force algorithm BF (on input length n_0) to polynomial time.
- (*Improve*). Otherwise, we utilize the moderately hard computation history to obtain a *moderately better* algorithm on input length n_1 following a hardness-vs-randomness framework[15], treat it as the new brute-force algorithm, and proceeds the same win-win argument on input length n_1.

Note that the exact meaning of a "moderately hard" "computation history" will be clear when we instantiate the framework with [CLO+23, CHR24].

The key observation is that the improvements in the case (*Improve*) could accumulate over iterations, and thus if the sequence of input lengths $n_0, n_1, \dots, n_\ell$ grows sufficiently fast (say $n_{i+1} = n_i^\beta$ for a large constant β) and ℓ is sufficiently large (say $\ell = \log n_0$), the final "brute-force" algorithm on input length n_ℓ will be very efficient. By splitting the input lengths into infinitely many disjoint sequences $\langle n_0, n_1, \dots, n_\ell \rangle$, we can obtain a *polynomial-time algorithm* that is guaranteed to be correct on at least one n_i for each of such sequence.

Instantiations of iterative win-win. The biggest technical challenge is to identify the exact meaning of being a "moderately hard" computation history and find the hardness-vs-randomness framework allowing us to gain improvement with the hardness of the computation history.

Construction of primes. For the pseudodeterministic construction of primes [CLO+23], the "win-or-speedup" is carried out by the uniform non-black-box hardness-vs-randomness framework developed by Chen and Tell [CT22], which builds on the interactive proof system due to

[14] Note that the new perspective is crucial as it is unclear how to come up with super-constantly many cases and specify super-constantly many different algorithms in the standard win-win arguments.

[15] The intuition is that the hardness-vs-randomness framework will provide a pseudorandom generator (or hitting set generator, HSG for short) over $\{0,1\}^{n_1}$ with a non-trivial seed length from the computation history of $\mathsf{BF}(1^{n_0})$ so that the "moderately better" algorithm on the input length n_1 can enumerate all the seeds and find out a string with the property $P \cap \{0,1\}^{n_1}$.

Goldwasser, Kalai, and Rothblum [GKR15]. Intuitively, the Chen-Tell framework allows us to construct a hitting set H_C from an *inefficient parallel computation* C.[16], such that given a dense polynomial-time decidable property that H_C fails to hit, one can simulate the computation C by a randomized polynomial-time algorithm.[17] The win-win argument goes as follows.

- (*Win*). Recall that the brute-force algorithm BF_0 for finding the smallest n_0-bit prime is highly parallel, we instantiate the Chen-Tell framework with the computation $\mathsf{BF}_0(1^{n_0})$ and obtain an HSG (hitting set generator) H_0 with output length n_1. If H_0 fails to hit any n_1-bit prime, we can simulate $\mathsf{BF}_0(1^{n_0})$ by a randomized polynomial-time algorithm.
- (*Improve*). If H_0 hits an n_1-bit prime, we obtain a slightly more efficient algorithm $\mathsf{BF}_1(1^{n_1})$ by enumerating H_0 and returning the first n_1-bit prime.

Notice that the slightly more efficient algorithm BF_1 in (*Improve*) case is still highly parallel, one can iteratively perform the win-win argument as discussed above to obtain $\mathsf{BF}_2, \mathsf{BF}_3, \ldots, \mathsf{BF}_\ell$ on input lengths $n_2, n_3, \ldots, n_\ell$, where (according to the time complexity of the hitting set generator) the running time of BF_{i+1} is polynomially bounded by the running time of BF_i. This means that BF_i runs in time $\exp\{\mathrm{poly}(n_0) \cdot \exp(O(i))\}$. By setting $n_{i+1} := n_i^\beta$ for a large constant β and $\ell := \lceil \log n_0 \rceil$, then

$$n_\ell = n_0^{\beta^{\lceil \log n_0 \rceil}} \geq 2^{\beta^{\log n_0}} = 2^{n_0^{\log \beta}}$$

the running time of BF_ℓ would be

$$\exp\{\mathrm{poly}(n_0) \cdot \exp(O(\log n_0))\} = \exp(\mathrm{poly}(n_0)) \leq \mathrm{poly}(n_\ell).$$

Therefore, we can obtain a polynomial-time (pseudodeterministic) algorithm that correctly finds primes on infinitely many input lengths: Either BF_ℓ is correct on the input 1^{n_ℓ}, or for some $i < \ell$ we can "win" by simulating $\mathsf{BF}_i(1^{n_i})$ using a randomized polynomial-time algorithm.

Circuit lower bounds. Recall that proving circuit lower bounds is equivalent to an explicit construction of canonical hard truth tables (and the property consisting of hard truth tables is a dense property). For the maximum circuit lower bound for $\Sigma_2\mathsf{E}$ (and $\mathsf{S}_2\mathsf{E}$) [CHR24], the win-win argument utilizes a reduction called *hardness condensation* (see, e.g., [BS06]), which takes a truth table of length $T = 2^n$ hard against size S and constructs a truth table of length $T' = 2^{n'}$ hard against size S', where n' could be much smaller than n but $S' \approx S$.[18]

Let $n_0, n_1, \ldots, n_\ell$ be a sequence of input lengths. We also assume all n_i's are powers of 2 and let m_i be such that $n_i = 2^{m_i}$. On input length $n = 2^m$, we want to find a canonical truth table of length n that is hard against $2^m/m$ size circuits. Starting with a brute-force algorithm BF_0 that enumerates and returns the first hard truth table, we perform the following win-win argument:

- (*Win*). If the computation history of BF_0 on input length $n_0 = 2^{m_0}$ (which is of length exponential in n_0) is the truth table of a $\mathrm{poly}(n_1)$-size circuit, we can simulate the brute-force algorithm in $\Sigma_2\mathsf{P}$ (and even $\mathsf{S}_2\mathsf{P}_{/1}$, with a more careful argument) by guessing the circuit and verifying the computation ($\star$).

[16] More formally, the inefficient parallel computation is modeled as a highly uniform, large size (e.g. exponential size), and low-depth layered circuit, see [CT22, CLO+23] for a formal definition.

[17] The original Chen-Tell hitting set generator only allows a quasi-polynomial-time algorithm for simulating C, which is improved in [CLO+23] using a better pseudorandom generator from [SU05].

[18] To see that this fits into the iterative win-win paradigm we described above, one can also view hardness condenseation as a hardness-vs-randomness framework, as it solves a derandomization task of generating a hard truth table (of length $T' \ll T$) using a hard truth table (of length T).

- (*Improve*). Otherwise, the computation history of BF_0 has circuit complexity $\mathrm{poly}(n_1) \gg 2^{m_1}/m_1$. By hardness condensation ($\diamond$), we can obtain a maximally hard truth table of length $n_1 = 2^{m_1}$, which is moderately more efficient than the brute-force algorithm.

A careful inspection of a hardness condensation algorithm implicit in [Kor22] shows that it fits perfectly into the win-win argument: both the verification of its computation ($\star$) and the hardness condensation procedure ($\diamond$) can be implemented into $\Sigma_2\mathsf{P}$ and even $\mathsf{S}_2\mathsf{P}_{/1}$. Moreover, by defining the computation history carefully, we can iteratively perform the win-win argument above to obtain algorithms $\mathsf{BF}_1, \mathsf{BF}_2, \dots, \mathsf{BF}_\ell$ on input lengths $n_1, n_2, \dots, n_\ell$, where $\mathsf{BF}_\ell(1^{n_\ell})$ runs in polynomial time, and thus an algorithm that correctly finds hard truth tables on infinitely many input lengths.

A "just-win" proof of $\mathsf{S}_2\mathsf{E}$ lower bounds. In a recent paper, Li [Li24] obtained an almost-everywhere and fully uniform maximum circuit lower bound for $\mathsf{S}_2\mathsf{E}$ using an elegant and elementary proof without relying on the win-win argument. The proof is inspired by the result in [CHR24], with an additional observation that (intuitively) we will just fall into the (*Win*) case if we use a specific brute-force algorithm via the hardness condensation procedure in [Kor22] and define the computation history of it carefully.[19]

2.2 Warmup: A Win-win Argument

Can we directly apply the hardness condensation procedure [Kor22] used in [CLO+23, Li24] to obtain a circuit lower bound for AMEXP, rather than $\Sigma_2\mathsf{E}$ or $\mathsf{S}_2\mathsf{E}$? Korten's hardness condensation procedure runs in P^{NP} (see, e.g., [Kor22, RSW22]). A natural idea would be to design a better algorithm for the hardness condensation procedure (say, a single-valued Arthur-Merlin algorithm, which would give circuit lower bounds for AMEXP) for the range avoidance problem. Unfortunately, there has been evidence that such an algorithm does not exist.[20]

Again, viewing the task of proving circuit lower bounds (e.g. Theorem 1.1) as a *derandomization* problem, i.e., pseudodeterministically hitting the dense coNP property consisting of hard truth tables, brought insights on what tools we should look for. Recall that [CLO+23] performs an (iterative) win-win argument using the uniform and instance-wise Chen-Tell HSG [CT22], it is natural to ask whether we should use a similar instance-wise HSG fooling (co-)nondeterministic computation.[21]

Fortunately, a recent work by van Melkebeek and Mocelin Sdroievski [vS23] (inspired by the Chen-Tell HSG [CT22]) provides a uniform and instance-wise hardness-vs-randomness connection for AM that is suitable for our application. By combining the PCP theorem (see, e.g., [AB09]) for non-deterministic computation and a hitting set generator [SU07] with Arthur-Merlin reconstruction, they proved that:

[19] Note that both [CHR24] and [Li24] indeed prove stronger results: they designed a single-valued algorithm (in the functional version of $\mathsf{S}_2\mathsf{P}$ or $\mathsf{S}_2\mathsf{P}_{/1}$) solving the range avoidance problem (see, e.g., [Kor22, RSW22]), which is known to imply circuit lower bounds (for $\mathsf{S}_2\mathsf{E}$ or $\mathsf{S}_2\mathsf{E}_{/1}$).

[20] It is proved in [CL24] that the range avoidance problem cannot be solved by non-uniform non-deterministic search algorithms under plausible cryptographic assumptions. Since a single-valued Arthur-Merlin algorithm can be derandomized using non-uniformity by a standard argument (see, e.g., [AB09]), the range avoidance problem is also unlikely to be solvable by a single-valued Arthur-Merlin algorithm.

[21] Our Theorem 1.3 can indeed hit any dense coAM property, where $\mathsf{coNP} \subseteq \mathsf{coAM}$. This is crucial for proving Theorem 1.2 as the dense property for proving it will be a coAM property rather than a coNP property; see Section 4 for more details.

Theorem 2.1 (informal, see Theorem 5.1)**.** *There is an efficient algorithm* HSG *and an efficient Arthur-Merlin protocol* Rec *such that the following holds. Let $n, m \in \mathbb{N}$, $T \le 2^{\mathrm{poly}(n)}$, M be a time-T Turing machine, and α be a string. For every* $\mathrm{poly}(m)$*-size* coAM *circuit that rejects at most a $1/3$-fraction of its inputs, as least one of the following two conditions holds:*

- *(Hit).* $\mathsf{HSG}(n, m, M, \alpha)$ *runs in time* $\mathrm{poly}(T)$ *and outputs a multset $H \subseteq \{0,1\}^m$ such that $D(z) = 1$ for some $z \in H$.*
- *(Reconstruction). The Arthur-Merlin protocol* $\mathsf{Rec}(n, m, M, \alpha, D, x)$ *works as follows: If $M(\alpha)$ halts in time T and outputs x, there is a strategy of Merlin that makes Arthur always accept; otherwise, Arthur rejects with high probability regardless of Merlin's strategy.*

Intuitively, we treat $M(\alpha)$ (the computation of M on input α) as a potential "hard computation", and show that either we can produce a hitting set fooling a coAM circuit D, or it is indeed not a "hard computation" as $M(\alpha)$ can be simulated by a fast Arthur-Merlin protocol Rec that takes D as its input.[22]

Sub-exponential time algorithm from a win-win argument. As a warmup, we first explain how to construct a non-trivial pseudodeterministic algorithm for hitting dense coAM property using Theorem 2.1 and a vanilla win-win argument.

Let BF be the brute-force algorithm that enumerates all n-bit strings $x_1, x_2, \dots, x_{2^n}$ in lexicographic order, checks whether $x_i \in P$ in exponential time by enumerating all witnesses and outputs the lexicographically first string in the property P. We will use BF as the machine M in Theorem 2.1 to obtain a hitting set for the dense property P decidable in coAM. Let $m = m(n) = n^c$ for a sufficiently large constant c. On input length m, it plugs BF and 1^n (as M and α) into Theorem 2.1 to construct a hitting set $H \subseteq \{0,1\}^m$ in time $2^{\mathrm{poly}(n)}$. Then there will be two cases.

- *Case 1: Hitting*. Suppose that H hits the property P for infinitely many input lengths m, i.e., there is an index r such that the r-th string in H is also in P. Let r^* be the lexicographically first such r. Then the following deterministic algorithm in time $2^{\mathrm{poly}(n)}$ with $\mathrm{poly}(n)$ bits of advice r^* will hit the property P infinitely often: On input length m, it simulates $\mathsf{BF}(1^n)$, constructs the hitting set H, and outputs the r^*-th string in H. This deterministic algorithm runs in $2^{m^{0.1}}$ time and takes $m^{0.1}$ bits of advice if c is chosen to be sufficiently large.
- *Case 2: Reconstruction*. Suppose that the hitting set generator fails to hit the property P on all but finitely many input lengths m. Fix any n and $m = m(n)$. By Theorem 2.1, we can verify whether $\mathsf{BF}(1^n)$ outputs x for any $x \in \{0,1\}^n$ by the Arthur-Merlin protocol Rec, where the distinguisher D is the coAM property P. This leads to an efficient single-valued AM protocol that simulates $\mathsf{BF}(1^n)$ on input length n by letting Merlin send the correct output x of $\mathsf{BF}(1^n)$ that is the lexicographically first n-bit string in P by the definition of BF.

We can also view it in the "input-length-pair-wise" perspective: Let $n_1, n_2, \dots$ and $m_1, m_2, \dots$ be two disjoint sequences of input lengths defined as $m_i := n_i^c$, $n_{i+1} := m_i^c$, for each $i \in \mathbb{N}$, our unified algorithm simulates the algorithm in the former case on input lengths m_i, and simulates the algorithm in the latter case on input lengths n_i. By a win-win argument on each pair (n_i, m_i) of input lengths, our unified algorithm is correct on infinitely many input lengths.

[22]There are two caveats. The reconstruction protocol Rec runs in slightly super-polynomial time due to overhead in the hitting set generator [SU07]. Moreover, the version of hardness-vs-randomness connection we will need is slightly different from the one in [vS23]; concretely, we will need HSG and Rec to work not only for a fixed time bounded $T = T(n)$, but also when T (encoded in binary) is given in their inputs. Details are given in Appendix B.

A technical challenge: hardness of deciding the property. Recall that a vanilla win-win argument (including the result above) is subject to the half-exponential barrier (see Section 2.1). Can we bypass the barrier using the iterative win-win paradigm in [CLO+23]?

A key difference between our task of hitting dense coAM properties and the task of hitting dense P properties considered in [CLO+23] is that we cannot decide the coAM property we need to hit efficiently by a deterministic algorithm (unless AM = P). Recall that in the (Improve) case of the iterative win-win argument in [CLO+23], we can construct a moderately more efficient *deterministic* algorithm constructing primes by enumerating over the hitting set, *testing* their primality, and outputting the first prime in the hitting set (see Section 2.1); this ensures that the improvement in the case (Improve) could accumulate over iterations. However, in our setting, it is unclear how to construct this "moderately more efficient algorithm" without the ability to decide the property.

In this paper, we provide two different approaches to partially resolve the issue by allowing the algorithm to take a short advice (see Theorem 1.3). The first proof follows from adapting the *iterative win-win paradigm* [CLO+23] to algorithms with short advice; see Section 2.3 and 5. The second proof follows from a novel *critical win-win argument* that bypasses the half-exponential barrier without performing a win-win argument on super-constantly many cases; see Section 2.4 and 6. As far as we can tell, these two proofs are technically incomparable and interesting as they provide two conceptually different approaches to bypass the half-exponential barrier.

2.3 Proof via Iterative Win-win with Advice

Recall that in the vanilla win-win argument (see Section 2.2), the algorithm in the (Hitting) case takes short advice and runs in sub-exponential time. In this subsection, we will briefly explain how to improve the running time to quasi-polynomial by adapting the iterative win-win paradigm [CLO+23] to work with algorithms that take short advice.

Let $n_0, n_1, \dots, n_\ell$ be a sequence of input lengths. Let $\mathsf{BF}_0(1^n)$ be the brute-force algorithm that finds the lexicographically first length-n string in the property P we want to hit. Similar to the win-win argument in Section 2.2, we plug BF_0 and 1^{n_0} (as M and α) into Theorem 2.1 to construct a candidate hitting set $H_0 \subseteq \{0,1\}^{n_1}$ in time $2^{\mathrm{poly}(n_0)}$. There are two cases:

- (Win). If H_0 fails to hit the property P, i.e., H_0 is not a hitting set fooling $P \in \mathsf{coAM}$, we know by Theorem 2.1 that the reconstruction AM protocol Rec will simulate $\mathsf{BF}_0(1^{n_0})$ efficiently. (Indeed, Rec runs in quasi-polynomial time, see Theorem 5.1 for the formal statement.)
- (Improve). Otherwise, we obtain a $2^{\mathrm{poly}(n_0)}$-time deterministic algorithm BF_1 that outputs a string in $P \cap H_0$ that takes a $\mathrm{poly}(n_0)$-bit advice α_1. If we set $n_1 \gg n_0$ (e.g., $n_1 = n_0^\beta$ for a large constant β), $\mathsf{BF}_1(1^{n_1})_{\alpha_1}$ runs moderately faster than the brute-force algorithm.

The crucial observation is that in the (Improve) case, we can keep performing the win-win argument as Theorem 2.1 is "instance-wise", i.e., it allows the machine M to take an input α rather than only 1^n. For simplicity of presentation, we assume that BF_1 takes both 1^{n_1} and α_1 as its input (rather than advice). We plug BF_1 and $(1^{n_1}, \alpha_1)$ (as M and α) into Theorem 2.1 to construct a candidate hitting set $H_1 \subseteq \{0,1\}^{n_2}$ in time $(2^{\mathrm{poly}(n_0)})^{O(1)}$, and consider the two cases:

- (Win). If H_1 fails to hit the property P, i.e., H_1 is not a hitting set fooling $P \in \mathsf{coAM}$ (on the input length n_2), we know by Theorem 2.1 that the AM protocol Rec will simulate $\mathsf{BF}_1(1^{n_1}, \alpha_1)$ efficiently, where $(1^{n_1}, \alpha_1)$ is given to the AM protocol as a part of its input. Compared to the (Win) case above, we will only obtain a single-valued AM protocol hitting

P given α_1 as its advice instead of a fully uniform AM protocol; nevertheless, this suffices to prove Theorem 1.3.

- (Improve). Otherwise, we obtain a $(2^{\text{poly}(n_0)})^{O(1)}$-time deterministic algorithm BF_2 that outputs a string in $P \cap H_1$ that takes two advice strings: the advice α_1 for BF_1 to compute the hitting set generator H_1, and an advice string α_2 of length $O(\text{poly}(n_0))$ that identifies a string in $P \cap H_1$.

This win-win argument can be iteratively performed over super-constantly many input lengths. That is, for each $i \geq 1$:

- BF_i takes $\alpha_1, \ldots, \alpha_{i-1}, \alpha_i$ as advice, where $\alpha_1, \ldots, \alpha_{i-1}$ are used to simulate BF_{i-1},[23]
- $H_i \subseteq \{0,1\}^{n_{i+1}}$ is the hitting set obtained by plugging BF_i and $(1^{n_i}, \alpha_1, \ldots, \alpha_i)$ (as M and α) into Theorem 2.1.
- BF_i first obtains the hitting set H_{i-1} using BF_{i-1} and $(1^{n-1}, \alpha_1, \ldots, \alpha_{i-1})$, and uses α_i to identify a length-n_i string in $H_{i-1} \cap P$.

This will lead to Theorem 1.3 by carefully tracking the time and advice complexity of BF_i and setting the sequence $n_0, \ldots, n_\ell$ according.

2.4 Proof via Critical Win-win

We now introduce an alternative approach to speed up the derandomization algorithm in Section 2.2, which we call a *critical win-win argument*. Rather than using the hardness-vs-randomness connection in Theorem 2.1 in black-box, it exploits special properties of the specific hitting set generator [SU07] underlying Theorem 2.1, and combines it with a *strong* and *Reed-Muller-based* PCP [Par21] (also see Appendix C).

Insight: the amount of hardness. Instead of introducing more cases in win-win arguments to amortize the cost as in the iterative win-win paradigm, the critical win-win argument is inspired by the observation that we can *reduce* the cost by considering the exact "amount of hardness" we need for derandomization. This observation has been used by Lu, Oliveira, and Santhanam [LOS21] to improve the explicit construction algorithm in [OS17] and subsequently by Hirahara, Ren, and Lu [HLR23] to prove circuit lower bounds, while the idea can be dated back to the circuit lower bound for MA [San09].

We explain the idea using the result of [LOS21] as an example. Recall that in [OS17] (see Section 2.1) a sub-exponential time algorithm for generating canonical primes is constructed by performing a win-win argument on whether $\mathsf{PSPACE} = \mathsf{BPP}$. If $\mathsf{PSPACE} \neq \mathsf{BPP}$, we can construct a PRG $G^{L_m^{\mathsf{TV}}} : \{0,1\}^{\text{poly}(m)} \to \{0,1\}^n$ that is guaranteed to hit the set of primes by plugging in a PSPACE-complete language L^{TV} into the framework in [TV07]; specifically, the PRG will utilize the truth table of L^{TV} on the input length $m = n^\varepsilon$ for some constant $\varepsilon \in (0,1)$. It will take $2^{\text{poly}(m)}$ time to produce the truth table via brute force, which leads to a sub-exponential time overhead in the final algorithm to generate a canonical prime number in [OS17].

The first observation in [LOS21] is that by providing the lexicographically first seed w such that $G^{L_m^{\mathsf{TV}}}(w)$ is prime as advice, it suffices to evaluate $G^{L_m^{\mathsf{TV}}}$ on a single seed for generating a

[23] Note that in [CLO+23], the sequence of brute-force algorithms are represented by different Turing machines, and they need to track the growth of the description lengths. We introduce a trick that allows us to represent $\mathsf{BF}_0, \ldots, \mathsf{BF}_\ell$ as a single Turing machine using the recursion theorem; see Section 5.1 for more details.

canonical prime number. Moreover, the PRG used in [TV07] (which is essentially the Nisan-Wigderson PRG [NW94]) is *local* in the sense that it allows us to output $G^{L_m^{\mathsf{TV}}}(w)$ given the seed w with $\mathrm{poly}(m)$ queries to $L_m^{\mathsf{TV}} : \{0,1\}^m \to \{0,1\}$ (rather than reading the entire truth table). Therefore, we can get rid of the $2^{\mathrm{poly}(m)}$ overhead if we can compute L_m^{TV} efficiently.

Crucially, it is observed in [LOS21] that it benefits to consider the *exact amount of hardness* of L_m^{TV}. Intuitively, it is proved in [TV07] that if L_m^{TV} is hard for T^c-time probabilistic algorithms for some constant $c > 1$, then $G^{L_m^{\mathsf{TV}}} : \{0,1\}^{\mathrm{poly}(m)} \to \{0,1\}^n$ fools any T-time algorithm for *every* function T. The flexibility in the hardness-vs-randomness connection allows us to consider what is the "minimum"[24] $T^*(n)$ such that $L^{\mathsf{TV}} \in \mathsf{BPPTIME}[T^*(n)]$, which characterizes "the exact amount of hardness" of L_{TV},[25] and use both the (probabilistic time) upper and lower bound for L^{TV}. Concretely, we will set m so that $T^*(m) = \mathrm{poly}(n)$ such that:

1. The $T^*(m)$-time upper bound allows us to evaluate L^{TV} efficiently, and thus by the locality of the PRG in [TV07], we can output a canonical prime $G^{L_m^{\mathsf{TV}}}(w)$ with high probability given w as advice in probabilistic time $\mathrm{poly}(m) \cdot T^*(m) = \mathrm{poly}(n)$.

2. The (roughly $T^*(m)$ time) lower bound for L^{TV} makes sure that L_m^{TV} will hit the set of primes on the input length n (as long as $T^*(m) \ge n^\delta$ for a sufficiently large constant δ) by the hardness-vs-randomness framework in [TV07].

Therefore, by considering the exact amount of hardness and using both the upper and lower bounds, [LOS21] improved the algorithm generating a canonical prime to polynomial time.

Locality of the HSG. Recall that the time bottleneck of the algorithm in Section 2.2 appears in the "hitting" case. Suppose that the hitting set H constructed $\mathsf{BF}(1^n)$ using Theorem 2.1 hits an m-bit string in P, we need to compute the entire hitting set H within time $2^{\mathrm{poly}(n)}$ to output a canonical element in $H \cap P$, resulting in an unaffordable exponential running time in m.

Based on the first observation in [LOS21], it is natural to ask whether it is overkill to generate the entire hitting set. Fortunately, similar to the PRG used in [LOS21], the hitting set construction [SU07] underlying Theorem 2.1 is *local* in the sense that it allows us to output a *single element* in H more efficiently. Opening up the proof of Theorem 2.1 (also see Theorem 5.1), we can see that H is constructed by plugging the *computation history* of $\mathsf{BF}(1^n)$ (as a certain PCP proof) into the hitting set generator in [SU07]. Given oracle access to the computation history of $\mathsf{BF}(1^n)$ and a seed i, we can compute the i-th string in H within time $\mathrm{poly}(m)$ (see Theorem 6.1 for the formal statement). As we can store the index i in the advice, the "hitting" case can be improved to $\mathrm{poly}(m)$ time if we can implement the oracle access to the computation history of $\mathsf{BF}(1^n)$ efficiently.

Critical win-win argument. The crucial idea of our critical win-win argument is to define a suitable notion of the "amount of hardness".

Instead of identifying a language as in [LOS21], our result is based on the "input-length-pair-wise" perspective of win-win arguments (see Section 2.1). Let $H_{n,m}$ be a hitting set over m-bit strings constructed from the computation history of $\mathsf{BF}(1^n)$. We will play with three input lengths $n, m, m+1$ such that

[24] More formally, we need to find $T^*(n)$ such that $L^{\mathsf{TV}} \in \mathsf{BPPTIME}[T^*(n)] \setminus \mathsf{BPPTIME}[n^b \cdot (T^*(n))^\delta]_{/\delta \log T^*(n)}$ in [LOS21] for some constants b and δ. We will ignore the formality and say "minimum" $T^*(n)$ informally here.

[25] Note that $T^*(n) = n^{\omega(1)}$ as $\mathsf{PSPACE} \neq \mathsf{BPP}$ and L^{TV} is PSPACE-complete.

- $H_{n,m}$ hits an m-bit string in P, and
- $H_{n,m+1}$ fails to hit any $(m+1)$-bit string in P,

We will call (n,m) a *critical pair*. To gain some intuition, one may think about the case where $H_{n,t}$ hits a t-bit string in P for every $t \le m$, and fails to hit any t-bit string in P for every $t > m$; in this simplified setting, the critical pair (n,m) captures the exact amount of "hardness" of the computation history of $\mathsf{BF}(1^n)$ that can be used for derandomization (on the task of hitting P).

The crucial observation leading to the critical win-win argument is that for each critical pair (n,m), we can efficiently construct (with short advice) a canonical m-bit string in $H_{n,m} \cap P$ with a suitable hardness-vs-randomness framework. Intuitively, the algorithm can "reconstruct" the computation history of $\mathsf{BF}(1^n)$ from the failure of hitting P using $H_{n,m+1}$, and output a string in $H_{n,m} \cap P$ according to an index encoded in the advice using the *locality* of the hitting set generator.[26] Formally, we will design a single-valued Arthur-Merlin algorithm $\mathsf{Critical}_\alpha$ such that for each critical pair (n,m), $\mathsf{Critical}(1^n, 1^m)_{\alpha_m}$ constructs a canonical m-bit string in $H_{n,m} \cap P$, runs in time $2^{\mathrm{polylog}(m)}$, and takes a short advice string α_m.[27]

Assume that such an algorithm Critical exists. We consider the following three cases:

- *Case 1: Easy Reconstruction.* If there are infinite many pairs (n,m) such that $m \le 2^{\log^k n}$ and $H_{n,m}$ fails to hit P, we can instantiate the original algorithm for "the reconstruction case" in Section 2.2. Namely, we directly run the reconstruction protocol in Theorem 2.1 to simulate $\mathsf{BF}(1^n)$ and output the lexicographically first n-bit string in P.
- *Case 2: Easy Hitting.* If for some large constant β, there are infinitely many pairs (n,m) such that $m \ge 2^{n^\beta}$ and $H_{n,m}$ hits m, we can instantiate the original algorithm for "the hitting case" in Section 2.2. It naively simulates $\mathsf{BF}(1^n)$, computes the entire hitting set $H_{n,m}$, and outputs the lexicographically first string in P (according to a short advice string). The simulation of $\mathsf{BF}(1^n)$ requires $2^{n^{O(1)}} = 2^{\mathrm{polylog}(m)}$ time.
- *Case 3: Critical Hitting.* Otherwise, for all but finite many n, $H_{n,m}$ hits P for all $m \le 2^{\log^k n}$, and $H_{n,m}$ fails to hit P for some $m \le 2^{n^\beta}$. Therefore for any sufficiently large n, there is an $m \in [2^{\log^k n}, 2^{n^\beta}]$ such that $H_{n,m}$ hits P and $H_{n,m+1}$ fails to hit P, or equivalently, (n,m) forms a critical pair. This means that for infinitely many m, there is an n such that
  - $2^{\log^k n} \le m \le 2^{n^\beta}$,
  - (n,m) is a critical pair.

  On each such input length m, if we take as advice the corresponding input length n and the advice α_m for Critical on the critical pair (n,m), we can simulate $\mathsf{Critical}(1^n, 1^m)_{\alpha_m}$ and output a canonical m-bit string in $H_{n,m} \cap P$ in time $2^{\mathrm{polylog}(m+1)} = 2^{\mathrm{polylog}\, m}$.

A closer look at Theorem 2.1: How can locality help? To explain how this algorithm σ_{crit} works, we need to take a closer look at how the uniform hardness-vs-randomness connection for AM [vS23] (also see Theorem 2.1 and Appendix B) is proved. The key technical ingredient is the

[26] Here, the specific mean of "reconstruction" will be explained below when we formally describe the theorem.

[27] In the previous algorithm in Section 2.2 and the iterative win-win framework, the parameter m should be bounded by $2^{\mathrm{polylog}(n)}$ to make sure the protocol in the "reconstruction" case is efficient enough (with running time $2^{\mathrm{polylog}(m)} = 2^{\mathrm{polylog}(n)}$) for input length n. This is no longer required as we will run the reconstruction protocol on input length m instead of n in the new algorithm, which is the key to avoiding iterative win-win.

hitting set generator in [SU07] (see Theorems 3.5 and 6.1). Intuitively, it works as follows: Let p be an m-variate low-degree polynomial over $\mathbb{F} := \mathbb{F}_q$, it either generates a valid hitting set fooling coAM based on p, or (given oracle access to an coAM circuit that it fails to fool) *reconstructs* p. Here, the reconstruction of p is achieved by an AM *commit-and-evaluate* protocol which consists of two AM protocols σ_{c} and σ_{e} that informally works as follows:

- In σ_{c}, Merlin will commit to a low-degree polynomial $g_\alpha\colon \mathbb{F}^m \to \mathbb{F}$, and Arthur will output a string α that is supposed to be a commitment made by Merlin;
- In σ_{e}, Arthur (given the commitment α and an $x \in \mathbb{F}^m$) will ask Merlin to send some $y \in \mathbb{F}$, which is supposed to the evaluation $g_\alpha(x)$ of the committed polynomial.

It is ensured that if the HSG constructed from p fails, then: There is a strategy of Merlin that commits to p in σ_{c} and makes Arthur output $y = p(x)$ in σ_{e}, and for any strategy of Merlin, once it commits, the evaluation protocol will be single-valued. That is, once Merlin commits a polynomial g_α, it will have to faithfully reveal $y = g_\alpha(x)$ in $\sigma_{\mathsf{e}}(\alpha, x)$ since attempting to reveal any value other than $g_\alpha(x)$ will be rejected by Arthur with high probability; see Theorem 3.5 for a formal description.[28]

The commit-and-evaluate reconstruction protocol makes it possible to verify any $\mathsf{NTIME}[T]$ language L by an AM protocol in time $\mathrm{polylog}(T)$: We plug the (low-degree extension of) the PCP proof for an inefficient deterministic computation as the polynomial p into the HSG so that if the HSG fails, we can achieve random access to the PCP proof by a commit-and-evaluate protocol, which (together with the PCP verifier in time $\mathrm{polylog}(T)$) implies fast simulation $\mathsf{NTIME}[T]$ by an Arthur-Merlin protocol. More concretely, Arthur will ask Merlin to commit to a polynomial encoding a PCP proof, and then simulate the PCP verifier (where queries to the proof are implemented by the evaluation protocol σ_{e}).[29] In particular, we can prove Theorem 2.1 by letting L be the verification of the deterministic computation "$M(\alpha) = x$".

How can we make use of the locality property of the HSG in [SU07] to design the algorithm σ_{crit}? Suppose that (n, m) is a critical pair, i.e., $H_{n,m}$ hits P but $H_{n,m+1}$ fails to hit P, our goal is to construct a canonical string in $H_{n,m} \cap P$. A natural idea is to mimic the proof of Theorem 2.1. Recall that $H_{n,m}$ is defined as the HSG in [SU07] where p is the computation history of $\mathsf{BF}(1^n)$ in the form of a (Reed-Muller-encoded) PCP proof. Since $H_{n,m+1}$ fails to hit P, the reconstruction protocol provides oracle accesses to a polynomial that is supposed to be the (Reed-Muller-encoded) computation history of $\mathsf{BF}(1^n)$. By running the PCP verifier, Arthur can make sure that Merlin commits to a correct computation history, and thus σ_{e} provides efficient oracle access to the committed computation history. By the locality of the HSG, we can then output the i-th string in $H_{n,m}$ given i efficiently. This implies that we can output a canonical string in $H_{n,m} \cap P$ if we take an advice string i such that the i-th string in $H_{n,m}$ is in P.

A flaw, and the solution using certain PCP systems. However, there is a subtle flaw in the argument. Although Merlin cannot change the polynomial once it is committed, it does not prevent Merlin from committing to another polynomial other than the polynomial p used to generate the HSG. Therefore, if there are multiple valid computation histories (in the form of PCP proofs), Arthur will accept when Merlin commits to a polynomial encoding any of the computation histories, which makes the protocol *not* single-valued. We stress that even though

[28] For readers familiar with cryptography, this is functionally similar to a commitment scheme with local opening (e.g. the commitment scheme in Kilian's protocol [Kil92]).

[29] Again, this is very similar to the construction of Kilian's *succinct argument* scheme [Kil92].

in Theorem 2.1 we only want to simulate *deterministic* Turing machines in AM whose computation pattern is unique, there could still be multiple valid PCP proofs for verifying the computation, which may make the protocol being not single-valued.

To resolve this issue, we need to look for a suitable definition of the computation history (in the form of a PCP proof) such that it is in some sense *unique*; that is, if the prover deviates from a *canonical PCP proof*, the verifier will reject with noticeable probability. If this is possible, Merlin can only commit to the canonical PCP proof (for verifying the computation of $\mathsf{BF}(1^n)$) while running the reconstruction protocol (utilizing the failure of hitting $H_{n,m}$), and thus the final protocol will be single-valued by the correctness of the commit-and-evaluate protocol.

Fortunately, it turns out that a strong and canonical PCP with Reed-Muller-encoded proofs (see Theorem 6.2) suffices, and it can be constructed from the standard algebraic proof of the PCP theorem [BFLS91, AS98, ALM$^+$98, BS05, Par21] with certain technical modifications (see Appendix C). A PCP system for an NP relation R is said to be strong and canonical if for each input x and each witness w such that $R(x, w)$ is true, there is a canonical PCP proof $\Pi = \Pi(x, w)$ satisfying that

1. the verifier accepts with probability 1 if the proof is Π, and
2. for some constant $\alpha \in (0, 1)$, the verifier rejects with probability at least $\alpha \cdot \delta$ if the proof is δ-far from Π.

Compared to a standard PCP system, a strong and canonical PCP ensures that if a proof oracle is accepted, it is likely to be close to the unique canonical proof oracle.

We can now sketch why this will resolve the aforementioned issue. Since the proof oracle of the PCP verifier in Theorem 6.2 is a Reed-Muller code word[30], if we define the computation history as the *canonical* proof oracle in Theorem 6.2 and plug it (as the polynomial p) into the HSG in [SU07] (also see Theorems 3.5 and 6.1), the reconstruction protocol (in case that the HSG fails) ensures that if Merlin commits to a polynomial g_α different from p (i.e. the polynomial encoding the *canonical* PCP proof), either g is noticeably far from any low-degree polynomial or (by Schwartz-Zippel lemma) g is very far from p. In the former case, Arthur can detect it using standard low-degree testing (see, e.g., Theorem C.4); in the latter case, the PCP verifier will reject it with noticeable probability. Therefore, by performing these two tests, Arthur can force Merlin to commit to the desired polynomial p, and thus make the protocol single-valued.[31]

Additional technical issues. It is worth noting that to implement the idea above, we need to ensure that the (Reed-Muller encoded) proof oracle of the PCP system (see Theorem 6.2) and the low-degree polynomial p in hitting set generator [SU07] are using the same parameters (i.e. field size, degree, and the number of variables). This requires a careful inspection of both proofs and several changes to them; in particular, we need to work with a Reed-Muller-based PCP with fields of size that is *exponentially larger* than that of the usual setting. We provide a formal description of our PCP system in Theorem 6.2 and a self-contained proof in Appendix C (with an overview of all changes we made). We also provide an exposition of the hitting set generator [SU07] in Appendix D highlighting the properties to be checked and the changes to be made.

[30] Indeed, the PCP proof in [BSGH$^+$06, Par21] is not a single Reed-Muller code word but the concatenation of several Reed-Muller code words. This is a minor technical issue and we refer readers to Appendix C for more details.

[31] One may suspect that a strong and canonical PCP (i.e. not necessarily Reed-Muller encoded) should suffice, as we can either define the PCP system as the composition of it and the low-degree extension, or view the hitting set generator as the composition of it and the low-degree extension. Unfortunately, neither of the approaches works: the composition of a strong and canonical PCP and the low-degree extension may not necessarily be strong and canonical, and the composition of a local hitting set generator and the low-degree extension may not necessarily be local.

2.5 Open Problems

An immediate open question stemming from our work is whether we can reduce the length of the non-uniform advice from Theorem 1.1, or even remove it, to obtain a uniform exponential circuit lower bound for AMEXP. Note that [BFT98] showed that MAE requires half-exponential-size circuits, but it is unclear how to adapt our techniques to improve this lower bound.

Another open problem is whether we can obtain exponential lower bounds for $\mathsf{AME} = \mathsf{AMTIME}[2^{O(n)}]$ instead of AMEXP (here we allow the sub-exponential amount of advice). The main technical issue that prevents us from proving such a lower bound is that the reconstruction procedure for the hitting set generator in [SU07] has a quasi-polynomial overhead from collapsing a logarithmic-round protocol to a constant-round protocol [BM88]; see Section 2 for details.

Finally, it is interesting to understand the strength of our techniques: iterative win-win argument and critical win-win argument. Can we adapt these techniques to prove new results in complexity theory? Is there a barrier (e.g. relativization [BGS75] or algebrization [AW09]) that prevents us from proving (say) $\mathsf{EXP}_{/2^{n^\varepsilon}} \not\subset \mathsf{SIZE}[2^n/n]$ or $\mathsf{AMEXP} \not\subset \mathsf{SIZE}^{\mathsf{AM}\cap\mathsf{coAM}}[2^n/n]$ using these techniques? Note that our proof does not relativize due to the usage of the PCP theorem, but it may algebrize under a suitable definition.

Acknowledgement

We thank Hanlin Ren and Ryan Williams for helpful discussion.

3 Preliminaries

Notation. For any language L, we use L_n to denote $L \cap \{0,1\}^n$. We use $L(x)$ to denote the bit indicating whether $x \in L$, i.e., $x \in L$ if and only if $L(x) = 1$.

We say that a string $x \in \{0,1\}^n$ is δ-far from a string $y \in \{0,1\}^n$ if the relative Hamming distance between x and y is at least δ (i.e., $\Pr_{i\in[n]}[x_i \neq y_i] \geq \delta$), and a function f is δ-far from a function g if $\Pr_x[f(x) \neq g(x)] \geq \delta$. We may identify a function $f : \mathbb{F}^m \to \mathbb{F}$ (or $g : \{0,1\}^n \to \{0,1\}$) and its truth table $tt(f) \in \mathbb{F}^{|\mathbb{F}|^m}$ (or $tt(g) \in \{0,1\}^{2^n}$).

3.1 Circuits and Oracle Circuits

Throughout this paper, we define *circuits* to be fan-in two Boolean circuits where each gate can compute an arbitrary Boolean function $f : \{0,1\}^2 \to \{0,1\}$ (see [AB09, Juk12] for formal definitions). The *size* of a circuit is defined as the number of gates in the circuit.

Let $L \subseteq \{0,1\}^*$ be a language. We define *L-oracle circuits* to be Boolean circuits consisting of both fan-in two gates computing arbitrary Boolean functions $f : \{0,1\}^2 \to \{0,1\}$ and unbounded fan-in oracle gates deciding L. More precisely, a fan-in m L-oracle gate has n input wires and an output wire such that given an input $x \in \{0,1\}^m$, it outputs $L(x)$. An L-oracle circuit can contain fan-in m L-oracle gates for any m.

Note that as L-oracle circuits may have oracle gates with unbounded fan-in, the *size* of an L-oracle circuit is defined as the number of *wires* (instead of the number of *gates*) in the circuit.

3.2 Arthur-Merlin Protocols

Let $L \subseteq \{0,1\}^*$ be a language. An Arthur-Merlin protocol for L [BM88, GS89] is a two-party constant-round interactive proof $\sigma(x, P, V)$, where a computationally unbounded prover P (called Merlin) aims to convince a probabilistic polynomial-time verifier V (called Arthur) that $x \in L$ for a string x owned by both parties.

We say that a *strategy of Merlin* is the next-message function of the prover used in the protocol $\sigma = \sigma(x, P, V)$. The protocol is said to be *sound* if for every $x \notin L$ and every strategy of Merlin, Arthur rejects with probability at least $2/3$. It is said to be *complete* if for every $x \in L$, there is a strategy of Merlin such that verifier accepts with probability at least $2/3$. The soundness error can be boosted to exponentially small by running the protocol parallelly and taking a majority vote, and the completeness error can be boosted to 0 (see [AB09, Remark 8.15]). Note that one can define Arthur-Merlin protocols for promise problems rather than languages accordingly.

Given a strategy τ of Merlin, we use $\sigma^\tau(x)$ to denote the output of the protocol on the input x when the prover sends messages according to τ. Note that $\sigma^\tau(x)$ is a random variable depending on the random tape of Arthur.

AM protocols with arbitrary output. We need to define Arthur-Merlin protocols with arbitrary output, where Arthur either outputs $\perp$ (i.e. rejection) or a string.

- An Arthur-Merlin protocol with non-Boolean output is said to be *partially single-valued* (PSV) with error $\varepsilon \leq 1/3$ if for every input, there is a string y such that the verifier outputs $\perp$ or y with probability $1 - \varepsilon$ for every strategy of Merlin.[32]
- For every single-valued function $f : \{0,1\}^* \to \{0,1\}^*$, an Arthur-Merlin protocol *conforms to f* with error $\delta \leq 1/3$ if, for every input x, there is a strategy of Merlin such that Arthur outputs $f(x)$ with probability at least $1 - \delta$.
- Moreover, we say an Arthur-Merlin protocol *computes f* (with PSV error ε and conformity error δ) if it is PSV with error ε and conforms to f with error δ.

Similar to the Boolean output case (see [AB09, Remark 8.15]), the PSV error can be boosted to exponentially small by parallel repetition, and the conformity error can be boosted to 0. In case that $\varepsilon = 1/3$ (resp. $\delta = 0$), we may drop the error parameter ε (resp. δ).

Non-uniform AM protocols. Note that by round reduction results (see [BM88, GS89]), we can simulate an $O(1)$-round AM protocol by a two-round public-coin protocol (in which Arthur speaks first and sends his internal randomness) with polynomial time overhead.

Let $r = r(n) = \mathrm{poly}(n)$ be the length of Arthur's random string and $p = p(n)$ be the length of Merlin's message. We define a non-uniform AM protocol, or an AM circuit, as a Boolean circuit $A : \{0,1\}^n \times \{0,1\}^r \times \{0,1\}^p \to \{0,1\}$ where given any input x, any random string $u \in \{0,1\}^r$, and response $m \in \{0,1\}^p$ to the message u from Merlin, Arthur accepts if and only if $A(x, u, m) = 1$. We define $A(x) = 1$ if over a uniformly random $u \leftarrow \{0,1\}^r$, with probaility at least $2/3$, $A(x, u, m) = 1$ for some $m \in \{0,1\}^p$; we define $A(x) = 0$ if over a uniformly random $u \leftarrow \{0,1\}^r$, with probability at least $2/3$, $A(x, u, m) = 0$ for every $m \in \{0,1\}^p$. Note that it is possible that $A(x) \neq 0$ and $A(x) \neq 1$; we say that an AM circuit A is *total* if $A(x) \in \{0,1\}$ for every input x.

[32] Note that here we do not impose any restriction on the string y; a trivial protocol where Arthur always rejects is also considered to be PSV.

Similarly, we can define a coAM circuit as $\overline{A} : \{0,1\}^n \times \{0,1\}^r \times \{0,1\}^p \to \{0,1\}$. We say $\overline{A}(x) = 1$ if over a uniformly random $u \leftarrow \{0,1\}^r$, with probability at least 2/3, $\overline{A}(x,u,m) = 1$ for every $m \in \{0,1\}^p$; and we say $\overline{A}(x) = 0$ if over a uniformly random $u \leftarrow \{0,1\}^r$, with probability at least 2/3, $\overline{A}(x,u,m) = 0$ for some $m \in \{0,1\}^p$. It is said to be total if $\overline{A}(x) \in \{0,1\}$ for every x.

3.3 The Recursion Theorem

We say that two Turing machines N_0, N_1 are *polynomially equivalent*, denoted by $N_0 \equiv_p N_1$, if for every input x, $b \in \{0,1\}$, and $T \in \mathbb{N}$, if $N_b(x)$ halts in T steps, $N_{1-b}(x)$ halts in $\mathrm{poly}(T)$ steps and $N_0(x) = N_1(x)$. Clearly, it is an equivalence relation.

Theorem 3.1. *Let $M(\langle N \rangle, x)$ be a Turing machine whose first input is parsed as the encoding of a Turing machine N. Then there is a Turing machine Q_M such that $Q_M(x) \equiv_p M(\langle Q_M \rangle, x)$.*

Proof of Theorem 3.1. Let D be the following Turing machine: Given any input $(\langle N \rangle, x)$, where $\langle N \rangle$ is the encoding of a Turing machine N, it constructs the encoding of the Turing machine $N(\langle N \rangle, \cdot)$, and simulates $M(\langle N(\langle N \rangle, \cdot) \rangle, x)$. Let $Q_M(x) := D(\langle D \rangle, x)$. Notice that

$$\begin{aligned} M(\langle Q_M \rangle, x) &\equiv_p M(\langle D(\langle D \rangle, \cdot) \rangle, x) && \text{(Definition of } Q_M\text{)} \\ &\equiv_p D(\langle D \rangle, x) && \text{(Definition of } D\text{)} \\ &\equiv_p Q_M(x). && \text{(Definition of } Q_M\text{)} \end{aligned}$$

Note that the second equivalence also relies on the correctness and efficiency of the simulation of a Turing machine given its encoding. □

Using Theorem 3.1 we can design recursive algorithms, i.e., algorithms using the encoding of a Turing machine that is polynomially equivalent to itself. To see this, we first define $M(\langle N \rangle, x)$, where M is the algorithm and $\langle N \rangle$ is supposed to be its own encoding. Then we can apply Theorem 3.1 to obtain $Q_M(x)$. Clearly, the first input of the Turing machine $M(\langle Q_M \rangle, \cdot)$ is fixed to be the encoding of Q_M, which is polynomially equivalent to $M(\langle Q_M \rangle, \cdot)$.

Moreover, if the running time of $M(\langle N \rangle, x)$ is bounded by $T(n)$ for any fixed Turing machine N, sufficiently large n, and $x \in \{0,1\}^n$, the recursive algorithm we obtain will also run in time $T(n)$, following the definition.

3.4 Reed-Muller Code

Let q be a prime power, $d < q$, and $r \geq 1$. An r-variate degree-d Reed-Muller code over $\mathbb{F}_q$, denoted by $\mathsf{RM}_{r,d,q}$, is the set of polynomials $p : \mathbb{F}_q^r \to \mathbb{F}_q$ over $\mathbb{F}_q$ with total degree at most d. In particular, we define $\mathsf{RS}_{d,q} := \mathsf{RM}_{1,d,q}$ be the Reed-Solomon code with degree d over $\mathbb{F}_q$.

We will need the following standard encoding (i.e. low-degree extension) and local decoding algorithms for Reed-Muller code (see, e.g., [AB09]).

Lemma 3.2 (Low-degree extension)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a function, $d = n^{O(1)}$, $q = d^{O(1)}$ be prime power. There is a unique degree-d polynomial $p : \mathbb{F}_q^r \to \mathbb{F}_q$ with $r = O(n/\log d)$ variables and a polynomial-time computable encoding $I : \{0,1\}^n \to \mathbb{F}_q^r$ such that for every $x \in \{0,1\}^n$, $p(I(x)) = f(x)$.*

Moreover, there is a polynomial-time algorithm such that given f as a string of length 2^n, it outputs the polynomial p as a string of length $q^r \lceil \log q \rceil$.

Lemma 3.3 (Local decoding of Reed-Muller code)**.** *There is a probabilistic polynomial-time oracle algorithm* Dec *such that given any input $x \in \mathbb{F}_q^r$ and non-adaptive oracle accesses to a function $\hat{f} : \mathbb{F}_q^r \to \mathbb{F}_q$ that is ε-close to some $p \in \mathsf{RM}_{r,d,q}$, where $\varepsilon < (1 - d/q)/4 - 1/q$,*

$$\Pr[\mathsf{Dec}^{\hat{f}}(x) = p(x)] \geq 1 - \delta$$

for some $\delta = O(1/(\varepsilon q))$.

3.5 Verification of Computation

We will need an efficient verification of deterministic computation, which can be easily constructed from PCP theorems. For some technical reasons, we need to have a uniform verifier that works given a time bound T encoded in binary as a part of its input, instead of only for a time bound $T(n)$ that is fixed in advance. Nevertheless, this is implicitly given in proofs of the PCP theorems, see, e.g., [BSGH+06, Har04].

Theorem 3.4. *Let M be a Turing machine. There is a probabilistic polynomial-time oracle algorithm $V_M^{\mathcal{O}}$ such that the following holds. Let $x \in \{0,1\}^n$ and $T \geq n$ be a time bound encoded in binary.*

- *(Completeness). If $M(x)$ halts in T steps and accepts, then there is an oracle $\mathcal{O} : \{0,1\}^{O(\log T)} \to \{0,1\}$ such that $\Pr[V_M^{\mathcal{O}}(x,T) = 1] = 1$. Moreover, there is a deterministic algorithm* Prf *that given (x,T) such that $M(x)$ halts in T steps and accepts, outputs the truth table of an oracle O in time $\mathrm{poly}(T)$ that makes the verifier always accept.*
- *(Soundness). Otherwise, for every $\mathcal{O} : \{0,1\}^{O(\log T)} \to \{0,1\}$, $\Pr[V_M^{\mathcal{O}}(x,T) = 1] \leq 1/3$.*

The verifier tosses $O(\log T)$ random bits and makes $O(1)$ non-adaptive queries to the oracle $\mathcal{O}$.

3.6 HSG with AM Reconstruction

Now we formally describe the hitting set generator we will need in both proofs.

Theorem 3.5 ([SU07])**.** *Let r, d and h be parameters such that r is a power of d and h is a prime power. Let $q := h^{100}$ and $m := h^{1/100}$. There is an algorithm* RMV *and a pair of Arthur-Merlin protocols $(\sigma_{\mathsf{c}}, \sigma_{\mathsf{e}})$ described as follows.*

- *Let $p \in \mathsf{RM}_{r,h,q}$. $\mathsf{RMV}_{h,d}(p)$ outputs a sequence $S = (y_1, \ldots, y_s)$ of m-bit strings of size $s = q^{O(r)}$ in time $q^{O(r)}$, which is intended to be a hitting set for* coAM *circuits.*
- *σ_{c} takes a* coAM *circuit $D : \{0,1\}^m \to \{0,1\}$ as input, and outputs[33] a string $\alpha \in \{0,1\}^\ell$ called the* commitment *in time $\mathrm{poly}(|D|, \ell)$, where $\ell = O(h^{10d} \log q + h^{10}(r/d) \log q)$.*
- *σ_{e} takes $x \in \mathbb{F}_q^r$, the circuit D, and the commitment $\alpha \in \{0,1\}^\ell$ (which is intended to be generated by σ_{c}), and outputs[34] some $y \in \mathbb{F}_q$ in time $h^{O(d \log_d^2 r)}$ and $O(1)$ rounds.*

The algorithms satisfy the following properties.

[33] We note that the protocol σ_{c} itself is not necessarily PSV and may not conform to any function.
[34] Similarly, the protocol σ_{e} itself is not necessarily PSV and may not conform to any function.

- (Conformity). *If D rejects every element from $\mathsf{RMV}_{h,d}(p)$, then there is a pair of strategies $(\tau_\mathsf{c}, \tau_\mathsf{e})$ of Merlin in σ_c and σ_e such that given $x \in \mathbb{F}_q^r$,*

$$\Pr\left[\sigma_\mathsf{e}^{\tau_\mathsf{e}}(x, D, \alpha := \sigma_\mathsf{c}^{\tau_\mathsf{c}}(D)) = p(x)\right] = 1.$$

Moreover, τ_c and τ_e can be simulated by deterministic polynomial-time (in the communication complexity of σ_c and σ_e, respectively) machines given oracle accesses to p.[35]

- (Resiliency). *If D rejects at most a $1/3$-fraction of its inputs, then for any commitment $\alpha \in \{0,1\}^\ell$, there is a unique $g_\alpha \in \mathbb{F}_q^r \to \mathbb{F}_q$ such that for every $x \in \mathbb{F}_q^r$ and every strategy τ_e of Merlin,*

$$\Pr\left[\sigma_\mathsf{e}^{\tau_\mathsf{e}}(x, D, \alpha) \in \{g_\alpha(x), \bot\}\right] \geq 1 - o(1).$$

The AM protocols $(\sigma_\mathsf{c}, \sigma_\mathsf{e})$ is called a *commit-and-evaluate* protocol, where Merlin commits to a function $p \in \mathsf{RM}_{r,h,q}$ (i.e., a string encoded by Reed-Muller code) in σ_c, and Arthur could evaluate the function (i.e., locally open the string that Merlin commited to) using σ_e given the commitment of Merlin in σ_c. Theorem 3.5 ensures that if the hitting set generator using a polynomial $p \in \mathsf{RM}_{r,h,q}$ fails to hit a dense property D, then the following conditions hold.

- Merlin has a strategy to commit to p so that for every x, Arthur accepts and outputs $p(x)$ in the evaluation phase on input x.
- Once making a commitment, any attempt of Merlin to deviate from the commited polynomial in the evaluation phase will be detected by Arthur for any input x.

Note that while the hitting set generator in Theorem 3.5 only works with low-degree polynomials, it can be easily adapted to an arbitrary Boolean function via low-degree extension (see Lemma 3.2) and local decoding of Reed-Muller code (see Lemma 3.3). We will explain the details in the subsequent sub-sections.

4 Circuit Lower Bounds from Theorem 1.3

In this section, we prove our circuit lower bounds (see Theorem 1.1 and Theorem 1.2) from the main theorem (see Theorem 1.3). Recall that the main theorem provides a single-valued Arthur-Merlin algorithm for hitting dense coAM properties.

Theorem 1.3 (Restated). *Let $k > 1$ be an arbitrary constant and $P \in \mathsf{coAM}$ be a language such that $|P_n| \geq 2^{n-1}$ for every $n \in \mathbb{N}$. There is a sequence of strings $\{x_n \in \{0,1\}^n\}_{n\in\mathbb{N}}$ and an Arthur-Merlin algorithm A that runs in time $2^{\log^{O(k)} n}$ and takes $2^{\log^{1/k} n}$ bits of advice $\{\alpha_n\}_{n\in\mathbb{N}}$ such that the following properties hold:*

- (Conformity). *For every $n \in \mathbb{N}$, there is a strategy of Merlin such that $\Pr[A(1^n, \alpha_n) = x_n] = 1$.*
- (Resiliency). *For every $n \in \mathbb{N}$ and any string $\zeta_n \in \{0,1\}^{2^{\log^{1/k} n}}$, there is a string $y_n \in \{0,1\}^n$ such that for any strategy of Merlin, $\Pr[A(1^n, \zeta_n) \in \{y_n, \bot\}] \geq 2/3$.*
- (Hitting). *For infinitely many $n \in \mathbb{N}$, $x_n \in P$.*

[35] Recall that the notation σ^τ means the (probabilistic) output of the protocol when the prover sends messages according to the strategy τ.

Proof of Theorem 1.1. We first prove the circuit lower bound against deterministic circuits. This essentially follows from the folkore view of circuit lower bounds as algorithms finding hard truth tables (see, e.g., [Kor22, CHR24]). We provide a self-contained proof for completeness.

Theorem 1.1 (Restated)**.** $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}} \not\subset \mathsf{SIZE}[2^n/n]$ *for any constant* $\varepsilon \in (0,1)$.

Proof. Let $k_\varepsilon := 2\lceil \varepsilon^{-1} \rceil$. We define P^{cc} be the language as follows:

- For any string $z \in \{0,1\}^N$, let $n = \lfloor \log N \rfloor$, $z \in P^{\mathsf{cc}}$ if and only if there is no Boolean circuit $C : \{0,1\}^n \to \{0,1\}$ of size $2^n/n$ such that the length-2^n prefix of z is the truth table of C.

Clearly, $P^{\mathsf{cc}} \in \mathsf{coNP} \subseteq \mathsf{coAM}$. Moreover, by a counting argument (see [Sha49, Lup58, FM05]), we know that $|P^{\mathsf{cc}}_N| \geq 2^{N-1}$ for every $N \in \mathbb{N}$.

Let A be the Arthur-Merlin algorithm and $\{x_N \in \{0,1\}^N\}_{N\in\mathbb{N}}$ be the strings in Theorem 1.3 for $k := k_\varepsilon$ and $P := P^{\mathsf{cc}}$. By the hitting property, we know that there is an infinite sequence $\vec{N} = \{N_1, N_2, \dots\}$ of input lengths such that $x_N \in P$ for any $N \in \vec{N}$.

We define the language L^{cc} as follows. Let $n \in \mathbb{N}$.

1. If there is an $N \in \vec{N}$ such that $2^n \leq N < 2^{n+1}$, fix the smallest such $N \in \vec{N}$ and for every $z \in \{0,1\}^n$, $z \in L^{\mathsf{cc}}$ if and only if the z-th bit[36] of x_N is 1. In other words, x_N is the truth table of L^{cc} on input length n.

2. Otherwise, $L_n := \varnothing$.

We will show that $L^{\mathsf{cc}} \in (\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}} \setminus \mathsf{SIZE}[2^n/n]$.

Claim 4.1. $L^{\mathsf{cc}} \in (\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}}$.

Proof. We first describe the advice. For every $n \in \mathbb{N}$, we consider two cases:

1. If there is an $N \in \vec{N}$ such that $2^n \leq N < 2^{n+1}$, fix the smallest such $N \in \vec{N}$, the advice β_n consists of a bit $b := 1$, N (encoded in $O(\log N) = O(n)$ bits), and the advice α_N for the algorithm A in Theorem 1.3 on input length N. The total advice complexity is at most
$$1 + O(n) + 2^{\log^{1/k} N} = O(n) + 2^{(n+1)^{1/k}} \leq 2^{n^\varepsilon}.$$

2. Otherwise, the advice β_n consists of a bit $b := 0$ and a padding string 0^ℓ, where $\ell = 2^{n^\varepsilon} - 1$.

The AMEXP algorithm for L^{cc} (resp. $\overline{L^{\mathsf{cc}}}$) with advice $\{\beta_n\}_{n\in\mathbb{N}}$ works as follows. Let $n \in \mathbb{N}$ be an input length and $z \in \{0,1\}^n$ be an input. The algorithm first parses the advice as (b, N, α_N). Arthur immediately rejects (resp. accepts) if $b = 0$. Otherwise, Arthur and Merlin simulate $A(1^N, \alpha_N)$ and generate the output $x_N \in \{0,1\}^N$. Arthur accepts if and only if the z-th bit (when we identify bits of length n and numbers in $[2^n]$) of x_N is 1 (resp. is 0).

We only analyze the algorithm for L^{cc}; the analysis of the algorithm for $\overline{L^{\mathsf{cc}}}$ is similar. The algorithm is clearly sound and complete on input length n in Item 2 we discussed above. Therefore, it remains to consider the case that there is an $N \in \vec{N}$ such that $2^n \leq N < 2^{n+1}$. Fix the smallest such $N \in \vec{N}$.

- (*Completeness*). Suppose that $z \in L^{\mathsf{cc}}$, then the z-th bit of x_N is 1. By the conformity of A in Theorem 1.3, there is a strategy of Merlin such that given $(1^N, \alpha_N)$, the protocol outputs x_N with probability 1. Therefore, as long as Merlin simulates the strategy, $A(1^N, \alpha_N) = x_N$ and Arthur will accept z.

[36] Here, we identify a string of length n with an index in $[2^n]$.

- (*Soundness*). Suppose that $z \notin L^{\mathrm{cc}}$, then the z-th bit of x_N is 0. It remains to prove that for any strategy τ of Merlin in our AMEXP algorithm given the input z, Arthur will reject with probability at least $2/3$.

  Let E be the event that the simulation of $A^{\tau}(1^N, \alpha_N)$ outputs x_N or $\perp$. By the definition of L^{cc}, we know that

  $$\Pr[\text{Arthur rejects} \mid E] = 1$$

  as the z-th bit of x_N is 0. By the resiliency of A in Theorem 1.3, we also know that $\Pr[\neg E] \le 1/3$. Therefore, in our algorithm, Arthur will accept z with probability at most

  $$\Pr[\text{Arthur accepts}] \le \Pr[\text{Arthur accepts} \mid E] \cdot \Pr[E] + \Pr[\neg E] \le 1/3. \qquad \square$$

Claim 4.2. $L^{\mathrm{cc}} \notin \mathsf{SIZE}[2^n/n]$.

Proof. It follows directly from the hitting property of A in Theorem 1.3 and the definition of the language that L^{cc} requires maximum circuit complexity on input lengths in

$$\vec{n} := \{n \mid \exists N \in \vec{N},\ 2^n \le N < 2^{n+1}\}. \qquad \square$$

This completes the proof. $\square$

Proof of Theorem 1.2. Next, we generalize the lower bound to circuits with an $\mathsf{AM} \cap \mathsf{coAM}$ oracle (see Section 3.1 for the definition of oracle circuits). Since the proof is essentially the same as Theorem 1.1, we will only sketch the proof.

Theorem 1.2 (Restated)**.** *For any language* $L \in \mathsf{AM} \cap \mathsf{coAM}$, $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^{\varepsilon}}} \not\subset \mathsf{SIZE}^L[2^n/n]$.

We will need the following lemma that is similar to the standard counting argument for circuits without oracle gates [Sha49, Lup58, FM05]. The proof of the lemma is similar to the proof in [FM05]; for completeness, we provide a proof sketch in Appendix A.

Lemma 4.3. *There are at most* $2^{s(O(1)+\lceil \log(n+s) \rceil)}$ *different* L*-oracle circuits of size* s.

Proof Sketch of Theorem 1.2. Fix $L \in \mathsf{AM} \cap \mathsf{coAM}$. The only change from the proof of Theorem 1.1 is that we will define the property P^{cc} as the truth tables hard against L-oracle circuits. More formally, we will define P^{cc} as the following language:

- For any string $z \in \{0,1\}^N$, let $n = \lfloor \log N \rfloor$, $z \in P^{\mathrm{cc}}$ if and only if there is no L-oracle circuit $C^L : \{0,1\}^n \to \{0,1\}$ of size $2^n/n$ such that the length-2^n prefix of z is the truth table of C.

It remains to check that $P^{\mathrm{cc}} \in \mathsf{coAM}$ and $|P^{\mathrm{cc}}_N| \ge 2^{N-1}$ for every $N \in \mathbb{N}$. Note that $|P^{\mathrm{cc}}_N| \ge 2^{N-1}$ directly follows from Lemma 4.3, as the number of L-oracle circuits of size $s := 2^n/n$ is at most

$$2^{s(O(1)+\lceil \log(n+s) \rceil)} \le 2^{(2^n/n)\cdot(O(1)+n-\Omega(\log n))} \le 2^{2^n-1} \le 2^{N-1}.$$

To see that $P^{\mathrm{cc}} \in \mathsf{coAM}$, one need to construct a polynomial-time Arthur-Merlin protocol for $\overline{P^{\mathrm{cc}}}$, as follows. Let $x \in \{0,1\}^N$ be any input and $n = \lfloor \log N \rfloor$.

- Merlin sends an L-oracle circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most $2^n/n$ as well as a list

  $$L_u := \langle (q_1^u, b_1^u), (q_2^u, b_2^u), \ldots, (q_\ell^u, b_\ell^u) \rangle$$

  for each $u \in \{0,1\}^n$, where $\ell \le 2^n/n$ is the number of oracle gates in C. The length of q_i^u is the fan-in of the (topological) i-th oracle gate in C.

- Arthur first verifies that for each $u \in \{0,1\}^n$ and $j \in [\ell]$, if on the input u, the (topological) first $j-1$ oracle gates have inputs $q_1^u, \dots, q_{j-1}^u$ and outputs $b_1^u, \dots, b_{j-1}^u$, then the j-th oracle gate has input q_j^u.
- Recall that one can reduce the error probability of the AM protocol for L to exponentially small via parallel repetition [BM88]. Arthur and Merlin simulate the protocol for L to verify that for every $u \in \{0,1\}^n$ and $j \in [\ell]$, $L(q_j^u) = b_j^u$. Note that they can simulate all ℓ queries concurrently so that it remains a constant-round protocol.
- Arthur accepts if none of the checks above fails and $C(u) = x_u$ for every $u \in \{0,1\}^n$ when the queries and answers to the oracle gate are specified by the list L_u as we discussed above.

The soundness and completeness of the protocol are straightforward. □

5 Hitting Dense coAM Properties via Iterative Win-win

In this section, we prove the main theorem following the iterative win-win paradigm.

We will need a uniform hardness-vs-randomness for AM using the HSG in Theorem 3.5. Similar results have been obtained in [SU07] using an instance-checker for E-complete (or EXP-complete) languages. For the purpose of performing an iterative win-win argument, we will need to achieve a *smooth* tradeoff between hardness and randomness, in the sense that the HSG and the reconstruction Arthur-Merlin protocol not only work for a fixed time bound $T(n)$, but also work when both algorithms are given a time bound T in binary. Indeed, this is implicit in a recent construction due to van Melkebeek and Mocelin Sdroievski [vS23].

Theorem 5.1 (Implicit in [vS23])**.** *There is an algorithm* HSG *and an Arthur-Merlin protocol* Rec *such that the following holds. Let $n, m, T \in \mathbb{N}$ be such that $n \le m \le T$, M be a Turing machine in a standard encoding such that $|M| \le \log\log T$, α be a string of length at most m, and $D : \{0,1\}^m \to \{0,1\}$ be a* $\mathrm{poly}(m)$*-size* coAM *circuit that rejects at most a $1/3$-fraction of its inputs. Then* $\mathsf{HSG}(n, m, T, M, \alpha)$ *runs in time* $\mathrm{poly}(T)$ *and outputs a multiset $H \subseteq \{0,1\}^m$ of size* $\mathrm{poly}(T)$ *such that one of the following two conditions holds.*

- *(**Hit**). There exists a $z \in H$ such that $D(z) = 1$.*
- *(**Reconstruct**). The Arthur-Merlin protocol* $\mathsf{Rec}(n, m, T, M, \alpha, D, x)$ *runs in $m^{O((\log\log T)^2)}$ time and has $O(1)$ rounds such that the following holds:*
  - *(Completeness). If $M(\alpha)$ halts in time T and outputs $x \in \{0,1\}^n$, there is a strategy of the prover such that the verifier accepts with probability* 1*.*
  - *(Soundness). Otherwise, for any strategy of the prover, the verifier rejects with probability at least* $1/2$*.*

For completeness, we provide a self-contained proof in Appendix B.

5.1 Proof of Theorem 1.3

Now we are ready to formally prove Theorem 1.3.

Theorem 1.3 (Restated). *Let $k > 1$ be an arbitrary constant and $P \in \mathsf{coAM}$ be a language such that $|P_n| \geq 2^{n-1}$ for every $n \in \mathbb{N}$. There is a sequence of strings $\{x_n \in \{0,1\}^n\}_{n\in\mathbb{N}}$ and an Arthur-Merlin algorithm A that runs in time $2^{\log^{O(k)} n}$ and takes $2^{\log^{1/k} n}$ bits of advice $\{\alpha_n\}_{n\in\mathbb{N}}$ such that the following properties hold:*

- (Conformity). *For every $n \in \mathbb{N}$, there is a strategy of Merlin such that $\Pr[A(1^n, \alpha_n) = x_n] = 1$.*
- (Resiliency). *For every $n \in \mathbb{N}$ and any string $\zeta_n \in \{0,1\}^{2^{\log^{1/k} n}}$, there is a string $y_n \in \{0,1\}^n$ such that for any strategy of Merlin, $\Pr[A(1^n, \zeta_n) \in \{y_n, \bot\}] \geq 2/3$.*
- (Hitting). *For infinitely many $n \in \mathbb{N}$, $x_n \in P$.*

Proof of Theorem 1.3. Let n_0 be sufficiently large. We define $n_0^{(0)} := n_0$ and for every $t \geq 1$,

$$n_0^{(t)} := 2^{2^{n_0^{(t-1)}}}.$$

Let $\beta = O(k)$ be a large constant to be determined later. For each $t \geq 0$, we define the sequence $\vec{n}^{(t)} = (n_1^{(t)}, n_2^{(t)}, \dots, n_\ell^{(t)})$ by

$$n_{i+1}^{(t)} := 2^{\log^\beta(n_i^{(t)})},$$

where $\ell = \ell(n_0^{(t)}) = \lceil \log\log(n_0^{(t)}) \rceil$. Notice that

$$n_\ell^{(t)} = 2^{\log^{\beta^\ell}(n_0^{(t)})} = 2^{2^{\mathrm{polylog}(n_0^{(t)})}} \ll n_0^{(t+1)}$$

for sufficiently large n_0, and therefore $\vec{n}^{(0)}, \vec{n}^{(1)}, \vec{n}^{(2)}, \dots$ are disjoint sequences of increasing numbers. We will describe the behavior of our algorithm on input lengths $\vec{n}_\ell^{(t)}$, and prove that for every $t \in \mathbb{N}$, there is an $i \leq \ell(n_0^{(t)})$ such that our algorithm correctly hits a canonical string of length $n_i^{(t)}$. Since our algorithm are uniform over all $t \in \mathbb{N}$, we will fix a $t \in \mathbb{N}$ and omit the superscript (t) in the rest of the proof if there is no ambiguity.

Algorithm BF_i and HSG H_i. For $i \in \{0, 1, \dots, \ell\}$, we define an algorithm BF_i that takes an advice α_i of length at most $c \cdot i \cdot n_0^c$ for some fixed constant c, a time bound T_i, and a hitting set H_i constructed from the computation history of $\mathsf{BF}_i(1^{n_i})_{\alpha_i}$. Note that although we define $\mathsf{BF}_0, \dots, \mathsf{BF}_\ell$ as different algorithms for simplicity, it will be guaranteed that it can be implemented by a single Turing machine; in other words, there is a Turing machine BF such that for every $i \in [\ell]$ and any advice α_i, $\mathsf{BF}(1^{n_i}, \alpha_i)$ and $\mathsf{BF}_i(1^{n_i})_{\alpha_i}$ output the same answer in the same time complexity up to a polynomial overhead.

- $\mathsf{BF}_0(1^{n_0})$ enumerates all possible strings of length n_0 and outputs the lexicographic first string $x \in P_{n_0}$. Since $P \in \mathsf{coAM} \subseteq \mathsf{EXP}$, $\mathsf{BF}_0(1^{n_0})$ runs in time $2^{n_0^{c'}}$ for a fixed constant c'.
- For every $i \in \{0, 1, \dots, \ell\}$, we define $H_i \subseteq \{0,1\}^{n_{i+1}}$ be the hitting set generator constructed from the computation history of $\mathsf{BF}_i(1^{n_i})_{\alpha_i}$. Concretely, let BF be the uniform Turing machine as mentioned above, H_i is defined as the multiset generated by

$$\mathsf{HSG}(n_i, n_{i+1}, T_i, \mathsf{BF}, (1^{n_i}, \alpha_i)), \tag{1}$$

  where HSG is the algorithm in Theorem 5.1. (Note that Theorem 5.1 requires the description length of the Turing machine $|\mathsf{BF}| \leq \log\log T_i$. Indeed, since BF will be a single Turing machine, we will have $|\mathsf{BF}| = O(1) \ll \log\log T_i$.)

- For every $i \in [\ell]$, let $j_i \in [|H_{i-1}|]$ be the smallest index such that the j_i-th string in H_{i-1} has the property P_{n_i}, and j_i can be fixed arbitrarily if there is no such index, e.g., $j_i := 1$. Let $\alpha_i := (j_1, j_2, \ldots, j_i)$ be the advice to BF_i. The algorithm $\mathsf{BF}_i(1^{n_i})_{\alpha_i}$ constructs H_{i-1} and outputs the j_i-th string in H_{i-1}. Note that in the definition of algorithm $\mathsf{BF}_i(1^{n_i})_{\alpha_i}$ (i.e. $\mathsf{BF}(1^{n_i}, \alpha_i)$ we need to know the description of BF in order to evaluate HSG in Equation (1); this can be done using a standard trick in the proof of the recursion theorem, see our remark below for (∇).

- For every $i \in \{0, 1, \ldots, \ell\}$, we define $T_i := 2^{n_0^{c'} \cdot c^{i+1}} + n_i^c$, where c is a sufficiently large constant to be determined later. For every $i \in [\ell]$, the advice complexity of BF_i is

$$\sum_{j=1}^{i} O(\log |H_j|) = \sum_{j=0}^{i-1} O(\log T_j) \leq O(\log n_i) + O(n_0^{c'} \cdot c^{i+2}) \leq n_0^{O(1)}. \qquad (2)$$

Remark on (∇). Now we formally describe how to define the Turing machine BF that uses its own code. We first define a Turing machine $M(\langle N \rangle, (1^n, \alpha))$ whose first input is the code of a Turing machine N. It simulates the algorithm $\mathsf{BF}(1^n, \alpha)$ described above in the following sense: whenever we need the code of BF, we plugin $\langle N \rangle$. By the recursion theorem (see Theorem 3.1), there is a Turing machine Q_M such that for every input $(1^n, \alpha)$, $Q_M(1^n, \alpha) = M(\langle Q_M \rangle, (1^n, \alpha))$, and moreover, if $M(\langle Q_M \rangle, (1^n, \alpha))$ halts in T steps, $Q_M(1^n, \alpha)$ halts in dT^d steps for some absolute constant d. We define $\mathsf{BF}(1^n, \alpha) = Q_M(1^n, \alpha)$.

Time Complexity. Recall that $\mathsf{BF}_i(1^{n_i}) := \mathsf{BF}(1^{n_i}, \alpha_i)$, and we want to ensure $\mathsf{BF}_i(1^{n_i})_{\alpha_i}$ runs in time T_i, where

$$T_i := 2^{n_0^{c'} \cdot c^{i+1}} + n_i^c.$$

Note that this holds for $i = 0$: $M(\langle \mathsf{BF} \rangle, 1^{n_0})$ is simply the brute-force algorithm running in time $2^{n_0^{c'}}$ which ignores the code of BF, and by Theorem 3.1 we know that $\mathsf{BF}(1^{n_0})$ runs in time $d2^{n_0^{c'} d} \leq T_0$ if c is chosen to be sufficiently large.

Assume that $\mathsf{BF}_i(1^{n_i}, \alpha_i)$ runs in time T_i. Then by the construction of M we know that $M(\langle \mathsf{BF} \rangle, (1^{n_{i+1}}, \alpha_{i+1}))$ runs in $T_i^{O(1)}$ time, where the polynomial overhead is from Theorem 5.1. Therefore, $\mathsf{BF}(1^{n_{i+1}}, \alpha_{i+1})$ runs in time $dT_i^{O(d)} \leq T_{i+1}$ for a sufficient large constant c.

Note that since $\ell = \lceil \log \log n_0 \rceil$, $\mathsf{BF}_\ell(1^{n_\ell})_{\alpha_\ell}$ runs in time

$$T_\ell = 2^{n_0^{c'} \cdot c^{\ell+1}} + n_\ell^c.$$

Recall that $\log(n_{i+1}) = \log^\beta(n_i)$ and thus (for sufficiently large β)

$$n_\ell = 2^{\log^{\beta^\ell} n_0} \geq 2^{2^{\log^2 n_0}} \geq 2^{n_0^c \cdot c^{\ell+1}},$$

we can see that $T_\ell \leq n_\ell^{O(1)}$. Therefore we will win (for this fixed t) if $H_{\ell-1}$ contains an element with property P_{n_ℓ}.

Algorithm Rec_i. Now we consider the case that $\mathsf{BF}_\ell(1^{n_\ell})_{\alpha_\ell} \notin P_{n_\ell}$. Note that $\mathsf{BF}_0(1^{n_0}) \in P_{n_0}$ as it is the brute-force algorithm. Therefore, there is an $i < \ell$ such that $\mathsf{BF}_i(1^{n_i})_{\alpha_i} \in P_{n_i}$ but

$\mathsf{BF}_0(1^{n_{i+1}})_{\alpha_{i+1}} \notin P_{n_{i+1}}$, and in such case, $H_i \subseteq \{0,1\}^{n_{i+1}}$ must fail to hit any string in $P_{n_{i+1}}$. Recall that

$$H_i := \mathsf{HSG}(n_i, n_{i+1}, T_i, \mathsf{BF}, (1^{n_i}, \alpha_i))$$

where HSG is the algorithm in Theorem 5.1. We then know by Theorem 5.1 that there is an Arthur-Merlin protocol such that given $(n_i, n_{i+1}, T_i, \mathsf{BF}, (1^{n_i}, \alpha_i), P, x)$, Arthur accepts honest Merlin if $\mathsf{BF}(1^{n_{i+1}}, \alpha_i)$ halts in T_i steps and outputs x, and rejects any dishonest Merlin with probability at least $2/3$ otherwise. (Note that $\mathsf{BF}(1^{n_{i+1}}, (1^{n_i}, \alpha_i))$ indeed runs in time T_i, as we discussed above.)

We define $\mathsf{Rec}(1^{n_i}, \alpha_i)$ be the following Arthur-Merlin protocol: Merlin sends $x \in \{0,1\}^{n_i}$ and they simulates the protocol above on $(n_i, n_{i+1}, T_i, \mathsf{BF}, (1^{n_i}, \alpha_i), P, x)$; Arthur rejects if Arthur (in the protocol above) rejects, and outputs x if Arthur (in the protocol above) accepts. Recall that Rec in Theorem 5.1 runs in time

$$n_{i+1}^{O((\log\log T_i)^2)} \leq 2^{O(\log^\beta n_i (\log\log T_i)^2)} \leq 2^{\log^{O(k)} n_i}$$

and has $O(1)$ rounds. The advice complexity of Rec_i is $n_0^{O(1)}$ (see Equation (2)), which is bounded by

$$n_{i-1}^{O(1)} \leq 2^{\log^{O(1/\beta)} n_i} \leq 2^{\log^{1/(10k)} n_i}.$$

for sufficiently large $\beta = O(k)$.

Iterative Win-Win. Now we describe our final algorithm. Fix any $t \geq 0$, we define $b_i = b_i^{(t)} \in \{0,1\}$ as

$$b_i := \begin{cases} 1 & i = \ell \text{ and } \mathsf{BF}_\ell(1^{n_i})_{\alpha_i} \in P_{n_\ell}, \\ 1 & i < \ell \text{ and } H_i \text{ fails to hit } P_{n_{i+1}}, \\ 0 & \text{otherwise.} \end{cases}$$

Our algorithm will take an advice (α_i, b_i) on input length n_i. The canonical output of our algorithm x_n is defined as

$$x_n := \begin{cases} 0^n & n \neq n_i^{(t)} \text{ for all } (t,i) \\ 0^n & n = n_i^{(t)} \text{ and } b_i^{(t)} = 0 \\ \mathsf{BF}_i(1^{n_i^{(t)}})_{\alpha_i^{(t)}} & n = n_i^{(t)} \text{ and } b_i^{(t)} = 1 \end{cases}$$

The algorithm σ_P is describe in Algorithm 1.

Note that we will choose parameter $\beta = O(k)$ to be sufficiently large. Since both BF_ℓ and Rec_i run in time at most $2^{\log^{O(k)} n}$ and have advice complexity at most $2^{\log^{1/(10k)} n}$, we know that the protocol σ_P (see Algorithm 1) runs in time $2^{\log^{O(k)} n}$ and have advice complexity at most $2^{\log^{1/k} n}$. It remains to check the conformity, resiliency, and hitting properties of the protocol.

Claim 5.2 (Conformity)**.** *For every input length n, there is a strategy for Merlin such that $\sigma_P(1^n) = x_n$ with probability* 1.

Proof. If $n \neq n_i^{(t)}$ for every $t \geq 0$ and $i \leq \ell(n_0^{(t)})$, or $n = n_i^{(t)}$ but $b_i^{(t)} = 0$, Arthur will output $0^n = x_n$ without any interaction with Merlin. Now fix any $t \geq 0$, consider the case that $n = n_i^{(t)}$ and $b_i^{(t)} = 1$. If $i = \ell$, Arthur will output $\mathsf{BF}(1^{n_i})_\alpha = x_n$. Otherwise, Arthur and Merlin will simulate $\mathsf{Rec}(1^{n_i}, \alpha_i)$. By the definition of b_i, we know that H_i fails to hit the property P on input length n_{i+1}, and therefore by Theorem 5.1, there is a strategy of Merlin in $\mathsf{Rec}(1^{n_i}, \alpha)$ such that Arthur accepts and outputs $\mathsf{BF}_i(1^{n_i})_{\alpha_i} = x_n$ with probability 1. ☐

Algorithm 1: Arthur-Merlin protocol σ_P for Theorem 1.3

Input 1^n *and an advice* (α_i, b_i) *of length at most* $2^{\log^{1/k} n}$ *as discussed above*
Output $\perp$ *or* $x \in \{0,1\}^n$
If $n \neq n_i^{(t)}$ for any $t \geq 0$ and $i \leq \ell(n_0^{(t)})$, output 0^n and halt;
If $b_i = 0$, output 0^n and halt;
Let $t \geq 0$, $i \leq \ell(n_0^{(t)})$ such that $n = n_i^{(t)}$. (We ignore the superscript (t) from now on.)
if $i = \ell$ **then**
    Simulate $\mathsf{BF}(1^{n_\ell}, \alpha_\ell)$;
else
    Simulate $\mathsf{Rec}(1^{n_i}, \alpha_i)$;

Claim 5.3 (Resiliency)**.** *For every input length n, and any advice ζ_n, there is a string $y_n \in \{0,1\}^n$ such that for any strategy of Merlin,* $\Pr[\sigma_P(1^n, \zeta_n) \in \{y_n, \perp\}] \geq 3/5$.

Proof. Fix any n and $\zeta = (\hat{\alpha}, \hat{b})$. If $\hat{b} = 0$ or $n \neq n_i^{(t)}$ for every $t \geq 0$ and $i \leq \ell(n_0^{(t)})$, Arthur will output 0^n without any interaction with Merlin, and thus we can define $y_n := 0^n$. Now fix any $t \geq 0$ and consider the case that $n = n_i^{(t)}$ and $\hat{b} = 1$. If $i = \ell$, Arthur will simulate $\mathsf{BF}(1^n, \hat{\alpha})$ without any interaction with Merlin, and thus we can define $y_n := \mathsf{BF}(1^n, \hat{\alpha})$. Otherwise, Arthur and Merlin will simulate $\mathsf{Rec}(1^n, \hat{\alpha})$. By Theorem 5.1, we know that for any strategy of Merlin, $\mathsf{Rec}(1^n, \hat{\alpha})$ will output either $\perp$ or $\mathsf{BF}(1^n, \hat{\alpha})$ with probability at least 3/5, and therefore we can define $y_n := \mathsf{BF}(1^n, \hat{\alpha})$. ☐

Claim 5.4 (Hitting)**.** *For every $t \geq 0$, there is an $i \leq \ell(n_0^{(t)})$ such that $x_{n_i^{(t)}} \in P$.*

Proof. Fix any $t \geq 0$. If $\mathsf{BF}_\ell(1^{n_\ell})_{\alpha_\ell} \in P$, we know that $\sigma_P(1^{n_\ell}, (\alpha_\ell, b)) =: x_{n_i} \in P$. Otherwise, since $\mathsf{BF}_0(1^{n_0}) \in P$, there must be an $i < \ell$ such that $\mathsf{BF}_i(1^{n_i})_{\alpha_i} \in P$ but $\mathsf{BF}_{i+1}(1^{n_{i+1}})_{\alpha_i} \notin P$. By the definition of α_i we know that H_i must fail to hit the property P on input length n_{i+1}. In that case, we will have $\mathsf{BF}_i(1^{n_i})_{\alpha_i} := x_{n_i} \in P$. ☐

Note that one can reduce the resiliency error (see Claim 5.3) to 1/3 by repetition. This completes the proof. ☐

6 Hitting Dense coAM Properties via Critical Win-win

In this section, we provide an alternative proof of the main theorem using a novel *critical win-win argument*. We will first introduce two technical ingredients, namely a local hitting set generator implicit in [SU07] and a strong PCP verifier from Reed-Muller Code, in Section 6.1 and Section 6.2. Then we will explain critical win-win argument and prove Theorem 1.3 in Section 6.3.

6.1 Local Hitting Set Generator

A key observation is that the RMV hitting set generator (see Section 3.6) is *local*, in the sense that there is an efficient oracle algorithm that given an index i, outputs the i-th string in the HSG with oracle accesses to the Reed-Muller encoded function f.

Theorem 6.1 (Local HSG with arbitrary field size, implicit in [SU07])**.** *Let r, d and h be parameters such that r is a power of d and h is a prime power. Suppose $d = O(1)$ and $h = \mathrm{poly}(r)$. Let q be a prime power with $h^{100} \le q \le 2^{h^{O(1)}}$ and h be a parameter with $h^{1/100} \le m \le q^{1/100}$. There is an algorithm* RMV *and a pair of Arthur-Merlin protocols $(\sigma_\mathsf{c}, \sigma_\mathsf{e})$ described as follows.*

- (Locality). *Let $p \in \mathsf{RM}_{r,h,q}$. There is an oracle algorithm $\mathsf{RMV}_{h,d}$ that takes a seed $z \in \{0,1\}^{O(r \log q)}$ and p as oracle, outputs a string in $\{0,1\}^m$ in time $\mathrm{poly}(m)$. The collection of all $\mathsf{RMV}^p_{h,d}(z)$ is intended to be a hitting set for* coAM *circuits.*
- *σ_c takes a* coAM *circuit $D : \{0,1\}^m \to \{0,1\}$ as input, and outputs a string $\alpha \in \{0,1\}^\ell$ called the* commitment *in time $\mathrm{poly}(|D|, \ell)$, where $\ell = \mathrm{poly}(m)$.*
- *σ_e takes $x \in \mathbb{F}_q^r$, the circuit D, and the commitment $\alpha \in \{0,1\}^\ell$ (which is intended to be generated by σ_c), and outputs some $y \in \mathbb{F}_q$ in time $m^{O(d \log_d^2 r)}$ and $O(1)$ rounds.*

The algorithms satisfy the following properties.

- (Conformity). *If D rejects every element from $\mathsf{RMV}_{h,d}(p)$, then there is a pair of strategies $(\tau_\mathsf{c}, \tau_\mathsf{e})$ of Merlin in σ_c and σ_e such that given $x \in \mathbb{F}_q^r$,*

$$\Pr\left[\sigma_\mathsf{e}^{\tau_\mathsf{e}}(x, D, \alpha := \sigma_\mathsf{c}^{\tau_\mathsf{c}}(D)) = p(x)\right] = 1.$$

  Moreover, τ_c and τ_e can be simulated by deterministic polynomial-time (in the communication complexity of σ_c and σ_e, respectively) machines given oracle accesses to p.[37]
- (Resiliency). *If D rejects at most a $1/3$-fraction of its inputs, then for any commitment $\alpha \in \{0,1\}^\ell$, there is a unique $g_\alpha \in \mathbb{F}_q^r \to \mathbb{F}_q$ such that for every $x \in \mathbb{F}_q^r$ and every strategy τ_e of Merlin,*

$$\Pr\left[\sigma_\mathsf{e}^{\tau_\mathsf{e}}(x, D, \alpha) \in \{g_\alpha(x), \perp\}\right] \ge 1 - o(1).$$

The theorem can be obtained from making a few straightforward modifications on the construction from [SU07], we include a proof sketch in Appendix D.

6.2 Strong PCP from Reed-Muller Code

Instead of using the standard PCP system (see Theorem 3.4) in the iterative win-win argument, we will need the a proof system with additional properties that can be composed with the local hitting set generator.

Theorem 6.2. *There is a constant $\alpha \in (0,1)$ such that for any Turing machine M, there is a constant $c \ge 1$ and a probabilistic polynomial-time oracle verifier $V_M^{\mathcal{O}}$ satisfying the following. Let $x \in \{0,1\}^n$, $T \ge n$ be a time bound encoded in binary, $r, h \ge 1$ and q be a power of a prime $p = O(1)$, such that $r = \Theta(\log T / \log\log T)$, $h \ge n^c \cdot T^{c/r}$, $h^c \le q \le T$.*

- *Given input (x, T, r, h, q) to the verifier $V_M^{\mathcal{O}}$, the proof oracle $\mathcal{O}$ is supposed to be a sequence of polynomials $f_1, f_2, \ldots, f_{6r+8} \in \mathsf{RM}_{3r+3,h,q}$. The verifier tosses $O((r+h)\log q)$ random coins, generates $k = O(rh)$ non-adaptive queries $(i_1, x_1), (i_2, x_2), \ldots, (i_k, x_k) \in [6r+8] \times \mathbb{F}_q^{3r+3}$, and decides in $\mathrm{poly}(r, h, \log q)$ time whether to accept the proof given answers $f_{i_1}(x_1), f_{i_2}(x_2), \ldots, f_{i_k}(x_k) \in \mathbb{F}_q$.*

[37] Recall that the notation σ^τ means the (probabilistic) output of the protocol when the prover sends messages according to the strategy τ.

- *(Completeness). If $M(x)$ halts in T steps and accepts, there is a* unique *oracle $\mathcal{O}^*$ such that $\Pr[V_M^{\mathcal{O}^*}(x,T,r,h,q)=1]=1$. We call this oracle $\mathcal{O}^*=(f_1^*,f_2^*,\dots,f_{6r+8}^*)$ the* canonical proof *corresponding to the input (x,T,r,h,q).*

- *(Soundness). If $M(x)$ does not halt in T steps, or $M(x)$ rejects, then for every oracle $\mathcal{O}=(f_1,f_2,\dots,f_{6r+8})$, $\Pr[V_M^{\mathcal{O}}(x,T,r,h,q)=1]\le 1-\alpha$.*

- *(Strong soundness). If $M(x)$ halts in T steps and accepts, then for every oracle $\mathcal{O}=(f_1,f_2,\dots,f_{6r+8})$, if for any constant $\delta\in(0,1)$, f_i is δ-far from the i-th polynomial f_i^* in the canonical proof for some $i\in[6r+8]$, then*
$$\Pr[V_M^{\mathcal{O}}(x,T,r,h,q)=1]\le 1-\alpha\cdot\delta,$$
where $\alpha\in(0,1)$ is a universal constant.

The proof system is implicit in the standard algebraic proof of the PCP theorem (see, e.g., [BSGH+06] and [Par21] for the strong soundness) with some minor technical modification. For completeness, we provide a self-contained proof of the theorem as well as a summary of the changes we need to make in Appendix C.

6.3 Pseudodeterministic Construction with Local HSG

Theorem 1.3 (Restated). *Let $k>1$ be an arbitrary constant and $P\in\mathsf{coAM}$ be a language such that $|P_n|\ge 2^{n-1}$ for every $n\in\mathbb{N}$. There is a sequence of strings $\{x_n\in\{0,1\}^n\}_{n\in\mathbb{N}}$ and an Arthur-Merlin algorithm A that runs in time $2^{\log^{O(k)} n}$ and takes $2^{\log^{1/k} n}$ bits of advice $\{\alpha_n\}_{n\in\mathbb{N}}$ such that the following properties hold:*

- (Conformity). *For every $n\in\mathbb{N}$, there is a strategy of Merlin such that $\Pr[A(1^n,\alpha_n)=x_n]=1$.*

- (Resiliency). *For every $n\in\mathbb{N}$ and any string $\zeta_n\in\{0,1\}^{2^{\log^{1/k} n}}$, there is a string $y_n\in\{0,1\}^n$ such that for any strategy of Merlin, $\Pr[A(1^n,\zeta_n)\in\{y_n,\bot\}]\ge 2/3$.*

- (Hitting). *For infinitely many $n\in\mathbb{N}$, $x_n\in P$.*

Proof. The proof starts by constructing a hitting set $H_{n,m}$ for each pair of parameters (n,m) with $n\le m$:

- Let the Turing machine BF be the brute-force algorithm to hit the property P: On any input length n, $\mathsf{BF}(1^n)$ enumerates all the possible strings x of length n, checks whether $x\in P$ in exponential time (as $P\in\mathsf{coAM}\subseteq\mathsf{EXP}$) and outputs the lexicographic first string in P. The running time of $\mathsf{BF}(1^n)$ is bounded by $T(n):=2^{n^{O(1)}}$.

- Let the Turing machine $\mathsf{BF}_{\mathsf{dec}}$ be the decision version of BF: On any input $(1^n,w)$, $\mathsf{BF}_{\mathsf{dec}}(1^n,w)$ checks whether w is the output of $\mathsf{BF}(1^n)$.

- Let r,h be parameters with $r:=\Theta(\log T(n)/\log\log T(n))$ and $h:=n^c\cdot T(n)^{c/r}$ (where c is the constant defined in Theorem 6.2), and let $q:=2^{100n^\beta}$ where β is a constant to be determined in Lemma 6.3. It's easy to check both r and h are polynomials in n. By Theorem 6.2, the Turing machine $\mathsf{BF}_{\mathsf{dec}}$ has a strong PCP corresponding to the input $(1^n,\mathsf{BF}(1^n))$ and parameters $(T(n),r,h,q)$. Let $\mathcal{O}^*=(f_1^*,\dots,f_{6r+8}^*)$ be the canonical proof, where $f_i^*\in\mathsf{RM}_{3r+3,h,q}$ for each i.

- Let $d > 0$ be a constant. For each $f_i^* \in \mathsf{RM}_{3r+3,h,q}$, by Theorem 6.1, $\mathsf{RMV}^{f_i^*}_{h,d,m}$ determines a multi-set S_i of m-bit strings, where each element is indexed by an $O(r \log q)$-bit seed.

- The hitting set $H_{n,m}$ is defined as the union of all the sets S_i. Each element of $H_{n,m}$ (suppose it comes from S_i) has a unique index consisting of the encoding of i and the index of this element in S_i. By Theorem 6.1, assuming oracle access to $\mathcal{O}^*$, we can compute $H_{n,m}(s)$, the element in $H_{n,m}$ indexed by seed s, within time $\mathrm{poly}(m)$, as long as $m \le q^{1/100}$.

With the RMV reconstruction protocol in Theorem 6.1, there is an Arthur-Merlin protocol that computes any $f_i^*(x)$ on any input x efficiently in case that the hitting set generator fails:

Lemma 6.3. *For any constant $c > 0$, there is an Arthur-Merlin protocol σ_{eval}, such that for any $n \le m$, $i \le 6r + 8$ and $x \in \mathbb{F}_q^r$ (where q and r are defined as before), $\sigma_{\mathsf{eval}}(n, m, i, x)$ computes $f_i^*(x)$ with PSV error m^{-c}. The running time of $\sigma_{\mathsf{eval}}(n, m, i, x)$ is either*

- *$m^{O(\log n)}$ when $H_{n,m}$ fails to hit P, or*

- *$m^{O(1)} \cdot 2^{n^\beta}$ when $H_{n,m}$ hits P. Here $\beta > 1$ is a constant that only depends on P.*

There is also an Arthur Merlin protocol σ_{bf} such that $\sigma_{\mathsf{bf}}(n, m)$ computes $\mathsf{BF}(1^n)$ with the same running time and PSV error as above.

The second bullet of Lemma 6.3 is relatively easy as Merlin can send the entire description of polynomials $(f_1^*, \ldots, f_{6r+8}^*)$ to Arthur. The intuition of the first bullet of Lemma 6.3 is that when $H_{n,m}$ fails to hit P, one can run RMV reconstruction protocol together with the PCP verifier of $\mathsf{BF}_{\mathsf{dec}}(1^n, \mathsf{BF}(1^n))$ to compute both the oracle $\mathcal{O}^*$ and the output $\mathsf{BF}(1^n)$. We defer the proof of Lemma 6.3 to the end of this subsection and proceed with the proof of Theorem 1.3 first.

For any n, m, we say (n, m) forms a *critical pair*[38], if m is the largest integer in $\left[n, 2^{n^\beta}\right]$ (where β is the constant in Lemma 6.3), such that $H_{n,m}$ hits P (or $m = n$ and $H_{n,m'}$ fails to hits $P_{m'}$ for any $m' \in \left[n, 2^{n^\beta}\right]$). We will consider the following two different cases based on the distributions of different types of critical pairs:

- **Case 1: Easy reconstruction.** If there are infinite many critical pairs (n, m) such that $m \le 2^{\log^k n}$, we can run the reconstruction $\sigma_{\mathsf{bf}}(n, m + 1)$ in Lemma 6.3 to compute $\mathsf{BF}(1^n)$ within time $m^{O(\log n)} = 2^{\mathrm{polylog}(n)}$.

- **Case 2: Critical Hitting.** Otherwise, all but finite many critical pairs (n, m) satisfy $m \ge 2^{\log^k n}$. For each such critical pair, either $H_{n,m+1}$ fails to hit P or $m = 2^{n^\beta}$, and we can use $\sigma_{\mathsf{eval}}(n, m + 1, i, x)$ in Lemma 6.3 to compute the oracle $\mathcal{O}^*$ at any $f_i^*(x)$ in $m^{O(logn)} = 2^{\mathrm{polylog}(m)}$ time in both cases. Then, using the facts that $H_{n,m}$ hits P and $H_{n,m}$ is local, we can compute an element of $H_{n,m} \cap P$ also in $2^{\mathrm{polylog}(m)}$ time.

Below, we exhibit our protocols for each case separately:

[38]Our definition of critical pair here is slightly different from the definition in Section 6.3 to minimize the redundancy in the formal proof. The definition here allows us to unify Case 2 ("Easy Hitting") and Case 3 ("Critical Hitting") in Section 2.4.

Case 1: Easy reconstruction. Suppose that there are infinitely many critical pairs (n,m) such that $m \le 2^{\log^k n}$. We define the following Arthur-Merlin protocol σ_{rec} which takes inputs of form 1^n with $(\log^k n + 1)$ bits of advice:

- **Step 1: Making pairs.** On an input 1^n, Arthur reads the advice to get an integer $\widetilde{m} \in [1, 2^{\log^k n} + 1]$ which is supposed to be the integer m that forms a critical pair with n. (We use $\widetilde{m} = 2^{\log^k n} + 1$ to denote the case $m > 2^{\log^k n}$.) Arthur halts and outputs 0^n immediately if $\widetilde{m} = 2^{\log^k n} + 1$.
- **Step 2: Reconstruction.** Arthur runs the reconstruction protocol $\sigma_{\mathsf{bf}}(n, \widetilde{m}+1)$ in Lemma 6.3 for $\widetilde{m}^{C_{\mathsf{bf}} \log n}$ time with the help of Merlin to compute $\mathsf{BF}(1^n)$, where $C_{\mathsf{bf}} > 0$ is the constant in Lemma 6.3 such that $\sigma_{\mathsf{bf}}(n, \widetilde{m}+1)$ runs in $\widetilde{m}^{C_{\mathsf{bf}} \log n}$ time *if $H_{n,\widetilde{m}+1}$ fails hit P*. (Arthur will reject immediately if the protocol runs for $\widetilde{m}^{C_{\mathsf{bf}} \log n}$ time without termination.)
- **Canonical output:** The canonical output $x_n \in \{0,1\}^n$ on the input length n is defined as the lexicographic first n-bit string with property P (i.e. the output of $\mathsf{BF}(1^n)$) when $m \le 2^{\log^k n}$, or 0^n when $m > 2^{\log^k n}$. It is clear that x_n hits P for infinitely many n by the assumption that there are infinitely many critical pairs (n,m) such that $m \le 2^{\log^k n}$ and the definition of $\mathsf{BF}(1^n)$.

Claim 6.4 (Efficiency). *For every input length n, $\sigma_{\mathsf{rec}}(1^n)$ takes $(\log^k n + 1)$ bits of advice and runs in $2^{\log^{O(k)} n}$ time.*

Proof. The running time is dominated by the reconstruction step, which takes at most $\widetilde{m}^{C_{\mathsf{bf}} \log n} = 2^{\log^{O(k)} n}$ time. Moreover, the only advice in σ_{rec} appears in the first step where Arthur use advice to encode the integer $\widetilde{m} \in [1, 2^{\log^k n} + 1]$, which takes $(\log^k n + 1)$ bits. □

Claim 6.5 (Conformity). *For every input length n, if σ_{rec} is given the desired advice as we discussed above, there is a strategy for Merlin such that $\sigma_{\mathsf{rec}}(1^n)$ outputs x_n with probability 1.*

Proof. Assume the advice is correct ($\widetilde{m} = m$), with $\widetilde{m} \le 2^{\log^k n}$ (otherwise Arthur will halt and output $0^n =: x_n$). Then, Merlin's strategy is to perform honestly during the protocol $\sigma_{\mathsf{bf}}(n, \widetilde{m}+1)$. As (n,m) is a critical pair, $H_{n,\widetilde{m}+1}$ must fail to hit P, which means $\sigma_{\mathsf{bf}}(n, \widetilde{m}+1)$ terminates within $\widetilde{m}^{O(\log n)}$ time and Arthur outputs $\mathsf{BF}(1^n)$ canonically. □

Claim 6.6 (Resiliency). *For every input length n, for any advice ζ_n and Merlin's strategy, $\sigma_{\mathsf{rec}}(1^n)_{/\zeta_n}$ rejects or outputs canonically with probability at least $2/3$.*

Proof. If the advice $\widetilde{m} = 2^{\log^k n} + 1$, Arthur will reject immediately. Whichever the advice $\widetilde{m}$ is, as long as $\widetilde{m} \le 2^{\log^k n}$, Arthur will run the protocol $\sigma_{\mathsf{bf}}(n, \widetilde{m}+1)$ which computes $\mathsf{BF}(1^n)$ with PSV error $\widetilde{m}^{-c} < 1/3$. Hence, whichever the strategy of Merlin and no matter whether $\sigma_{\mathsf{bf}}(n, \widetilde{m}+1)$ can terminate within time $\widetilde{m}^{C_{\mathsf{bf}} \log n}$ or not, Arthur will either reject or output $\mathsf{BF}(1^n)$ with probability at least $2/3$, as desired. □

Case 2: Critical Hitting. In the remaining case, there are infinite many critical pairs (n, m) such that $m > 2^{\log^k n}$, we define the following Arthur-Merlin protocol σ_{hit} which takes inputs of form 1^m with $O(2^{\log^{1/k} m})$ bits of advice:

- **Step 1: Making pairs.** On an input 1^m, Arthur reads the advice to get an integer $\widetilde{n} \in [1, 2^{\log^{1/k} m} + 1]$, which is supposed to be the smallest n that forms a critical pair with m. (We use $\widetilde{n} = 2^{\log^{1/k} m} + 1$ to indicate the case that n does not exist or $n \geq 2^{\log^{1/k} m}$.) Arthur halts and outputs 0^m immediately if $\widetilde{n} = 2^{\log^{1/k} m} + 1$.
- **Step 2: Hitting.** Arthur reads the advice to get an $O(\widetilde{n})$-bit index $\widetilde{s}$ which is supposed to be the lexicographic first seed s such that $H_{\widetilde{n},m}(s) \in P$. By Theorem 6.1, Arthur uses the local algorithm $\mathsf{RMV}^p_{h,d}$ to compute $H_{\widetilde{n},m}(\widetilde{s})$ with oracle $\mathcal{O}^*$. Whenever Arthur need to query the oracle $\mathcal{O}^*$ for $f^*_i(x)$, Merlin and Arthur simulate the protocol $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ in Lemma 6.3 for $m^{C_{\mathsf{eval}} \log \widetilde{n}}$ steps to compute $f^*_i(x)$, i.e., Arthur halts and outputs 0^m when $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ exceeds this time bound. Arthur accepts and outputs the output string of local algorithm in Theorem 6.1 if all Arthur accepts in all simulations of $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, \cdot)$.
- **Canonical output:** The canonical output x_m is defined as the string in $H_{n,m} \cap P$ with lexicographic smallest seed (or 0^m when there is no n such that (n, m) is a critical pair and $n < 2^{\log^{1/k} m}$).

Claim 6.7 (Efficiency)**.** *$\sigma_{\mathsf{hit}}(1^m)$ takes $O(2^{\log^{1/k} m})$ bits of advice and runs in time $2^{O(\log^2 m)}$.*

Proof. The advice consists of the encoding of $\widetilde{n}$ and an $O(\widetilde{n})$-bit seed s, which is $O(\widetilde{n}) = O(2^{\log^{1/k} m})$ bits in total. Moreover, as the local algorithm in Theorem 6.1 takes $\mathrm{poly}(m)$ time where each query to the oracle $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ takes $m^{C_{\mathsf{eval}} \log \widetilde{n}}$ time (recall that Arthur immediately halts if the simulation of σ_{eval} exceeds the time bound), the total time complexity is $m^{O(\log \widetilde{n})} = 2^{O(\log^2 m)}$. □

Claim 6.8 (Conformity)**.** *For every input length m, if σ_{hit} is given the desired advice as we discussed above, there is a strategy for Merlin such that $\sigma_{\mathsf{hit}}(1^m)$ outputs canonically with probability 1.*

Proof. Assume the advice is correct ($\widetilde{n} = n$ and $\widetilde{s} = s$) with $\widetilde{n} < 2^{\log^{1/k} m}$ (otherwise Arthur will halt and output $0^m =: x_m$). Merlin's strategy is to perform honestly in any simulation of the protocol $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ (see Lemma 6.3), which ensures that each $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ terminates within $m^{C_{\mathsf{eval}} \log \widetilde{n}}$ time:

- When $m < 2^{n^\beta}$, as (n, m) forms a critical pair, $H_{\widetilde{n},m+1}$ must fail to hit P, which means $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ terminates within $m^{C_{\mathsf{eval}} \log \widetilde{n}}$ time;
- When $m = 2^{n^\beta}$, even if $H_{\widetilde{n},m+1}$ hits P with the running time of $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ being $m^{O(1)} 2^{\widetilde{n}^\beta}$, it is still bounded by $m^{C_{\mathsf{eval}} \log \widetilde{n}}$ as $m = 2^{n^\beta}$.

Hence, Arthur can get the correct $f^*_i(x)$ whenever Arthur and Merlin simulate $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$, and will output $H_{\widetilde{n},m}(\widetilde{s})$ with probability 1. □

Claim 6.9 (Resiliency)**.** *For every input length m, for any advice $\zeta_m = (\widetilde{n}, \widetilde{s})$ and any Merlin's strategy, $\sigma_{\mathsf{hit}}(1^m)_{/\zeta_m}$ rejects or outputs $H_{\widetilde{n},m}(\widetilde{s})$ with probability at least $2/3$.*

Proof. Suppose $\widetilde{n} \le 2^{\log^{1/k} m}$, otherwise Arthur will reject directly. Then, for any strategy of Merlin, Arthur either rejects or outputs $\widetilde{f}_i^*(x)$ when performing $\sigma_{\mathsf{eval}}(\widetilde{n}, m+1, i, x)$ with probability $(1 - m^{-c})$ as σ_{eval} has PSV error m^{-c} in Lemma 6.3.[39] As Arthur will query $\widetilde{\mathcal{O}}^*$ for at most $\mathrm{poly}(m)$ times, by choosing a sufficiently large constant $c > 0$ and using union bound, we can make sure that Arthur either rejects or outputs the correct $\widetilde{f}_i^*(x)$ for all oracle queries with probability at least 2/3, which means Arthur either rejects or outputs $H_{\widetilde{n},m}(\widetilde{s})$ with probability at least 2/3. □

Combining the protocols σ_{rec} and σ_{hit} for two cases, we conclude the proof of Theorem 1.3. □

Finally, we prove Lemma 6.3 to conclude this subsection.

Lemma 6.3 (Restated). *For any constant $c > 0$, there is an Arthur-Merlin protocol σ_{eval}, such that for any $n \le m$, $i \le 6r + 8$ and $x \in \mathbb{F}_q^r$ (where q and r are defined as before), $\sigma_{\mathsf{eval}}(n, m, i, x)$ computes $f_i^*(x)$ with PSV error m^{-c}. The running time of $\sigma_{\mathsf{eval}}(n, m, i, x)$ is either*

- *$m^{O(\log n)}$ when $H_{n,m}$ fails to hit P, or*
- *$m^{O(1)} \cdot 2^{n^\beta}$ when $H_{n,m}$ hits P. Here $\beta > 1$ is a constant that only depends on P.*

There is also an Arthur Merlin protocol σ_{bf} such that $\sigma_{\mathsf{bf}}(n, m)$ computes $\mathsf{BF}(1^n)$ with the same running time and PSV error as above.

Proof. Both σ_{eval} and σ_{bf} consist of three parts: a commitment protocol σ_{c}, an evaluation protocol σ_{e} and a verification protocol σ_{v}.

In the protocol $\sigma_{\mathsf{c}}(n, m)$, Merlin will commit to a proof oracle $\mathcal{O} = (f_1, \ldots, f_{6r+8})$ of $\mathsf{BF}_{\mathsf{dec}}(1^n, \mathsf{BF}(1^n))$. He will commits in different ways depending on whether $H_{n,m}$ hits P, as follows:

- Merlin first sends a bit $b \in \{0, 1\}$ indicating whether $H_{n,m}$ hits P.
- If $b = 0$, meaning that $H_{n,m}$ fails to hit P, then for all $1 \le i \le 6r + 8$, Merlin commits to the polynomial $f_i \in \mathsf{RM}_{3r+3,h,q}$ by the RMV commitment protocol in Theorem 6.1 with parameters h, d, m.[40] Arthur prepares a co-nondeterministic circuit D, which is the $\mathrm{poly}(m)$-sized (randomized) circuit accepting all the m-bit strings in P.
- If $b = 1$, Merlin will send the entire description of polynomials $(f_1, \cdots, f_{6r+8})$ to Arthur.

In the evaluation protocol σ_{e}, Arthur takes Merlin's commitment a from σ_{c} as a part of the input, and try to evaluate $f_i(x)$ on any input (n, m, i, x). Specifically:

- If $b = 0$, Arthur runs the RMV evaluation protocol in Theorem 6.1 to compute $f_i(x)$.
- If $b = 1$, Arthur computes $f_i(x)$ directly from the description of f_i.

Finally, in the verification protocol $\sigma_{\mathsf{v}}(n, m, a)$, Arthur checks whether $\mathcal{O}$ is the canonical proof $\mathcal{O}^*$ by running the PCP oracle verifier $V^{\mathcal{O}}_{\mathsf{BF}_{\mathsf{dec}}}$. Specifically, Arthur first runs low-degree test for all the polynomials in $\mathcal{O}$. Then, Merlin sends a string w which is supposed to be $\mathsf{BF}(1^n)$, and Arthur simulates the PCP oracle verifier $V^{\mathcal{O}}_{\mathsf{BF}_{\mathsf{dec}}}$ on the input $((1^n, w), T(n), r, h, q)$. Whenever Arthur needs to query the oracle $\mathcal{O}$ for the value of $f_i(x)$, he runs the evaluation protocol $\sigma_{\mathsf{e}}(n, m, i, x, a)$.

[39] Here, $\widetilde{f}_i^*$ is analogue of f_i^* but with respect to $\widetilde{n}$, i.e., the polynomial in the canonical proof of $\mathsf{BF}_{\mathsf{dec}}(1^{\widetilde{n}}, \mathsf{BF}(1^{\widetilde{n}}))$. The notation $\widetilde{\mathcal{O}}^*$ is defined similarly.

[40] Parameters r, h, q follow the same definition as in the definition of $H_{n,m}$.

Arthur accepts and outputs w if all the protocols $\sigma_{\mathsf{e}}(n,m,i,x,a)$ accept and the PCP verifier $V^{\mathcal{O}}_{\mathsf{BF}_{\mathsf{dec}}}$ accepts.

Combining all three protocols, σ_{bf} and σ_{eval} are constructed as follows: In $\sigma_{\mathsf{bf}}(n,m)$, Arthur and Merlin will first perform $\sigma_{\mathsf{c}}(n,m)$ to get the commitment a, and then run $\sigma_{\mathsf{v}}(n,m,a)$ (which requires execution of σ_{c} as a oracle) to output $\mathsf{BF}(1^n)$. In $\sigma_{\mathsf{eval}}(n,m,i,x)$, they first run σ_{c} and σ_{v} as in $\sigma_{\mathsf{bf}}(n,m)$; if $\sigma_{\mathsf{v}}(n,m,a)$ accepts, they further run $\sigma_{\mathsf{e}}(n,m,i,x,a)$ to compute $f_i^*(x)$.

Below, we check the efficiency, conformity and PSV error of σ_{bf} and σ_{eval} to complete the proof.

Claim 6.10 (Efficiency)**.** *The running time of both σ_{bf} and σ_{eval} are either $m^{O(\log n)}$ when $H_{n,m}$ fails to hit P, or $m^{O(1)} \cdot 2^{n^\beta}$ when $H_{n,m}$ hits P, where $\beta > 0$ is a constant that only depends on the language P.*

Proof. When $H_{n,m}$ fails to hit P, σ_{c} and σ_{e} are RMV commit-and-evaluation protocols, within time $m^{O(\log n)}$. In the verification protocol σ_{v}, Arthur runs low-degree test and PCP verifier, which take $\mathrm{poly}(m)$ time and call the oracle $\mathcal{O}$ for at most $\mathrm{poly}(m)$ times. Combining them, both σ_{bf} and σ_{eval} terminate within $m^{O(\log n)}$ time.

When $H_{n,m}$ hits P, σ_{c} and σ_{e} are brute-force algorithms for sending and evaluating $\mathcal{O}$ within 2^{n^β} time for some constant $\beta > 0$. The total running time of σ_{bf} and σ_{eval} will be $m^{O(1)}2^{n^\beta}$. □

Claim 6.11 (Conformity)**.** *There is a strategy of Merlin such that $\sigma_{\mathsf{bf}}(n,m)$ outputs $\mathsf{BF}(1^n)$ with probability 1, and $\sigma_{\mathsf{eval}}(n,m,i,x)$ outputs $f_i^*(x)$ with probability 1.*

Proof. Merlin's strategy is to send b honestly, to commit to the canonical proof $\mathcal{O}^*$, and to perform honestly in the RMV evaluation protocols. By the conformity of RMV reconstruction protocol in Theorem 6.1, Arthur will get the correct value $f_i^*(x)$ whenever he performs the evaluation protocol $\sigma_{\mathsf{e}}(n,m,i,x,a)$ when $b=0$, and clearly when $b=1$ as well. By the completeness of strong PCP, $V^{\mathcal{O}^*}_{\mathsf{BF}_{\mathsf{dec}}}$ will accept and output $\mathsf{BF}(1^n)$ with probability 1, which means both σ_{bf} and σ_{eval} will output correctly with probability 1. □

Claim 6.12 (PSV error)**.** *For any strategy of Merlin, $\sigma_{\mathsf{bf}}(n,m)$ will output $\{\mathsf{BF}(1^n),\perp\}$ with probability at least $\Omega(1)$, and $\sigma_{\mathsf{eval}}(n,m,i,x)$ will output $\{f_i^*(x),\perp\}$ with probability at least $\Omega(1)$.*

Proof. We first shows that, whichever the strategy of Merlin is, after the commitment step $\sigma_{\mathsf{bf}}(n,m)$, there are polynomials $\mathcal{O} = (f_1,\dots,f_{6r+8})$ depending on the commitment a, such that Arthur will output $\{f_i(x),\perp\}$ with probability $1-m^{-c}$ when performing $\sigma_{\mathsf{e}}(n,m,i,x,a)$, where $c>0$ is any fixed constant. Actually, this can be proved by the resiliency of RMV reconstruction protocol in Theorem 6.1[41] when $b=0$; and it is also clear when $b=1$, as Merlin sends the description of polynomials $(f_1,\dots,f_{6r+8})$ directly to Arthur.

Now, in the verification protocol σ_{v}, Arthur will query $\mathcal{O}$ for at most $\mathrm{poly}(m)$ times. By taking sufficiently large $c>0$ and using union bound, we can make sure Arthur either rejects or gets the value $f_i(x)$ for each query to $\mathcal{O}$ with probability at least $\Omega(1)$. The output of σ_{v} will depend on the proof oracle $\mathcal{O}$:

- If the committed proof $\mathcal{O}$ is the canonical one $\mathcal{O}^*$, then similar to the proof of conformity, Arthur will either output $\mathsf{BF}(1^n)$ or reject with probability at least $\Omega(1)$.
- If the committed proof $\mathcal{O}$ is not the canonical one $\mathcal{O}^*$, then either some function f_i is actually not a Reed-Muller codeword in $\mathsf{RM}_{3r+3,h,q}$ (which will be rejected during the low-degree test with probability $\Omega(1/r)$) or at least one pair of (f_i, f_i^*) is at least 1/2-far by the

[41] We need to repeat the reconstruction protocol for Theorem 6.1 for $\mathrm{poly}(m)$ times to boost the resiliency to $1-m^{-c}$.

property of Reed-Muller code (which means $V^{\mathcal{O}}_{\mathsf{BF}_{\mathsf{dec}}}$ will reject with probability at least $\Omega(1)$ by the strongness of PCP in Theorem 6.2).

Combining the two cases together, we get AM algorithms σ_{bf} and σ_{eval} as desired with PSV error $\Omega(1)$ and conformity error 0. □

Finally, we concludes the proof of Lemma 6.3 by noting that Lemma 6.3 requires a stronger PSV error m^{-c}, which can be achieved by parallel repetition of the protocol σ_{bf} and σ_{eval} constructed above for $O(c \cdot \log(m))$ times without significantly increasing the running time. □

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[AC22] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. *SIAM Journal on Computing*, March 2022.

[Ajt83] M. Ajtai. Σ^1_1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, July 1983.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

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

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

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

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

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proc. 13th Computational Complexity Conference (CCC)*, pages 8–12, 1998.

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

[BHPT24] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular PCPs. *SIAM Journal on Computing*, 53(2):480–523, April 2024.

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity class. *Journal of Computer and System Sciences*, 36(2):254–276, April 1988.

[BS05] Eli Ben-Sasson and Madhu Sudan. Simple PCPs with poly-log rate and query complexity. In *Proc. 37th ACM Symposium on Theory of Computing (STOC)*, pages 266–275, 2005.

[BS06] Joshua Buresh-Oppenheim and Rahul Santhanam. Making hard problems harder. In *Proc. 21st Computational Complexity Conference (CCC)*, pages 73–87, 2006.

[BSGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, January 2006.

[Cai07] Jin-Yi Cai. $S_2^p \subseteq ZPP^{NP}$. *Journal of Computer and System Sciences*, 73(1):25–35, February 2007.

[Che24] Lijie Chen. Nondeterministic quasi-polynomial time is average-case hard for ACC circuits. *SIAM Journal on Computing*, pages 19–332, February 2024.

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *Proc. 55th ACM Symposium on Theory of Computing (STOC)*, pages 1058–1066, 2023.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *Proc. 56th ACM Symposium on Theory of Computing (STOC)*, pages 1990–1999, 2024.

[CL24] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In *Proc. 56th ACM Symposium on Theory of Computing (STOC)*, pages 620–629, 2024.

[CLO$^+$23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *Proc. 64th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1261–1270, 2023.

[CT22] Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 125–136, 2022.

[FM05] Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information Processing Letters*, 95(2):354–357, July 2005.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, December 1984.

[GG11] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. Preprint ECCC:TR11-136, October 2011.

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *Proc. 26th International Conference on Approximation Algorithms for Combinatorial Optimization Problems and 27th International Conference on Randomization and Computation (APPROX/RANDOM)*, 2023.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, September 2015.

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *Proc. 25th International Conference on Approximation Algorithms for Combinatorial Optimization Problems and 26th International Conference on Randomization and Computation (APPROX/RANDOM)*, 2022.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. *Advances in Computing Research*, 5:73–90, 1989.

[Har04] Prahladh Harsha. *Robust PCPs of proximity and shorter PCPs*. PhD thesis, Massachusetts Institute of Technology, 2004.

[Has86] J Hastad. Almost optimal lower bounds for small depth circuits. In *Proc. 18th ACM Symposium on Theory of Computing (STOC)*, pages 6–20, 1986.

[HLR23] Shuichi Hirahara, Zhenjian Lu, and Hanlin Ren. Bounded relativization. In *Proc. 38th Computational Complexity Conference (CCC)*, pages 1–45, 2023.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Proc. 55th ACM Symposium on Theory of Computing (STOC)*, pages 1076–1089, 2023.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, December 2001.

[Juk12] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, Berlin, Heidelberg, 2012.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, October 1982.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc. 24th ACM Symposium on Theory of Computing (STOC)*, pages 723–732, 1992.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *Proc. 12th Innovations in Theoretical Computer Science (ITCS)*, 2021.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing (STOC)*, pages 302–309, 1980.

[Kor22] Oliver Korten. The hardest explicit construction. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 433–444, 2022.

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *Proc. 56th ACM Symposium on Theory of Computing (STOC)*, pages 2000–2007, 2024.

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Proc. 53rd ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 303–316, 2021.

[Lup58] Oleg Borisovich Lupanov. The synthesis of contact circuits. In *Doklady Akademii Nauk*, volume 119, pages 23–26, 1958.

[MVW99] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *Proc. 5th International Conference on Computing and Combinatorics (COCOON)*, pages 210–220, 1999.

[MW20] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM Journal on Computing*, 49(5):STOC18–300, January 2020.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[OS17] Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Proc. 49th ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 665–677, 2017.

[Par21] Orr Paradise. Smooth and strong PCPs. *Computational Complexity*, 30(1):1, January 2021.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 640–650, 2022.

[San09] Rahul Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM Journal on Computing*, 39(3):1038–1061, January 2009.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[Sha49] Claude. E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, January 1949.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, March 2005.

[SU07] Ronen Shaltiel and Christopher Umans. Low-end uniform hardness vs. randomness tradeoffs for am. In *Proc. 39th ACM Symposium on Theory of Computing (STOC)*, pages 430–439, 2007.

[TV07] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, December 2007.

[vS23] Dieter van Melkebeek and Nicollas Mocelin Sdroievski. Instance-wise hardness versus randomness tradeoffs for arthur-merlin protocols. In *Proc. 38th Computational Complexity Conference (CCC)*, pages 1–36, 2023.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1985.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. International Symposiumon on Symbolic and Algebraic Computation (EUROSAM)*, pages 216–226, 1979.

A Proof of Lemma 4.3

Lemma 4.3 (Restated)**.** *There are at most $2^{s(O(1)+\lceil\log(n+s)\rceil)}$ different L-oracle circuits of size s.*

Proof. It suffices to show that for each L-oracle circuit of size s, there is a "program" for a well-defined computing device that can be described in $(s+1)(O(1)+\log(n+s))$ bits that is functionally equivalent to the circuit. Indeed, we will prove this by modifying the construction in [FM05, Section 2].

Let n be the number of input bits and s be the size (i.e. number of wires) of an oracle circuit. An (L-oracle) *stack program* is described by a sequence of instructions in one of the four forms: "push i" (for $i \in [n+s]$), "pass", "call", and "op j" (for $j \in [16]$). The execution of a stack program is described as follows. Let $x \in \{0,1\}^n$ be the input. Let S, A be two stacks and O be a table of length $n+s$. Initialize $S \leftarrow \varnothing$, $A \leftarrow \varnothing$, $O_i \leftarrow x_i$ for every $i \in [n]$, and $O_i := 0$ for $i > n$.

- push i: Push $O_i \in \{0,1\}$ to the stack S.
- pass: Let $t \in \{0,1\}$ be the top of S. Pop the stack S, and push t to A.
- call: Let u be the string obtained by concatenating the bits in A. Empty the stack A, and push $L(u)$ to S.
- op j: Let $t_1, t_2 \in \{0,1\}$ be the two bits on the top of S. Pop t_1 and t_2 out of S. Push $\mathsf{op}_j(t_1,t_2)$ to S, where $\mathsf{op}_j : \{0,1\}^2 \to \{0,1\}$ is the j-th (out of 16) binary Boolean function.

The output of a stack program is top of S after running all instructions.

We will show that an L-oracle circuit of size s can be converted into a stack program with s push operations and at most $2s$ other operations[42], which can be easily encoded using $s \cdot \lceil\log(n+s)\rceil + O(s)$ bits.

The convertion can be done by the following algorithm. Fix any topological order of gates $g_1, g_2, \ldots, g_m$, where $m \le s-1$. For $i = 1, 2, \ldots, m$:

- Let d_i be the number of in-wires of g_i, we enumerate $j = 1, 2, \ldots, d_i$.
  - If the j-th in-wire is from the k-th input bit, write an instruction "push k".
  - If the j-th in-wire is from g_k, write an instruction "push $(n+k)$".
- If g_i is an oracle gate, write d_i copies of the instruction "pass", followed by "call".

[42] Recall that the size of an oracle circuit is defined as the number of *wires* in the circuit.

- If g_i is a gate computing $\mathsf{op}_j : \{0,1\}^2 \to \{0,1\}$, write an instruction op j.

The correctness of the convertion algorithm is straightforward. Moreover, since each wire will create exactly one "push" operation, there are exactly s "push" operations. This implies that there are at most s "pass" and "op" operations, and at most s "call" operations. $\square$

B Uniform Hardness-vs-Randomness for AM

Theorem 5.1 (Restated)**.** *There is an algorithm* HSG *and an Arthur-Merlin protocol* Rec *such that the following holds. Let $n, m, T \in \mathbb{N}$ be such that $n \le m \le T$, M be a Turing machine in a standard encoding such that $|M| \le \log\log T$, α be a string of length at most m, and $D : \{0,1\}^m \to \{0,1\}$ be a* $\mathrm{poly}(m)$*-size* coAM *circuit that rejects at most a 1/3-fraction of its inputs. Then* $\mathsf{HSG}(n, m, T, M, \alpha)$ *runs in time* $\mathrm{poly}(T)$ *and outputs a multiset $H \subseteq \{0,1\}^m$ of size* $\mathrm{poly}(T)$ *such that one of the following two conditions holds.*

- *(**Hit**). There exists a $z \in H$ such that $D(z) = 1$.*
- *(**Reconstruct**). The Arthur-Merlin protocol* $\mathsf{Rec}(n, m, T, M, \alpha, D, x)$ *runs in $m^{O((\log\log T)^2)}$ time and has $O(1)$ rounds such that the following holds:*
  - *(Completeness). If $M(\alpha)$ halts in time T and outputs $x \in \{0,1\}^n$, there is a strategy of the prover such that the verifier accepts with probability* 1.
  - *(Soundness). Otherwise, for any strategy of the prover, the verifier rejects with probability at least* 1/2.

Proof. Let $M'(\alpha, x, T)$ be the following Turing machine: Given any (α, x, T), it simulates $M(\alpha)$ for T steps and accepts if and only if $M(\alpha)$ halts and outputs x. Note that M' runs in time at most T^2. Let Prf be the algorithm in Theorem 3.4 with $M := M'$, we define $\pi := \mathsf{Prf}((\alpha, x, T), T^2)$ be the PCP proof for $M'(\alpha, x, T) = 1$, where $\pi : \{0,1\}^{O(\log T)} \to \{0,1\}$.

Let h be the smallest power of two such that $h \ge m^{100}$, $q := h^{100}$, $d := O(1)$ be a power of two, and $r = O(\log|\pi| / \log h)$ so that there is a unique low-degree extension for π in $\mathbb{F}_q$ with degree h and r variables (see Lemma 3.2). We assume without loss of generality that r is a power of d, $r = O(d \log T / \log h)$, and $h = m^{100}$ (as we can always ignore some bits of a hitting set). Let $p : \mathbb{F}_q^r \to \mathbb{F}_q$ be the unique low-degree extension of π with degree h and an efficient encoding $I : [|\pi|] \to \mathbb{F}_q^r$ such that for every $x \in [|\pi|]$, $p(I(x)) = \pi_x$. We define $H \subseteq \{0,1\}^m$ be the hitting set of size $q^{O(r)} = \mathrm{poly}(T)$ from $\mathsf{RMV}_{h,d}(p)$ in Theorem 3.5 using aforementioned parameters.

Fix any coAM circuit $D : \{0,1\}^m \to \{0,1\}$. Suppose that $D(z) = 0$ for every $z \in H$, i.e., we are not in the "(Hit)" case. Then the reconstruction Arthur-Merlin protocol $\mathsf{Rec}(n, m, T, M, \alpha, D, x)$ works as follows:

1. Arthur and Merlin simulate the commit protocol σ_{c} in Theorem 3.5, and outputs a commitment $\gamma \in \{0,1\}^\ell$ of length $\ell = O(h^{O(d)} r \log q) = \mathrm{poly}(m, \log T)$. The protocol runs in time $\mathrm{poly}(|D|, \ell) = \mathrm{poly}(m, \log T)$. Honest Merlin is supposed to commit to the polynomial $p : \mathbb{F}_q^r \to \mathbb{F}_q$ as defined above.

2. Let $V_{M'}^{\mathcal{O}}$ be the verifier in Theorem 3.4 for the Turing machine M'. Arthur simulates the verifier $V_{M'}^{\mathcal{O}}((\alpha, x, T), T^2)$ and for each query $q_i \in [\mathrm{poly}(T)]$ made by the verifier, they simulate $\sigma_{\mathsf{e}}(I(q_i), D, \gamma)$ and use the output of the protocol σ_{e} mod 2 as the answer to the oracle query. Arthur rejects immediately if it rejects in the simulation of σ_{e}.

3. Arthur accepts if and only if $V_{M'}$ accepts.

Efficiency. Note that the verifier $V^{\mathcal{O}}_{M'}$ runs in time $\mathrm{poly}(n, |\alpha|, \log T)$ and makes $O(1)$ queries to its oracle, for each of which Arthur and Merlin need to simulate σ_{e} in time

$$h^{O(d \log_d^2 r)} = m^{O(\log^2(\log T / \log m))} = m^{O((\log \log T)^2)}.$$

and $O(1)$ rounds. Therefore, the whole protocol runs in time $m^{O((\log \log T)^2)}$ and has $O(1)$ rounds.

Completeness. Suppose that $M(\alpha)$ terminates in T steps and outputs $x \in \{0,1\}^n$. By the definition of M' we know that $M'(\alpha, x, T)$ terminates in T^2 steps and accepts. In this case, Merlin could commit to the polynomial $p : \mathbb{F}_q^r \to \mathbb{F}_q$ as mentioned above, so that for every $x \in [|\pi|]$, there is a strategy for Merlin such that $\sigma_{\mathsf{e}}(I(x), D, \gamma) = p(I(x)) = \pi[x]$ with probability 1, by the perfect completeness of the commit-and-evaluate protocol in Theorem 3.5. Therefore, by the perfect completeness of the verifier (see Theorem 3.4), $V^{\mathcal{O}}_{M'}((\alpha, x, T), T^2) = 1$ with probability 1. This means that Arthur will accept in the protocol $\mathsf{Rec}(n, m, T, M, \alpha, D, x)$ with probability 1.

Soundness. Suppose that either $M(x)$ does not terminate in T steps or it does not output $x \in \{0,1\}^n$. By the definition of M' we know that $M'(\alpha, x, T)$ terminates in T^2 steps and rejects. Fix any strategy of Merlin in Rec, which consists of a strategy τ_{c} for σ_{c} in Step 1 of Rec, and a strategy τ_{e} for σ_{e} in Step 2 of Rec. Fix any commitment $\gamma \in \{0,1\}^\ell$ in Step 1. By the resiliency of the commit-and-evaluate protocol in Theorem 3.5, we know that there exists a function $g : \mathbb{F}_q^r \to \mathbb{F}_q$ such that $\sigma_{\mathsf{e}}^{\tau_{\mathsf{e}}}(u, D, \gamma) \in \{g(u), \bot\}$ with probability at least $1 - o(1)$ (over the randomness of Arthur in Step 2 of Rec.)

Let E_1 be the event that $\sigma_{\mathsf{e}}^{\tau_{\mathsf{e}}}(I(q_i), D, \gamma) \in \{\bot, g(I(q_i))\}$ for some query q_i made in Step 2 of the protocol Rec in the simulation of $V^{\mathcal{O}}_{M'}$. Similarly, let E_2 be the event that $\sigma_{\mathsf{e}}^{\tau_{\mathsf{e}}}(I(q_i), D, \gamma) = \bot$. Notice that:

- $\Pr[\mathsf{Rec}\text{ accepts} \mid E_1, E_2] = 0$, by the definition of Rec.
- $\Pr[\mathsf{Rec}\text{ accepts} \mid E_1, \neg E_2] \le 1/3$. This is because given $E_1 \wedge \neg E_2$, $\sigma^{\tau_e}(I(q_i), D, \gamma) = g(I(q_i))$, and thus Arthur accepts if and only if the verifier $V^{\mathcal{O}_g}_{M'}((\alpha, x, T), T^2)$ accepts for the oracle $\mathcal{O}_g : \{0,1\}^{O(\log T)} \to \{0,1\}$ defined as $\mathcal{O}_g(x) := g(I(x)) \bmod 2$. Recall that $M'(\alpha, x, T)$ terminates in T^2 steps rejects, and the acceptance probability follows from the soundness of the verifier (see Theorem 3.4).

Therefore, we have

$$\Pr[\mathsf{Rec}\text{ accepts}] \le \Pr[\neg E_1] + \Pr[\mathsf{Rec}\text{ accepts} \mid E_1, E_2] + \Pr[\mathsf{Rec}\text{ accepts} \mid E_1, \neg E_2] < 1/2.$$

This completes the proof. □

C Strong PCP with Reed-Muller-encoded Proofs

Theorem 6.2 (Restated). *There is a constant $\alpha \in (0,1)$ such that for any Turing machine M, there is a constant $c \ge 1$ and a probabilistic polynomial-time oracle verifier $V^{\mathcal{O}}_M$ satisfying the following. Let $x \in \{0,1\}^n$, $T \ge n$ be a time bound encoded in binary, $r, h \ge 1$ and q be a power of a prime $p = O(1)$, such that $r = \Theta(\log T / \log \log T)$, $h \ge n^c \cdot T^{c/r}$, $h^c \le q \le T$.*

- *Given input (x, T, r, h, q) to the verifier $V^{\mathcal{O}}_M$, the proof oracle $\mathcal{O}$ is supposed to be a sequence of polynomials $f_1, f_2, \dots, f_{6r+8} \in \mathsf{RM}_{3r+3,h,q}$. The verifier tosses $O((r+h)\log q)$ random coins, generates $k = O(rh)$ non-adaptive queries $(i_1, x_1), (i_2, x_2), \dots, (i_k, x_k) \in [6r+8] \times \mathbb{F}_q^{3r+3}$, and decides in* $\mathrm{poly}(r, h, \log q)$ *time whether to accept the proof given answers $f_{i_1}(x_1), f_{i_2}(x_2), \dots, f_{i_k}(x_k) \in \mathbb{F}_q$.*

- *(Completeness). If $M(x)$ halts in T steps and accepts, there is a* unique *oracle $\mathcal{O}^*$ such that $\Pr[V_M^{\mathcal{O}^*}(x,T,r,h,q)=1]=1$. We call this oracle $\mathcal{O}^*=(f_1^*,f_2^*,\ldots,f_{6r+8}^*)$ the* canonical proof *corresponding to the input (x,T,r,h,q).*

- *(Soundness). If $M(x)$ does not halt in T steps, or $M(x)$ rejects, then for every oracle $\mathcal{O}=(f_1,f_2,\ldots,f_{6r+8})$, $\Pr[V_M^{\mathcal{O}}(x,T,r,h,q)=1]\le 1-\alpha$.*

- *(Strong soundness). If $M(x)$ halts in T steps and accepts, then for every oracle $\mathcal{O}=(f_1,f_2,\ldots,f_{6r+8})$, if for any constant $\delta\in(0,1)$, f_i is δ-far from the i-th polynomial f_i^* in the canonical proof for some $i\in[6r+8]$, then*

  $$\Pr[V_M^{\mathcal{O}}(x,T,r,h,q)=1]\le 1-\alpha\cdot\delta,$$

  where $\alpha\in(0,1)$ is a universal constant.

Our construction of the verifier $V_M^{\mathcal{O}}$ mostly follows from the standard poly-logarithmic PCP using Reed-Muller code (see, e.g., [Par21]). A subtle difference is that we will need the field size q to be as large as $T\ge 2^n$, while in the standard setting, it is usually set to be $\Theta(\log T)$.[43]

We will follow the strong PCP construction from Reed-Muller code in [Par21, Section 5] (also see [BS05, Har04]), with several modifications:

1. The two main components of the PCP construction, namely low-degree testing and zero-on-the-subcube, needs to be updated to work with finite fields of large size. The original verifier involves queries to the projection of polynomials to a random line, which has a time overhead proportional to the field size. We need to treat the projection as a univariate polynomial and apply the decoding of the Reed-Solomon code (see Appendix C.2 and C.3).

2. The original PCPP (PCP of Proximity) protocol in [Par21] for the circuit evaluation problem assumes an explicit access to the circuit. In our case, however, the circuit is of size $\mathrm{poly}(T)$ and thus we can only assume an oracle access to it. Nevertheless, this can be solved by a standard algebrization technique (see Appendix C.4 for more details) with a careful verification of the strongness and canonicity of the PCP proof.

3. In the construction of [Par21], the first proof oracle f_1 has only r variables instead of $3r+3$ variables. One can obtain a sound PCP by simply ignoring the last $2r+3$ variables. However, to ensure the strong soundness of the PCP verifier, we need to apply an additional individual degree testing (see Appendix C.5 for more details).

4. There is a bug in the proof of [Par21] regarding the uniqueness of the "division witness" (see Proposition C.5 and the discussion below). We fixed this bug for our purpose of proving Theorem 6.2 by introducing an additional "individual-degree check".

C.1 Definitions and Tools

We start with some useful definitions. Let $\mathbb{F}=\mathbb{F}_q$. A function $f:\mathbb{F}^r\to\mathbb{F}^k$ defined as $f(x):=(f_1(x),\ldots,f_k(x))$ is a k-dimension vector-valued polynomial of degree h if for every $i\in[k]$, $f_i\in\mathsf{RM}_{r,h,q}$. We denote the set of all k-dimension vector-valued polynomials of degree h as $\mathsf{RM}_{r,h,q}^k$. Similarly, we define $\mathsf{RS}_{h,q}^k:=\mathsf{RM}_{1,h,q}^k$. The (Hamming) distance between $f,g\in\mathsf{RM}_{r,h,q}^k$ is defined as $\delta(f,g):=\Pr_x[f(x)\ne g(x)]$.

[43] We note that making q to be as large as $T\ge 2^n$ will greatly increase the length of the PCP proof, which is not helpful in the standard setting, while it is necessary in our critical win-win argument.

We define a line $\mathcal{L}$ through $\mathbb{F}^r$ with *intercept* $x \in \mathbb{F}^r$ and *slope* $h \in \mathbb{F}^r$ to be $\mathcal{L} := \{x + ih \mid i \in \mathbb{F}\}$. Let $\mathsf{L}_{r,q}$ be the set of all such lines. A uniformly random line is defined by sampling the intercept and slope uniformly over $\mathbb{F}^r$. The restriction of f to a line $\mathcal{L}$, denoted by $f|_{\mathcal{L}} : \mathbb{F} \to \mathbb{F}^r$, is the function $f|_{\mathcal{L}}(i) := f(x + ih)$.

Proposition C.1. *If $f \in \mathsf{RM}_{r,h,q}$, then $f|_{\mathcal{L}} \in \mathsf{RS}_{h,q}$ for any line $\mathcal{L}$.*

Lemma C.2 (Schwatz-Zippel Lemma [Sch80, Zip79])**.** *For any finite field $\mathbb{F} = \mathbb{F}_q$ and intergers r, h, if $f \in \mathsf{RM}_{r,h,q}$ is a non-zero polynomial, then $\Pr_{x \leftarrow \mathbb{F}^m}[f(x) = 0] \le h/|\mathbb{F}|$.*

C.2 Low-Degree Testing

Proposition C.3 ([Par21, Proposition 5.5])**.** *Assume that $\mathbb{F} = \mathbb{F}_q$ for $q > 25k$. Let $g : \mathsf{L}_{r,q} \times \mathbb{F}^h \to \mathbb{F}^k$ be an arbitrary oracle such that for each line $\mathcal{L}$, $g_{\mathcal{L}} := g(\mathcal{L}, \cdot) \in \mathsf{RS}^k_{h,q}$. If $f : \mathbb{F}^r \to \mathbb{F}^k$ is δ-far from being in $\mathsf{RM}^k_{r,h,q}$, then over a uniformly random line $\mathcal{L}$ and a uniformly random $u \in \mathbb{F}$, $f|_{\mathcal{L}}(u) \neq g_{\mathcal{L}}(u)$ with probability at least $\delta/40$.*

The following low-degree testing algorithm is a straightforward improvement of [Par21, Algorithm 5.6] when the field size q is large.

Theorem C.4 (Low-Degree Testing)**.** *Let $\mathbb{F} = \mathbb{F}_q$ for $q > 25k$, $r, h \ge 1$, and $\delta \in (0, 1)$. There is an algorithm such that given oracle access to $f : \mathbb{F}^r \to \mathbb{F}^k$, it tosses $O((r + h)\log q)$ random coins, makes $h + 2$ non-adaptive oracle queries, runs in time $\mathrm{poly}(r, h, \log q)$ such that:*

- *(Completeness). If $f \in \mathsf{RM}^k_{r,h,q}$, the algorithm accepts with probability* 1.
- *(Soundness). If f is δ-far from $\mathsf{RM}^k_{r,h,q}$, the algorithm rejects with probability at least $\delta/40$.*

Proof. Let $f = (f_1, \dots, f_k) : \mathbb{F}^r \to \mathbb{F}^k$. The algorithm works as follows: It uniformly samples a line $\mathcal{L} = \{x + it \mid i \in \mathbb{F}\}$ using $O(r \log q)$ random bits, and $m = h + 2$ points $u^1, \dots, u^m \in \mathbb{F}$ using $O(m \log q)$ random bits. It makes m oracle queries $f|_{\mathcal{L}}(u^1), , \dots, f_{\mathcal{L}}(u^m) \in \mathbb{F}^k$. For each $i \in [k]$, it computes the unique univariate polynomial $g_i \in \mathsf{RS}_{h,q}$ such that for every $j \in [h+1]$, $g_i(u^j) = f_{\mathcal{L}}(u^j)_i$ by Lagrange interpolation. The algorithm accepts if and only if for every $i \in [k]$, $g_i(u^m) = f_{\mathcal{L}}(u^m)_i$.

To prove the completeness of the algorithm, we can see that if $f \in \mathsf{RM}^k_{r,h,q}$, then by Proposition C.1 for every line $\mathcal{L}$, $f|_{\mathcal{L}} \in \mathsf{RS}^k_{h,q}$. This means that for every $i \in [k]$, $f_i|_{\mathcal{L}}$ is of degree h, and thus the polynomial g_i from Lagrange interpolation agrees with $f_i|_{\mathcal{L}}$.

Now assume that f is δ-far from $\mathsf{RM}^k_{r,h,q}$. Let $\hat{g} : \mathsf{L}_{r,q} \times \mathbb{F}^h \to \mathbb{F}^k$ be the oracle that minimizes $\Pr_u[\hat{g}_{\mathcal{L}}(u) \neq f|_{\mathcal{L}}(u)]$ for every line $\mathcal{L}$, where $\hat{g}_{\mathcal{L}} := g(\mathcal{L}, \cdot) \in \mathsf{RS}^k_{h,q}$. By Proposition C.3, we know that over a uniformly random line $\mathcal{L}$ and $u \in \mathbb{F}$, $\Pr[g_{\mathcal{L}}(u) \neq f|_{\mathcal{L}}(u)] \ge \delta/40$. Now we fix any $u^1, \dots, u^{h+1} \in \mathbb{F}$. Over a uniformly random line $\mathcal{L}$ and u^m, let $g = (g_1, \dots, g_k) \in \mathsf{RS}^k_{h,q}$, where g_i is obtained from Lagrange interpolation as described above, then

$$\Pr_{\mathcal{L}, u^m}[g(u^m) \neq f|_{\mathcal{L}}(u^m)] \ge \Pr_{\mathcal{L}, u}[\hat{g}_{\mathcal{L}}(u) \neq f|_{\mathcal{L}}(u)] \ge \delta/40.$$

This means that with probability at least $\delta/40$, there is an $i \in [k]$ such that $g_i(u^m) \neq f|_{\mathcal{L}}(u^m)_i$, in which case the algorithm rejects. □

C.3 Zero-on-Subcube

The Zero-on-Subcube (ZoS) problem is defined as follows. Fix a finite field $\mathbb{F} = \mathbb{F}_q$, $r, h \geq 1$, and $H \subseteq \mathbb{F}$. Given a polynomial $f \in \mathsf{RM}_{r,h,q}$, the ZoS problem is to decide whether $f(x) = 0$ for all $x \in H^r$. Let $\mathsf{ZoS}_{r,h,q,H}$ be the set of all such polynomials.

Proposition C.5 ([Par21, Fact 5.10])**.** *Let $\mathbb{F} = \mathbb{F}_q$, $r, h \geq 1$, $H \subseteq \mathbb{F}$, and $f \in \mathsf{RM}_{r,h,q}$. Then $f \in \mathsf{ZoS}_{r,h,q,H}$ if and only if there are $P = (P_1, \ldots, P_r) \in \mathsf{RM}^r_{r,h,q}$ and $Q = (Q_1, \ldots, Q_r) \in \mathsf{RM}^r_{r,h-|H|,q}$ such that for every $i \in [r]$:*

$$P_{i-1}(x) = \mu(x_i) \cdot Q_i(x) + P_i(x),$$
$$P_r(x) = 0,$$

where $P_0 := f$ and $\mu \in \mathsf{RS}_{|H|,q}$ is defined as $\mu(z) := \Pi_{u \in H}(z - u)$.

Moreover, For every $f \in \mathsf{ZoS}_{r,h,q,H}$, there is a unique pair $(P = (P_1, \ldots, P_r), Q = (Q_1, \ldots, Q_r))$ satisfying the condition above such that for every $i \in [r]$, the individual degree of x_i in P_i is at most $|H| - 1$.

It it claimed in [Par21] that (P, Q) is unique even without the individual degree constraint we mentioned in the "moreover" part, which does not look right to us. We resolved this issue by introducing the constraint and performing an additional "individual-degree testing" (see the proof of Theorem C.6) based on Proposition C.5.

Proof of Proposition C.5. The equivalence is implicit in [BS05]; for completeness, we provide a proof here. Note that ($\Leftarrow$) is straightforward. If such (P, Q) exists, then

$$f(x) = \sum_{i \in [r]} \mu(x_i) Q_i(x),$$

and the LHS is clearly zero on H^m since $\mu(x_i) = 0$ for all $i \in [r]$ and $x_i \in H$.

Now we prove the other direction by constructing $(P_1, Q_1), \ldots, (P_r, Q_r)$ inductively. Indeed, we will further ensure in the construction that $x_1, \ldots, x_i$ have individual degrees at most $|H| - 1$ in P_i. Let $P_0 := f$. Assume that we have already constructed P_{i-1} such that $x_1, \ldots, x_{i-1}$ have individual degree at most $|H| - 1$. Note that we can view P_{i-1} as a univariate polynomial over the ring $\mathbb{F}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. Therefore, by division with remainder, there are polynomials Q_i, P_i such that

$$P_i(x) = \mu(x_i) \cdot Q_i(x) + P_i(x)$$

satisfying that the individual degree of x_i in $P_i(x)$ is at most $|H| - 1$. (Recall that μ is a univariate polynomial of degree $|H|$.)

Now we show that $P_r(x) = 0$. For each $x \in H^m$, we have

$$P_r(x) = f(x) - \sum_{i \leq r} \mu(x_i) Q(x) = 0 - \sum_{i \leq r} 0 = 0.$$

Note that each variable has individual degree at most $|H| - 1$. This immediately implies that $P_r(x)$ is the zero polynomial.

It remains to prove the uniqueness. Towards a contradiction, we assume that there are two such pairs (P, Q) and (P', Q') satisfying the individual degree requirement. Let $P = (P_1, \ldots, P_r)$, $Q = (Q_1, \ldots, Q_r)$, $P' = (P'_1, \ldots, P'_r)$, and $Q' = (Q'_1, \ldots, Q'_r)$. Clearly if $Q = Q'$ then $P = P'$. Let i be the smallest index such that $Q_i \neq Q'_i$, then:

$$P_{i-1}(x) = P'_{i-1}(x) = \mu(x_i) Q_i(x) + P_i(x) = \mu(x_i) Q'_i(x) + P'_i(x),$$

which means that

$$P_i(x) - P_i'(x) = \mu(x_i)(Q_i'(x) - Q_i(x)).$$

However, this is impossible as x_i has has degree at most $|H|-1$ in the LHS and has degree at least $|H|$ in the RHS. $\square$

For $f \in \mathsf{ZoS}_{r,h,q,H}$, we call the unique $P \in \mathsf{RM}^r_{r,h,q}, Q \in \mathsf{RM}^r_{r,h-|H|,q}$ in Proposition C.5 the *division witness* of $f \in \mathsf{ZoS}_{r,h,q,H}$.

Theorem C.6 (Modification over [Par21, Lemma 5.13]). *There is an absolute constant $\beta > 0$ such that the following holds. Let $\mathbb{F} = \mathbb{F}_q$, $r, h \geq 1$, $H \subseteq \mathbb{F}$, such that $q \geq 10 \cdot \max\{|H|, h\}$ and $\beta \leq 1/4 - h/q$. There is an algorithm $V^{f,P,Q}_{\mathsf{ZoS}}$ such that given oracle access to $f : \mathbb{F}^r \to \mathbb{F}$, and $P = (P_1, \ldots, P_r), Q = (Q_1, \ldots, Q_r) \in \mathbb{F}^r \to \mathbb{F}^r$, it tosses $O((r + h + |H|)\log q)$ random coins, makes $O(r|H| + h)$ non-adaptive oracle queries, runs in time $\mathrm{poly}(r, h, \log q, |H|)$ such that:*

- *(Completeness). If $f \in \mathsf{ZoS}_{r,h,q,H}$ and (P, Q) is its division witness, the algorithm always accepts.*
- *(Strong soundness). Let (f', P', Q') be the tuple satisfying $f' \in \mathsf{ZoS}_{r,h,q,H}$ with division witness (P', Q') that minimizes*

  $$\delta := \max\{\delta(f, f'), \delta(P, P'), \delta(Q, Q')\}.$$

  Then the algorithm rejects with probability at least $\beta\delta$.

Proof. The verifier works as follows.

1. (Low-degree check). Run the algorithm in Theorem C.4 on f, P, and Q. This is to check that f and P are polynomials of degree at most h, and Q is of degree at most $h - |H|$.
2. (Division witness). Let $m = 10|H|$. We uniformly sample $z \in \mathbb{F}^r$ and $u_1, \ldots, u_m \in \mathbb{F}$.
   (a) (Individual degree check). For each $i \in [r]$ we perform the following test. Let $\mathcal{L}$ be the line with intercept z and slope e^i, where $e^i_i = 1$ and $e^i_j = 0$ for every $j \neq i$. That is, $\mathcal{L}$ is the line parallel to the i-th axis passing though z. We query $P_i|_{\mathcal{L}}(u_1), \ldots, P_i|_{\mathcal{L}}(u_m)$. Let g be the unique degree-$(|H|-1)$ polynomial such that $g(u_i) = P_i|_{\mathcal{L}}(u_i)$ for every $i \leq |H|$ using Lagrange interpolation. We then check whether $g(u_i) = P_i|_{\mathcal{L}}(u_i)$ for every $i \in [m]$.
   (b) (Division check). For each $i \in [r]$, and check whether

   $$P_{i-1}(z) = \mu(z_i) \cdot Q_i(z) + P_i(z),$$

   where $P_0 := f$ and $\mu(x) := \prod_{u \in H}(x - u)$.
   (c) (Identity check). Check that $P_r(z) = 0$.

The algorithm accepts if it passes all the checks. The randomness complexity, query complexity, and time complexity of the algorithm are obvious. The completeness of the algorithm follows directly from Theorem C.4 and Proposition C.5. Therefore, it suffices to prove the soundness.

Case 1. We define $\hat{f} \in \mathsf{RM}_{r,h,q}$ that minimizes $\delta(\hat{f}, f) =: \delta_f$, $\hat{P} \in \mathsf{RM}^r_{r,h,q}$ that minimizes $\delta(\hat{P}, P) =: \delta_P$, and $\hat{Q} \in \mathsf{RM}^r_{r,h-|H|,q}$ that minimizes $\delta(\hat{Q}, Q) =: \delta_Q$. Suppose that $\max(\delta_f, \delta_P, \delta_Q) \geq 1/8$. Clearly, $\delta \geq \max(\delta_f, \delta_P, \delta_Q)$. By Theorem C.4, the low-degree check fails with probability at least $1/320 \geq \beta\delta$, if we choose $\beta \leq 1/320$. Therefore, we can assume in the rest of the proof that f, P, Q are $(1/8)$-close to $\hat{f}, \hat{P}, \hat{Q}$, respectively.

Case 2. Suppose that there is an $i \in [r]$ such that x_i has individual degree at least $|H|$ in $\hat{P}_i(x)$. Let $\mathcal{L}$ be the line with intercept z that is parallel to the i-th axis, and $z^j =: (z_1, \ldots, z_{i-1}, u_j, z_{i+1}, \ldots, z_r)$ for every $j \in [m]$. Note that $P_i|_{\mathcal{L}}(u_i) = P_i(z^j)$. Since the marginal distribution of z^j is uniformly random over $\mathbb{F}^r$, and $\hat{P}_i$ is $(1/4)$-close to P_i, we know that $\Pr[\hat{P}_i(z^j) \neq P_i(z^j)] \leq 1/8$ and

$$\mathop{\mathbb{E}}_{z,u_1,\ldots,u_m}\left[\sum_{j\in[m]} \mathbb{I}\left[\hat{P}_i(z^j) \neq P_i(z^j)\right]\right] \leq \frac{m}{8}.$$

By Markov inequality, we know that with probability at least $2/3$,

$$\sum_{j\in[m]} \mathbb{I}\left[\hat{P}_i(z^j) \neq P_i(z^j)\right] \leq \frac{3m}{8}. \tag{3}$$

Since $\hat{P}_i(x)$ has total degree at most h, we know that for every univariate polynomial g of degree at most $|H|-1$, $\delta(\hat{P}_i|_{\mathcal{L}}, g) \geq 1 - \max\{|H|, h\}/q \geq 1/2$. Fix any $u_1, \ldots, u_{|H|}$ and let g be the unique degree-$|H|$ polynomial such that $g(u_j) = P_i|_{\mathcal{L}}(u_j)$ for every $j \in [|H|]$. Over uniformly random $u_{|H|+1}, \ldots, u_m$, we know that the expected number of indices $|H|+1 \leq j \leq m$ such that $g(u_j) = \hat{P}_i|_{\mathcal{L}}(u_j)$ is at most $(m-|H|)/2$. By Markov inequality, we know that with probability at least $2/3$:

$$\sum_{j\in[m]} \mathbb{I}\left[\hat{P}_i(z^j) = g(u_j)\right] \leq \frac{3(m-|H|)}{4} \leq \frac{3m}{8} \tag{4}$$

By the union bound, we know that with probability at least $1/3$, we will have both (3) and (4); this implies that there is a $j \in [m]$ such that $P_i(z^j) \neq g(u_j)$, in which case the individual degree check fails. The acceptance probability is at most β if we set $\beta \leq 1/3$.

Case 3. Suppose that $(\hat{P}, \hat{Q})$ is not the division witness of $\hat{f} \in \mathsf{ZoS}_{r,h,q,H}$. Since $\hat{P}$ satisfies the individual degree requirement in Proposition C.5, either there is an $i \in [r]$ such that $\hat{P}_{i-1}(x) \neq \mu(x_i) \cdot \hat{Q}_i(x) + \hat{P}_i(x)$, or $\hat{P}_r(x) \neq 0$. In the former case, we know by Schwatz-Zippel Lemma (see Lemma C.2) that over a uniformly random $z \leftarrow \mathbb{F}^r$,

$$\Pr_z[\hat{P}_{i-1}(z) \neq \mu(z_i) \cdot \hat{Q}_i(z) + \hat{P}_i(z)] \geq 1 - \frac{h}{q}.$$

By the union bound and the closeness of P, Q to $\hat{P}, \hat{Q}$, we can show that

$$\Pr_z[P_{i-1}(z) \neq \mu(z_i) \cdot Q_i(z) + P_i(z)] \geq \frac{1}{4} - \frac{h}{q} \geq \beta,$$

i.e., the division check fails with probability at least β. Similarly, if $\hat{P}_r(x) \neq 0$, the identity check fails with probability at least δ.

Case 4. It remains to consider the case that $(\hat{P}, \hat{Q})$ is a division witness of $\hat{f} \in \mathsf{ZoS}_{r,h,q,H}$. In such case, we know that $\delta = \max\{\delta_f, \delta_P, \delta_Q\}$. By Theorem C.4, we know that the low-degree check must fail with probability at least $\max\{\delta_f, \delta_P, \delta_Q\}/40 \geq \beta\delta$ if we set $\beta \leq 1/40$. □

C.4 Algebrization

Note that the verifier for ZoS in Theorem C.6 can be regarded as a satisfiability algorithm in the algebraic setting. In this sub-section, we show how to reduce the verification of uniform computation to ZoS and thus deduce Theorem 6.2. As this step is quite standard, we will only present proof sketches of the claims.

Cook-Levin witness. Recall the standard reduction from program verification to succinct 3-CNF satisfiability. This essentially follows from the proof of the Cook-Levin theorem.

Theorem C.7 (Folklore)**.** *Let M be a Turing machine. Let $x \in \{0,1\}^n$, and $T \geq n$ be a time bound encoded in binary. There is a 3-CNF formula $\varphi_{x,T}(z)$ with $\ell = O(T^2)$ clauses over $\ell = O(T^2)$ variables such that it is satisfiable if and only if $M(x)$ halts in T steps and accepts. In particular, if $M(x)$ halts in T steps and accepts, there is a unique satisfying assignment for $\varphi_{x,T}(z)$.*

Moreover, for some constant $d \in \mathbb{N}$ depending on M, there is an AC^0_d circuit $C_{x,T}$ of size $\mathrm{poly}(n, \log T)$ such that given $(i_1, i_2, i_3, c_1, c_2, c_3) \in [\ell^3] \times \{0,1\}^3$, it outputs 1 if and only if there is a clause containing the i_1-th, the i_2-th, and i_3-th variables in $\varphi_{x,T}$, and it is not satisfied after we set them to be $\neg c_1, \neg c_2, \neg c_3$, respectively.

Furthermore, there is a uniform algorithm in time $\mathrm{poly}(n, \log T)$ such that given x, T, it outputs the description of the circuit $C_{x,T}$.

Proof Sketch. Consider an $O(T) \times T$ computation tableaux whose i-th column is supposed to be the configuration of $M(x)$ on the i-th step (see, e.g., [AB09]). For each entry in the tableax, we add $O(1)$ clauses to check the correctness of its local transition. We introduce $O(T)$ clauses to check that the input configuration is correct, and $O(T)$ clauses to check that the final configuration is a halting and accepting configuration. The uniqueness of the satisfying assignment follows from a careful design of the clauses.

We then show that the circuit $C_{x,T}$ is an AC^0 circuit of size $\mathrm{poly}(n, \log T)$. Assume that the computation tableaux is of size $H \times T$, where $H = O(T)$. Without loss of generality, we assume that both $H = 2^h$ and $T = 2^t$ are a power of two. Moreover, we assume that for each entry $(j,k) \in [H] \times [T]$ of the computation tableaux, we need $S = 2^s$ variables and $C = 2^c$ clauses to check local consistency based on the finite control of the Turing machine M, where $S, C = O(1)$. Let $\mathsf{bin}(x)$ be the binary encoding of x. Given an index i, we can locate the i-th variable, i.e., knowing that it is the l-th variable for the entry (j,k) in the computation tableaux, where $\mathsf{bin}(i) = \mathsf{bin}(j) \circ \mathsf{bin}(k) \circ \mathsf{bin}(l)$. For simplicity, we identify the index i and the location (j,k,l) of the i-th variable.

We now describe in more detail how the 3-CNF formula $\varphi_{x,T}$ is defined. For each (j,k), the variable $z_{(j,k,1)}$ is supposed to be the symbol on the computation tableaux at (j,k), or equivalently, the j-th symbol on the tape in the k-th step of the execution of $M(x)$; the variable $z_{(j,k,2)} = 1$ if and only if the head is at this location; the variable $z_{(j,k,3)} = 1$ if and only if the location is not blank (i.e. $\perp$); the other variables encode in binary the internal state in case that $z_{(j,k,2)} = 1$, and are all zero if $z_{(j,k,2)} \neq 0$. It can be verified that under this encoding, the circuit $C_{x,T}(i_1, i_2, i_3, c_1, c_2, c_3)$ can be constructed using $O(1)$ addition, subtraction, and comparison operations over numbers encoded in binary, and therefore can be simulated by (polynomial-time uniform) AC^0 circuits. $\square$

Algebrization of circuits. We now describe the standard algebrization of circuits. Let $C : \{0,1\}^n \to \{0,1\}$ be an AC^0 circuits of size s and depth d, $\mathbb{F} = \mathbb{F}_q$ be a finite field, and $H \subseteq \mathbb{F}$ be a subset of size h.

For simplicity, we will choose $h = 2^k$ to be a power of two, and thus we can identify H and $\{0,1\}^k$. Let $I : H \to \{0,1\}^k$ be any bijection. We define $I_H : \mathbb{F} \to \mathbb{F}^k$ be the polynomial

$$I_H(z)_j = \sum_{u \in H} \prod_{u' \in H \setminus \{u\}} \frac{I(u)_j (z - u')}{u - u'}.$$

Notice that $I_H(z)$ is of degree at most $|H|$, and for every $u \in H$, $I_H(u) = I(u) \in \{0,1\}^k$.

Without loss of generality we assume that k divides n. Let $r := n/k$. From the function $I : H \to \{0,1\}^k$ we described above, we can induce a bijection between H^r and $\{0,1\}^n$; we identify H^r and $\{0,1\}^n$ using the bijection. We will need the following standard algebrization of circuits.

Lemma C.8. *There is a unique polynomial $\hat{C} \in \mathsf{RM}_{r,s^d \cdot h,q}$ such that for every $x \in H^r$, $\hat{C}(x) = C(x) \in \{0,1\}$. Moreover, there is a polynomial-time algorithm such that given the description of C and any $x \in \mathbb{F}^r$, it outputs $\hat{C}(x) \in \mathbb{F}$.*

Proof Sketch. By replacing AND gates using multiplications and NOT gates using $x \mapsto 1 - x$, we can construct a polynomial $P : \mathbb{F}^n \to \mathbb{F}$ of degree s^d such that for every $z = (z_1, \ldots, z_n) \in \mathbb{F}^n$ such that $z_1, \ldots, z_n \in \{0,1\}$, $P(z) = C(z) \in \{0,1\}$. The polynomial $\hat{C}$ is constructed as follows. Let $x = (x_1, \ldots, x_r) \in \mathbb{F}^r$, we define

$$\hat{C}(x) = P(I_H(x_1), I_H(x_2), \ldots, I_H(x_r)).$$

The correctness of the construction is easy to verify. □

C.5 Putting Things Together

Now we are ready to prove Theorem 6.2 by combining the Cook-Levin reduction from verification of computation of satisfiability (see Theorem C.7), the algebrization of circuits (see Lemma C.8), and the protocols for ZoS (see Theorem C.6) and low-degree testing (see Theorem C.4).

Theorem 6.2 (Restated)**.** *There is a constant $\alpha \in (0,1)$ such that for any Turing machine M, there is a constant $c \geq 1$ and a probabilistic polynomial-time oracle verifier $V_M^{\mathcal{O}}$ satisfying the following. Let $x \in \{0,1\}^n$, $T \geq n$ be a time bound encoded in binary, $r, h \geq 1$ and q be a power of a prime $p = O(1)$, such that $r = \Theta(\log T / \log\log T)$, $h \geq n^c \cdot T^{c/r}$, $h^c \leq q \leq T$.*

- *Given input (x, T, r, h, q) to the verifier $V_M^{\mathcal{O}}$, the proof oracle $\mathcal{O}$ is supposed to be a sequence of polynomials $f_1, f_2, \ldots, f_{6r+8} \in \mathsf{RM}_{3r+3,h,q}$. The verifier tosses $O((r+h)\log q)$ random coins, generates $k = O(rh)$ non-adaptive queries $(i_1, x_1), (i_2, x_2), \ldots, (i_k, x_k) \in [6r+8] \times \mathbb{F}_q^{3r+3}$, and decides in* $\mathrm{poly}(r, h, \log q)$ *time whether to accept the proof given answers $f_{i_1}(x_1), f_{i_2}(x_2), \ldots, f_{i_k}(x_k) \in \mathbb{F}_q$.*
- *(Completeness). If $M(x)$ halts in T steps and accepts, there is a* unique *oracle $\mathcal{O}^*$ such that* $\Pr[V_M^{\mathcal{O}^*}(x, T, r, h, q) = 1] = 1$. *We call this oracle $\mathcal{O}^* = (f_1^*, f_2^*, \ldots, f_{6r+8}^*)$ the* canonical proof *corresponding to the input (x, T, r, h, q).*
- *(Soundness). If $M(x)$ does not halt in T steps, or $M(x)$ rejects, then for every oracle $\mathcal{O} = (f_1, f_2, \ldots, f_{6r+8})$,* $\Pr[V_M^{\mathcal{O}}(x, T, r, h, q) = 1] \leq 1 - \alpha$.
- *(Strong soundness). If $M(x)$ halts in T steps and accepts, then for every oracle $\mathcal{O} = (f_1, f_2, \ldots, f_{6r+8})$, if for any constant $\delta \in (0,1)$, f_i is δ-far from the i-th polynomial f_i^* in the canonical proof for some $i \in [6r+8]$, then*

  $$\Pr[V_M^{\mathcal{O}}(x, T, r, h, q) = 1] \leq 1 - \alpha \cdot \delta,$$

  where $\alpha \in (0,1)$ is a universal constant.

Proof. Now we fix a machine M, an input $x \in \{0,1\}^n$, and $T \geq n$. We will choose the constant c to be sufficiently large, and the absolute constant $\alpha \in (0,1)$ to be sufficiently small. Let $\varphi(z) = \varphi_{x,T}(z)$ be the 3-CNF formula, $\ell = O(T^2)$, and $C = C_{x,T} : [\ell]^3 \times \{0,1\}^3 \to \{0,1\}$ be the AC^0 circuit in Theorem C.7. Let $r = \Theta(\log T / \log\log T)$, $h \geq n^c \cdot T^{c/r}$, $q \in [h^c, T]$, and $\mathbb{F} = \mathbb{F}_q$.

Algebrization. Let $H \subseteq \mathbb{F}$ be a subset such that $|H|^r \geq \ell$ and $\{0,1\} \subseteq H$. Without loss of generality, we assume that $|H|$ is a power of two and $|H|^r = \ell$. Fix any bijection $I : H \to \{0,1\}^{\log|H|}$ such that $I(0) = 0^{\log|H|}$ and $I(1) = 0 \circ 1^{\log|H|-1}$ and thus we can identify H and $\{0,1\}^{\log|H|}$. Let C' be an AC^0 circuit with input length $3\log\ell + 3\log|H|$, such that for each $(i_1, i_2, i_3, c_1, c_2, c_3) \in (\{0,1\}^{\log\ell})^3 \times (\{0,1\}^{\log|H|})^3$, it outputs 0 if $c_1, c_2, c_3 \notin \{0,1\}$, and output $C(i_1, i_2, i_3, c_1, c_2, c_3)$ otherwise[44]. By Theorem C.7, we know that $M(x)$ halts in T steps and accepts if and only if there is an $e : [\ell] \to \{0,1\}$ such that for every $i_1, i_2, i_3 \in H^r$, $c_1, c_2, c_3 \in H$,

$$C'(i_1, i_2, i_3, c_1, c_2, c_3) \cdot (e(i_1) - c_1) \cdot (e(i_2) - c_2) \cdot (e(i_3) - c_3) = 0.$$

Moreover, if $M(x)$ halts in T steps and accepts, the function $e : [\ell] \to \{0,1\}$ satisfying the property above is unique.

Let $\hat{C} : \mathbb{F}^r \to \mathbb{F}$ be the polynomial in Lemma C.8 for the circuit C'; it satisfies that for every $u \in (H^r)^3 \times H^3$, where we identify H^r and $[\ell]$, $\hat{C}(u) = C'(u) \in \{0,1\}$. For a function $e : [\ell] \to \{0,1\}$, we define $\hat{e} \in \mathsf{RM}_{r, r(|H|-1), q}$ be the unique polynomial such that $\hat{e}(u) = e(u) \in \{0,1\}$ for every $u \in H^r$, i.e.,

$$\hat{e}(x_1, \ldots, x_r) := \sum_{v_1, \ldots, v_r \in H} \prod_{v_1' \neq v_1, \ldots, v_r' \neq v_r} e(v_1, \ldots, v_r) \frac{\prod_j (x_j - v_j')}{\prod_j (v_j - v_j')}.$$

From the discussion above, we know that $M(x)$ halts in T steps and accepts if and only if the polynomial $F_{\hat{e}} : \mathbb{F}^{3r+3} \to \mathbb{F}$ defined as

$$F_{\hat{e}}(z_1, z_2, z_3, b_1, b_2, b_3) := \hat{C}(z_1, z_2, z_3, b_1, b_2, b_3) \cdot (\hat{e}(z_1) - b_1) \cdot (\hat{e}(z_2) - b_2) \cdot (\hat{e}(z_3) - b_3) \tag{5}$$

is zero on the subcube H^{3r+3}, where $z_1, z_2, z_3 \in \mathbb{F}^r$ and $b_1, b_2, b_3 \in \mathbb{F}$. Moreover, recall that $\hat{C}$ is of degree at most $|H| \cdot \mathrm{poly}(n, \log T)$ and $\hat{e}$ is of degree at most $r(|H|-1)$, we know that $F_{\hat{e}}$ is of degree at most $|H|^4 \cdot r \cdot \mathrm{poly}(n, \log T)$. Note that

$$\left(|H|^4 \cdot r \cdot \mathrm{polylog}\, T\right)^r \leq \ell^4 \cdot (\log T)^{O(r)} \leq T^c \;\Rightarrow\; |H|^4 \cdot r \cdot \mathrm{poly}(n, \log T) \leq h$$

for sufficiently large c, and therefore $F_{\hat{e}} \in \mathsf{RM}_{r,h,q}$.

Description of V_M. Now we are ready to describe the verifier V_M formally. It is given the oracle access to a sequence of polynomials $f_1, f_2, \ldots, f_{6r+8} : \mathbb{F}^{3r+3} \to \mathbb{F}$. It is supposed to be as follows: f_1 is the encoded assignment $\hat{e}$ for some assignment e for $\varphi(z)$, f_2 is the polynomial $F_{\hat{e}}$ defined in Equation (5), and $f_2, \ldots, f_{6r+8}$ are supposed to be the division witness of $F_{\hat{e}} \in \mathsf{ZoS}_{3r+3,h,q,H}$. A caveat is that as $\hat{e}$ is an r-variate polynomial while f_1 is a $(3r+3)$-variate polynomial, the individual degrees of the last $2r+3$ variables in f_1 are supposed to be 0. The verifier works as follows:

[44]Here, $c_1, c_2, c_3 \in \{0,1\}$ means that $c_1, c_2, c_3 \in \{0^{\log|H|}, 0 \circ 1^{\log|H|-1}\}$; recall that we identify H and $\{0,1\}^{\log|H|}$ by the bijection I fixed above.

1. (Low-degree check). Run the algorithm in Theorem C.4 on f_1 and f_2 to check that they are of degree at most $r(|H|-1)$ and h, respectively. We then uniformly sample $x = (x_1, \ldots, x_{3r+3}) \in \mathbb{F}^{3r+3}$ and $u \in \mathbb{F}$ and perform the following check:

   (a) (Individual degree check). For every $i \in [r+1, 3r+3]$, we define

   $$x^i := (x_1, \ldots, x_{i-1}, u, x_{i+1}, \ldots, x_{3r+3})$$

   and check whether $f_1(x^i) = f_1(x)$.

2. (Consistency check). For a uniformly random $x \in \mathbb{F}^{3r+3}$, let $x = (z_1, z_2, z_3, b_1, b_2, b_3) \in (\mathbb{F}^r)^3 \times \mathbb{F}^3$, check

   $$f_2(x) = F_{f_1}(x) := \hat{C}(z_1, z_2, z_3, b_1, b_2, b_3) \cdot (f_1(z_1) - b_1) \cdot (f_1(z_2) - b_2) \cdot (f_1(z_3) - b_3),$$

   where in $f_1(z_i)$ we fix all but the first r variables to be 0.

3. (ZoS check). Run the algorithm in Theorem C.6, where f is instantiated with $F_{\hat{e}} \in \mathsf{RM}_{3r+3,h,q}$ and P, Q are instantiated with $f_2, f_3, \ldots, f_{6r+8}$.

The verifier accepts if it passes all the checks. Note that the randomness complexity, query complexity, and decision complexity of V_M is easy to verify.

Completeness and the canonical proof. Suppose that $M(x)$ halts in T steps and accepts, we know by the discussion above that there is a unique function $e : [\ell] \to \{0,1\}$ (representing an assignment for φ) such that $\varphi(e) = 1$, and thus $F_{\hat{e}} \in \mathsf{ZoS}_{3r+3,h,q,H}$. We define the *canonical proof* as $f_1^* := e$, $f_2^* := F_{\hat{e}}$, and $f_3^*, \ldots, f_{6r+8}^*$ to be the division witness of $F_{\hat{e}} \in \mathsf{ZoS}_{3r+3,h,q,H}$. By the completeness of Theorem C.6, it is easy to see that if $M(x)$ halts in T steps and accepts, the verifier V_M given the canonical proof accepts with probability 1.

Soundness and strong soundness. Now we prove the soundness and strong soundness of the verifier V_M. Fix any oracle $\mathcal{O} = (f_1, \ldots, f_{6r+8})$ and let $f_1' \in \mathsf{RM}_{3r+3,r(|H|-1),q}, f_2' \in \mathsf{RM}_{3r+3,h,q}$ be the closest polynomials to f_1 and f_2, respectively. Let $\delta \le 1/8$ be a constant.

Case 1 (failing low-degree check). Suppose that f_1 is $(\delta/2)$-far from f_1', or f_2 is $(\delta/2)$-far from f_2', then by Theorem C.4, the algorithm rejects with probability at least $\delta/80$, which is at least $\alpha\delta$ if we set $\alpha < 1/80$.

Case 2 (failing individual degree check). Suppose that there is an $i \in [r+1, 3r+3]$ such that the i-th variable has individual degree at least 1 in f_1'. Fix this i. By the union bound, we know that with probability at least $1/2$ over a uniformly random $x = (x_1, \ldots, x_{3r+3}) \in \mathbb{F}^{3r+3}$ and $u \in \mathbb{F}$, let $x^i := (x_1, \ldots, x_{i-1}, u, x_{i+1}, \ldots, x_{3r+3})$, we will have $f_1'(x) = f_1(x)$ and $f_1'(x^i) = f_1(x^i)$. Since f_1' is of degree at most h, we know that if we set all but the i-th variables uniformly at random, with probability at least $1 - h/q = 1 - o(1)$, the obtained univariate polynomial will be a non-zero univariate polynomial of degree at most h. This means that with probability $1 - o(1)$, we will have $f_1'(x) \neq f_1'(x^i)$, and by the union bound, we know with probability at least $1/2 - o(1)$, $f_1(x) \neq f_1(x^i)$ and the verifier will reject.

Case 3 (failing consistency check). Suppose that $f_2' \neq F_{f_1'}$, then by Schwartz-Zippel lemma (see Lemma C.2) we know that $\delta(f_2', F_{f_1'}) \geq h/q = 1 - o(1)$. Since $\delta(f_1, f_1') \leq 1/8$, $\delta(f_2, f_2') \leq 1/8$, and $F_{f_1}(x)$ makes three queries to f_1 where each query is uniformly distributed, we know by the union bound that over a uniformly random $x \in \mathbb{F}^{3r+3}$, with probability at least $1/2 - o(1)$, $f_2(x) \neq F_{f_1}(x)$ and thus the consistency check fails. The rejection probability is at least α if we set $\alpha < 1/2 - o(1)$.

Case 4 (soundness). Suppose that $M(x)$ does not halt in T steps, or $M(x)$ rejects, we know that $\varphi(z)$ (from Theorem C.7) is unsatisfiable. Let α' be the constant α in Theorem C.6. We will set $\alpha < \alpha'/10$. Suppose, towards a contradiction, that the ZoS check passes with probability larger than α, then we know that f_2 is not (α/α')-far from a degree-h polynomial f_2'' that is zero on the subcube H^{3r+3}. Therefore, $\delta(f_2', f_2'') \leq 1/8 + \alpha/\alpha' < 1 - h/q$, and thus by Schwartz-Zippel lemma (see Lemma C.2) we know that $f_2' = f_2''$. Furthermore, since $f_2' = F_{f_1'}$ by Case 3, we know by the definition of $F_{f_1'}$ that there is an assignment that makes $\varphi(z)$ accepts, which leads to a contradiction.

Case 5 (strong soundness). Suppose that $M(x)$ halts in T steps and accepts, but for some $i \in [6r+8]$, $\delta(f_i, f_i^*) > \delta$. Fix i to be the smallest such index. Let α' be the constant in Theorem C.6 and $\alpha < \alpha'/10$, we consider the following cases.

1. Suppose that $i = 1$, we will show that f_2 is (α/α')-far from being a degree-h polynomial f_2'' that is zero on the subcube H^{3r+3}, and thus the ZoS check fails with probability at least α. Towards a contradiction we assume that $\delta(f_2, f_2'') \leq \alpha/\alpha'$, then $\delta(f_2', f_2'') \leq 1/8 + \alpha/\alpha' < 1 - h/q$, therefore by Schwartz-Zippel lemma (see Lemma C.2), $f_2' = f_2''$. Recall that by Case 3 we have $f_2' = F_{f_1'}$, and since f_2' is zero on the subcube H^{3r+3}, we know that for every $z_1, z_2, z_3 \in H^r$ and $b_1, b_2, b_3 \in \{0,1\}$,

$$C(z_1, z_2, z_3, b_1, b_2, b_3) \cdot (f_1'(z_1) - b_1) \cdot (f_1'(z_2) - b_2) \cdot (f_1'(z_3) - b_3) = 0,$$

   which further means that f_1' restricting to H^r (i.e. $[\ell]$) is a satisfying assignment of $\varphi(z)$. Note that by Theorem C.7 the satisfying assignment is unique. Since f_1' is of degree $r(|H| - 1)$, we know by the uniqueness of low-degree extension[45] that $f_1' = f_1^*$, and thus $\delta(f_1, f_1^*) \leq \delta/2$, a contradiction.

2. Suppose that $i > 1$, then by $\delta(f_1, f_1') \leq \delta/2$ we know that $f_1' = f_1^*$. Note that after Case 3 we know that $f_2' = F_{f_1'} = F_{f_1^*} = f_2^*$, which implies that $i > 2$. Recall that f_2^* is zero on the subcube H^{3r+3}. By the strong soundness of the ZoS check (see Theorem C.6), we know that it must fail with probability at least $\delta\alpha' \geq \delta\alpha$.

In either case, the verifier rejects with probability at least $\delta\alpha$. □

D On the RMV Generator

In this section, we sketch the proof of Theorem 6.1 to complete the proof in Section 6.3. In the following, we assume the reader is familiar with the proofs of Theorem 3.5 in [SU07].

[45] Recall that by Case 2 we already ensure that all but the first r variables of f_1' are of degree 0.

Theorem 6.1 (Restated)**.** *Let r, d and h be parameters such that r is a power of d and h is a prime power. Suppose $d = O(1)$ and $h = \mathrm{poly}(r)$. Let q be a prime power with $h^{100} \le q \le 2^{h^{O(1)}}$ and h be a parameter with $h^{1/100} \le m \le q^{1/100}$. There is an algorithm RMV and a pair of Arthur-Merlin protocols $(\sigma_{\mathsf{c}}, \sigma_{\mathsf{e}})$ described as follows.*

- (Locality). *Let $p \in \mathsf{RM}_{r,h,q}$. There is an oracle algorithm $\mathsf{RMV}_{h,d}$ that takes a seed $z \in \{0,1\}^{O(r \log q)}$ and p as oracle, outputs a string in $\{0,1\}^m$ in time $\mathrm{poly}(m)$. The collection of all $\mathsf{RMV}^p_{h,d}(z)$ is intended to be a hitting set for coAM circuits.*
- *σ_{c} takes a coAM circuit $D : \{0,1\}^m \to \{0,1\}$ as input, and outputs a string $\alpha \in \{0,1\}^\ell$ called the* commitment *in time $\mathrm{poly}(|D|, \ell)$, where $\ell = \mathrm{poly}(m)$.*
- *σ_{e} takes $x \in \mathbb{F}_q^r$, the circuit D, and the commitment $\alpha \in \{0,1\}^\ell$ (which is intended to be generated by σ_{c}), and outputs some $y \in \mathbb{F}_q$ in time $m^{O(d \log_d^2 r)}$ and $O(1)$ rounds.*

The algorithms satisfy the following properties.

- (Conformity). *If D rejects every element from $\mathsf{RMV}_{h,d}(p)$, then there is a pair of strategies $(\tau_{\mathsf{c}}, \tau_{\mathsf{e}})$ of Merlin in σ_{c} and σ_{e} such that given $x \in \mathbb{F}_q^r$,*

$$\Pr\left[\sigma_{\mathsf{e}}^{\tau_{\mathsf{e}}}(x, D, \alpha := \sigma_{\mathsf{c}}^{\tau_{\mathsf{c}}}(D)) = p(x)\right] = 1.$$

  Moreover, τ_{c} and τ_{e} can be simulated by deterministic polynomial-time (in the communication complexity of σ_c and σ_{e}, respectively) machines given oracle accesses to p.[46]

- (Resiliency). *If D rejects at most a $1/3$-fraction of its inputs, then for any commitment $\alpha \in \{0,1\}^\ell$, there is a unique $g_\alpha \in \mathbb{F}_q^r \to \mathbb{F}_q$ such that for every $x \in \mathbb{F}_q^r$ and every strategy τ_{e} of Merlin,*

$$\Pr\left[\sigma_{\mathsf{e}}^{\tau_{\mathsf{e}}}(x, D, \alpha) \in \{g_\alpha(x), \bot\}\right] \ge 1 - o(1).$$

Theorem 6.1 is almost identical to Theorem 3.5, with two differences below:

- In Theorem 6.1, the hitting set is required to be *locally constructable*: Given any seed $z \in \{0,1\}^{O(r \log q)}$, we can compute the element in the hitting set indexed by z in $\mathrm{poly}(m)$ time.
- In Theorem 6.1, we allow an exponentially large field size $q \le 2^{r^{O(1)}}$ and an exponentially long output length $m \le q^{1/100}$.

Below, we will first review the proof of Theorem 3.5 in [SU07], and then show how to modify the proof to support the two additional requirements.

The construction in [SU07]. The construction relies on the following local extractor.

Definition D.1 (Local extractor for subsets [SU07, Definition 3.5])**.** Let C be a set, $t, m \in \mathbb{N}$. A (k, ε) local C-extractor is an oracle function $E \colon \{0,1\}^t \to \{0,1\}^m$ for which the following holds:

1. for every random variable X distributed on C with min-entropy at least k, $E^X(U_t)$ is ε-close to uniform, and

[46] Recall that the notation σ^τ means the (probabilistic) output of the protocol when the prover sends messages according to the strategy τ.

2. E runs in $\mathrm{poly}(m,t)$ time.

Lemma D.2 ([SU07, Lemma 3.7], implicit in [SU05])**.** *Fix parameters $r < h$, and let $C = \mathsf{RM}_{r,h,q}$ be a Reed-Muller code. Set $k = h^5$. There is an explicit $(k, 1/k)$ local C-extractor E with seed length $t = O(r \log q)$ and output length $h = k^{1/5}$.*

Based on the local extractor, given a polynomial $p : \mathbb{F}_q^r \to \mathbb{F}_q$ of degree h, the hitting set $\mathsf{RMV}_{h,d}(p)$ is constructed recursively. Specifically, it is the union of the following two parts:

- (**Direct part**). As $p \in \mathsf{RM}_{r,h,q}$ is a Reed-Muller codeword, the local extractor in Lemma D.2 defines a function $E^p : \{0,1\}^t \to \{0,1\}^m$ for $t = O(r \log q)$. We put all the 2^t outputs of E^p into the hitting set $\mathsf{RMV}_{h,d}(p)$.
- (**Recursive part**). If $r > d$, let $B = \mathbb{F}^{r/d}$. By grouping each consecutive r/d variables, we can view p as a function $p : B^d \to \mathbb{F}$. Consider all $dq^{r-r/d}$ possible functions $p_L : B \to \mathbb{F}$ obtained by fixing all but one variables in p to arbitrary values in B. We also put all the hitting sets $\mathsf{RMV}_{h,d}(p_L)$ for these functions $p_L : \mathbb{F}^{r/d} \to \mathbb{F}$ into $\mathsf{RMV}_{h,d}(p)$.

This recursion process defines the hitting set $\mathsf{RMV}_{h,d}(p)$. The recursion tree has depth $\log_d r$, where a node with depth ℓ has at most dq^{r/d^ℓ} children, hence the total number of nodes is bounded by $\prod_{\ell=0}^{\log_d r-1}(dq^{r/d^\ell}) = d^{\log_d r} \cdot q^{\sum_{\ell=0}^{\log_d r-1} r/d^\ell} \le q^{O(r)}$. As each node contributes a direct part of size at most $2^t = q^{O(r)}$, the hitting set $\mathsf{RMV}_{h,d}(p)$ has at most $q^{O(r)}$ elements, and can be computed within time $q^{O(r)}$. For every node u of the recursion tree, we use H_u to denote the direct part this nodes contributes to $\mathsf{RMV}_{h,d}(p)$. (Thus, $\mathsf{RMV}_{h,d}(p)$ is simply the union of all the H_u.)

Locality of the hitting set. In the following, we will argue that the locality of the hitting set $\mathsf{RMV}_{h,d}(p)$ follows from the locality of the extractor in Lemma D.2.

First, we assign an $O(r \log q)$-bit index for each of the $q^{O(r)}$ elements in $\mathsf{RMV}_{h,d}(p)$: the prefix of $O(r \log q)$ bits specifies the node u in the recursion tree such that H_u contains this element, and the remaining $t = O(r \log q)$ bits is the index of this element within H_u.

Then, given an seed $z \in \{0,1\}^{O(r \log q)}$, we can compute the element in $\mathsf{RMV}_{h,d}(p)$ with index z in $\mathrm{poly}(m)$ time. We first determine a node u in the recursion tree by z's prefix z_{pre} within time $\mathrm{poly}(r \log q)$. Then, we obtain the oracle for the polynomial p_u at u by fixing some of the variables in the oracle of p. Finally, assuming oracle access to p_u, the elements indexed by z in the hitting set can be computed from the t-bit suffix z_{suf} of z in H_u via the local extractor $E^{p_u}(z_{\mathrm{suf}})$ within time $\mathrm{poly}(m, r \log q)$, as desired.

Larger field size and output lengths. We also claim that all the arguments in [SU07] naturally generalize to support a larger field size and a longer output length.

The output length is controlled by the local extractor in Lemma D.2, which by default outputs strings of length h. To use Lemma D.2 to output longer strings, we view the set $C = \mathsf{RM}_{r,h,q}$ as a subset of $\mathsf{RM}_{r,\max\{h,m\},q}$ and set parameter $k = \max\{h,m\}^5$. Then, by Lemma D.2, we have a $(k, 1/k)$ local $\mathsf{RM}_{r,h,q}$-extractor with output length m.[47] Moreover, increasing k to $\max\{h,m\}^5$ would not cause an unaffordable running time or inefficient reconstruction, as in Theorem 6.1 the running times in all the efficiency conditions are measured in m instead of h.

[47] If $m < h$, the extractor in Lemma D.2 has output length $\max\{h,m\} = h$; we can simply retain the first m bits of each output.

To support a larger field size $q = 2^{h^{O(1)}}$, we note that in the proof of [SU07], whenever the running time depends on the field size, the dependence is a multiplicative factor of $\mathrm{polylog}(q) = h^{O(1)}$, which is consistent with Theorem 6.1. The only exception is in the low-degree test: [SU07] uses a low-degree test with running time $\mathrm{poly}(q, r)$ for functions in $\mathbb{F}_q$ with r variables. By replacing this low-degree test with a faster one (e.g., Theorem C.4), the running time is reduced to $\mathrm{poly}(r, h, \log q) = \mathrm{poly}(h)$ too, as desired.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il