Electronic Colloquium on Computational Complexity, Report No. 184 (2025)

New Algebrization Barriers to Circuit Lower Bounds via Communication Complexity of Missing-String

Lijie Chen* Yang Hu† Hanlin Ren‡

November 18, 2025

Abstract

The *algebrization barrier*, proposed by Aaronson and Wigderson (STOC '08, ToCT '09), captures the limitations of many complexity-theoretic techniques based on arithmetization. Notably, several circuit lower bounds that overcome the relativization barrier (Buhrman–Fortnow–Thierauf, CCC '98; Vinodchandran, TCS '05; Santhanam, STOC '07, SICOMP '09) remain subject to the algebrization barrier.

In this work, we establish several new algebrization barriers to circuit lower bounds by studying the communication complexity of the following problem, called XOR-Missing-String: For $m < 2^{n/2}$, Alice gets a list of m strings $x_1, \ldots, x_m \in \{0,1\}^n$, Bob gets a list of m strings $y_1, \ldots, y_m \in \{0,1\}^n$, and the goal is to output a string $s \in \{0,1\}^n$ that is not equal to $x_i \oplus y_j$ for any $i, j \in [m]$.

1. We construct an oracle A_1 and its multilinear extension $\widetilde{A_1}$ such that $\mathsf{PostBPE}^{\widetilde{A_1}}$ has linear-size A_1-oracle circuits on infinitely many input lengths. That is, proving $\mathsf{PostBPE} \not\subseteq$ i.o.-$\mathsf{SIZE}[O(n)]$ requires non-algebrizing techniques. This barrier follows from a $\mathsf{PostBPP}$ communication lower bound for XOR-Missing-String. This is in contrast to the well-known algebrizing lower bound $\mathsf{MA_E} (\subseteq \mathsf{PostBPE}) \not\subseteq \mathsf{P}/_{\mathrm{poly}}$.
2. We construct an oracle A_2 and its multilinear extension $\widetilde{A_2}$ such that $\mathsf{BPE}^{\widetilde{A_2}}$ has linear-size A_2-oracle circuits on all input lengths. Previously, a similar barrier was demonstrated by Aaronson and Wigderson, but in their result, $\widetilde{A_2}$ is only a *multiquadratic* extension of A_2. Our results show that communication complexity is more useful than previously thought for proving algebrization barriers, as Aaronson and Wigderson wrote that communication-based barriers were "more contrived". This serves as an example of how XOR-Missing-String forms new connections between communication lower bounds and algebrization barriers.
3. Finally, we study algebrization barriers to circuit lower bounds for $\mathsf{MA_E}$. Buhrman, Fortnow, and Thierauf proved a *sub-half-exponential* circuit lower bound for $\mathsf{MA_E}$ via algebrizing techniques. Toward understanding whether the half-exponential bound can be improved, we define a natural subclass of $\mathsf{MA_E}$ that includes their hard $\mathsf{MA_E}$ language, and prove the following result: For every *super-half-exponential* function $h(n)$, we construct an oracle A_3 and its multilinear extension $\widetilde{A_3}$ such that this natural subclass of $\mathsf{MA_E}^{\widetilde{A_3}}$ has $h(n)$-size A_3-oracle circuits on all input lengths. This suggests that half-exponential might be the correct barrier for $\mathsf{MA_E}$ circuit lower bounds w.r.t. algebrizing techniques.

*University of California, Berkeley. lijiechen@berkeley.edu.
†Institute for Interdisciplinary Information Sciences, Tsinghua University. y-hu22@mails.tsinghua.edu.cn.
‡Institute for Advanced Study. h4n1in.r3n@gmail.com.

ISSN 1433-8092

Contents

1 Introduction

Proving unconditional circuit lower bounds is one of the major challenges in theoretical computer science, with the holy grail of proving $\mathsf{NP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$. Unfortunately, our progress toward this goal is barely satisfactory, as it is even open to prove a super-polynomial size lower bound for huge, exponential-time classes such as NEXP. Only for even larger complexity classes such as $\Sigma_2\mathsf{EXP}$ [Kan82], which are in (or beyond) the second level of exponential-time hierarchy, are super-polynomial lower bounds known.

Our lack of progress in proving circuit lower bounds is partially explained by a series of *barrier results* such as relativization [BGS75], natural proofs [RR97], and algebrization [AW09]. Among these barriers, relativization and algebrization are particularly relevant to lower bounds against *unrestricted circuits* for *large* complexity classes. For example, Wilson [Wil85] constructed an oracle world where P^{NP} has linear-size circuits, which explains our inability to prove fixed-polynomial size lower bounds for P^{NP}.

Missing-String: a duality-based approach to relativizing circuit lower bounds. Previous relativization barriers for circuit lower bounds are proved in an *ad hoc* fashion, which involves carefully analyzing the interaction between the oracle and the machines to be diagonalized [Wil85, BFT98, Aar06]. A recent paper by Vyas and Williams [VW23] introduced the Missing-String problem as a systematic approach to such relativization barriers:

Problem 1.1 (Missing-String(n, m))**.** *Let n, m be such that $m < 2^n$. Given (query access to) a list of length-n strings $x_1, x_2, \ldots, x_m$, output a length-n string that is not in this list.*

The query complexity of Missing-String captures relativizing circuit lower bounds in the following sense: relativization barriers to proving $\mathcal{C}\text{-}\mathsf{EXP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ are essentially $\mathcal{C}^{\mathrm{dt}}$ lower bounds for Missing-String. (Here, $\mathcal{C}\text{-}\mathsf{EXP}$ is the exponential-time version of $\mathcal{C}$ and $\mathcal{C}^{\mathrm{dt}}$ means the decision-tree version of $\mathcal{C}$.) This connection was made explicit in [VW23], who demonstrated an equivalence between relativization barriers to exponential lower bounds for $\Sigma_2\mathsf{E}$ and the non-existence of small depth-3 circuits for Missing-String.[1]

Intriguingly, Missing-String connects *circuit lower bounds* with *circuit upper bounds* and provides a *duality*-based approach to both: Decision tree upper bounds for Missing-String imply relativizing circuit lower bounds, and decision tree lower bounds for Missing-String imply relativized worlds with circuit upper bounds (i.e., relativization barrier to circuit lower bounds). Besides providing a clean and systematic method for relativization barriers, this "algorithmic" perspective has indeed made progress in circuit lower bounds: By designing algorithms for the *Range Avoidance* problem [KKMP21, Kor21, RSW22], which is the "white-box" version of Missing-String,[2] recent work [CHR24, Li24] proved an exponential-size, relativizing lower bound for the complexity class $\Sigma_2\mathsf{E}$ (which also implies a quasi-polynomial size depth-3 circuit upper bound for Missing-String, settling the question in [VW23]).

The quest of algebrization. However, the relativization barrier does not capture many circuit lower bounds that follow from *nonrelativizing* results such as $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92] and

[1] $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ denotes *single-exponential* time and $\mathsf{EXP} = \mathsf{DTIME}[2^{n^{O(1)}}]$ denotes *exponential time*; classes such as $\Sigma_2\mathsf{E}$ and $\Sigma_2\mathsf{EXP}$ are defined analogously. Exponential time and single-exponential time are basically interchangeable in the context of super-polynomial lower bounds by a padding argument.

[2] In the Range Avoidance problem, we are given the description of a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ and our goal is to output a string $y \in \{0,1\}^{n+1}$ that is not in the range of C. It is easy to see that *relativizing* algorithms for Range Avoidance are equivalent to decision trees for Missing-String.

$\mathsf{MIP} = \mathsf{NEXP}$ [BFL91, BFNW93], whose proofs are based on nonrelativizing proof techniques such as *arithmetization*. This includes circuit lower bounds for PP [Vin05] and MA [BFT98, San09], both of which are *provably nonrelativizing* [BFT98, Aar06].

To shed light on the limitations of such proof techniques, Aaronson and Wigderson [AW09] proposed the *algebrization* barrier: a lower bound statement $\mathcal{C} \not\subseteq \mathcal{D}$ *algebrizes* if $\mathcal{C}^{\widetilde{A}} \not\subseteq \mathcal{D}^{A}$ for all oracles A and all low-degree extensions $\widetilde{A}$ of A. Aaronson and Wigderson showed that many arithmetization-based results indeed algebrize; in particular, $\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}} \not\subseteq \mathsf{P}^{A}/_{\mathrm{poly}}$ for every oracle A and low-degree extension $\widetilde{A}$ of A [AW09, Theorem 3.17]. In fact, all super-polynomial lower bounds against general circuits that we are aware of are algebrizing. Thus, algebrization is a more suitable framework than relativization for capturing the limitations of our current techniques.

Unfortunately, our understanding of the algebrization barrier remains primitive. For instance, several aspects of the algebrizing lower bound $\mathsf{MA}_{\mathsf{E}} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ [BFT98] remain poorly understood even with respect to algebrizing techniques:

- This lower bound only holds *infinitely-often*. That is, [BFT98] only exhibited a language $L \in \mathsf{MA}_{\mathsf{E}}$ where there are *infinitely many* input lengths on which L has high circuit complexity. Recently, the relativizing infinitely-often lower bound for $\Sigma_2\mathsf{E}$ was improved to almost everywhere by [Li24], and it is tempting to ask whether the same improvement can be made with respect to this lower bound. Does MA_{E} require large circuit complexity on *every* input length? If so, can we show this via algebrizing proof techniques?

- This lower bound is only *sub-half-exponential*. Roughly speaking, a function h is *half-exponential* if $h(h(n)) \approx 2^n$. Half-exponential bounds appear naturally in many win-win analyses in complexity theory [MVW99], and [BFT98] is no exception—it is only known how to prove a size-$h(n)$ lower bound for MA_{E} when h is smaller than half-exponential. The recent "iterative win-win paradigm" [CLO+23, CHR24, CLL25] provides new techniques for overcoming the half-exponential barrier, which has indeed improved the circuit lower bound for $\Sigma_2\mathsf{EXP}$ to a near-maximum $(2^n/n)$ one [CHR24]. Can we use similar techniques to prove an exponential size lower bound for MA_{E}? If so, can we show this via algebrizing proof techniques?

Our lack of understanding on algebrization barriers raises the following question:

Is there a *duality-based approach* to algebrization barriers?

In particular, can we establish algebrization barriers via lower bounds for Missing-String? It follows from [AW09] that *communication* lower bounds for Missing-String imply algebrization barriers to circuit lower bounds.[3] Hence, a naïve attempt is to prove communication lower bounds for Missing-String and translate them into algebrization barriers. Unfortunately, it turns out that Missing-String admits an efficient deterministic communication protocol by a simple binary search.[4]

The main conceptual contribution of this paper is the following communication problem:

[3] In fact, lower bounds for Missing-String in the *algebraic query model* suffices. However, we find the algebraic query model somewhat counter-intuitive to work with (for example, the input to the query algorithms consists of a Missing-String instance along with its *low-degree extension*). The "transfer principle" [AW09, Section 4.3] allows us to simulate such query algorithms by communication protocols, hence we choose to study communication complexity.

[4] Suppose that $m \leq 2^n/10$, Alice has inputs $x_1, x_2, \ldots, x_m \in \{0,1\}^n$ and Bob has inputs $y_1, y_2, \ldots, y_m \in \{0,1\}^n$. For each string $s \in \{0,1\}^n$, it costs only $O(\log m)$ bits of communication to obtain the value $f(s) := |\{i : x_i \leq s\}| + |\{i : y_i \leq s\}|$. Hence, we can use binary search to find two (lexicographically) adjacent strings $\mathrm{pred}(s)$ and s such that $f(\mathrm{pred}(s)) = f(s)$ by communicating $O(n \log m)$ bits. Clearly, s is not in the set $\{x_1, \ldots, x_m, y_1, \ldots, y_m\}$.

Problem 1.2 (XOR-Missing-String(n,m))**.** *Let n, m be such that $m < 2^{n/2}$. Alice receives m strings $x_1, x_2, \ldots, x_m \in \{0,1\}^n$ and Bob receives $y_1, y_2, \ldots, y_m \in \{0,1\}^n$. Their goal is to communicate with each other and output an n-bit string s such that s is not equal to $x_i \oplus y_j$ for every $1 \leq i, j \leq m$. Here $\oplus$ denotes the bit-wise XOR operation over strings.*

We show that XOR-Missing-String is hard for various communication complexity models, and transfer our communication complexity lower bounds to algebrization barriers. We now discuss our results in more detail.

1.1 Barriers to pr-PostBPE Circuit Lower Bounds

Our first result is an algebrization barrier to proving pr-$\mathsf{PostBPE} \not\subseteq$ i. o.-$\mathsf{SIZE}[O(n)]$, i.e., an almost-everywhere circuit lower bound for pr-PostBPE. Here, pr-PostBPE is the class of promise problems decided by a probabilistic exponential-time machine with *postselection* [HHT97]: conditioned on some event (which may happen with exponentially small probability), the machine outputs the correct answer with high probability. Postselection is a powerful computational resource: PostBPP contains MA (thus NP) [HHT97] and PostBQP (polynomial-time *quantum* computation with postselection) equals PP [Aar05].[5]

Theorem 1.3. *There exists an oracle A_1 and its multilinear extension $\widetilde{A_1}$ such that*

$$\text{pr-}\mathsf{PostBPE}^{\widetilde{A_1}} \subseteq \text{i. o.-}\mathsf{SIZE}^{A_1}[O(n)].$$

In particular, this also implies

$$\text{pr-}\mathsf{MA}_{\mathsf{E}}^{\widetilde{A_1}} \subseteq \text{i. o.-}\mathsf{SIZE}^{A_1}[O(n)].$$

The proof of pr-$\mathsf{MA}_{\mathsf{E}} \not\subseteq \mathsf{P}/_{\text{poly}}$ is algebrizing [BFT98, San09, AW09]. Our result implies that improving this circuit lower bound to almost-everywhere requires non-algebrizing techniques.

Theorem 1.3 follows from a PostBPP communication lower bound for XOR-Missing-String:

Theorem 1.4. *Let $n \geq 1$ and $20n \leq m < 2^{n/2}$ be integers. Any PostBPP communication protocol that solves XOR-Missing-String(n,m) with error $\leq 2^{-5n}$ must have communication complexity $\Omega(m)$.*

Assume that $n \ll m \ll 2^{o(n)}$. A trivial protocol for XOR-Missing-String is to output a uniformly random n-bit string, which has communication complexity $O(n)$ and error $m^2/2^n$. Thus, Theorem 1.4 states that even if postselection is allowed, if we want to beat the error bound of this trivial protocol ($\approx 2^{-n}$), then the amount of communication needed is close to the maximum ($\Omega(m)$).

Moreover, since the error of any *pseudodeterministic*[6] communication protocol can be reduced by a 2^{-k} factor by repeating the protocol $\Theta(k)$ times and taking the majority answer, Theorem 1.4

[5] For example, a PostBPP machine can solve NP-complete problems as follows. With exponentially small probability, output "No" and halt. If this does not happen, choose an NP witness uniformly at random and "kill yourself" if this witness is invalid (that is, we *postselect* on the event that our witness is valid). If the algorithm survives, it outputs "Yes." Conditioned on survival, with high probability, our algorithm outputs "Yes" on Yes instances and outputs "No" on No instances.

[6] A randomized protocol for a search problem is *pseudodeterministic* [GG11] if there is a *canonical* output that is correct and outputted with probability $\geq 2/3$. The trivial protocol for XOR-Missing-String that outputs a random guess is not pseudodeterministic, since running it two times using different randomness is likely to yield different answers.

implies that any pseudodeterministic PostBPP protocol that solves XOR-Missing-String(n, m) (with correct probability, say, $2/3$) must have communication complexity $\Omega(m/n)$. This stands in stark contrast to the case without pseudodeterministic constraints, as the trivial protocol is correct with probability $1 - m^2/2^n \gg 2/3$. We also remark that lower bounds against pseudodeterministic PostBPP communication protocols suffice for constructing an oracle A such that $\mathsf{PostBPE}^{\widetilde{A}}$ has small A-oracle circuits; however, we will need the full power of Theorem 1.4 against every low-error protocol to prove that *the promise version* of $\mathsf{PostBPE}^{\widetilde{A}}$ has small A-oracle circuits.

1.2 Barriers to BPE Circuit Lower Bounds

Our second result is an algebrization barrier to proving $\mathsf{BPE} \not\subseteq \mathsf{SIZE}[O(n)]$, i.e., an *infinitely-often* circuit lower bound for BPE:

Theorem 1.5. *There exists an oracle A_2 and its multilinear extension $\widetilde{A_2}$ such that*

$$\mathsf{BPE}^{\widetilde{A_2}} \subseteq \mathsf{SIZE}^{A_2}[O(n)].$$

Previously, Aaronson and Wigderson [AW09] constructed an oracle A and its *multiquadratic* extension $\widetilde{A}$ such that $\mathsf{BPEXP}^{\widetilde{A}} \subseteq \mathsf{P}^A/_{\mathrm{poly}}$. Their proof works with the algebraic query model directly, and it is unclear how to prove the same result for multilinear extensions using their techniques.

Besides demonstrating the power of our communication- and duality-based approach, an algebrization barrier w.r.t. multilinear extension also aligns better with the notion of *affine relativization* [AB18]. A statement $\mathcal{C} \subseteq \mathcal{D}$ is said to *affine relativize* if $\mathcal{C}^{\widetilde{A}} \subseteq \mathcal{D}^{\widetilde{A}}$ for every $\widetilde{A}$ that is the multilinear extension of some oracle A. The crucial difference from algebrization (in the sense of [AW09]) is that the left-hand side is also required to relativize with the multilinear extension. Affine relativization is arguably a cleaner and more robust notion than algebrization.[7] It is important to only take multilinear extensions, as [AB18] was only able to show that, e.g., $\mathsf{PSPACE}^{\widetilde{A}} \subseteq \mathsf{IP}^{\widetilde{A}}$ for every multilinear oracle $\widetilde{A}$; it is unclear if such a statement holds when $\widetilde{A}$ is merely multiquadratic. (This is also why Aaronson and Wigderson [AW09] choose to relativize IP with the low-degree extension of A but to relativize PSPACE with A itself.) In this regard, Theorem 1.5 also shows that $\mathsf{BPE} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ is not affine relativizing (since $\mathsf{BPE}^{\widetilde{A}} \subseteq \mathsf{SIZE}^{A}[O(n)] \subseteq \mathsf{SIZE}^{\widetilde{A}}[O(n)]$).

To obtain Theorem 1.5, we prove a lower bound for XOR-Missing-String *against multiple pseudodeterministic* BPP *protocols simultaneously* in the following model.

- There are t communication protocols $Q_1, Q_2, \ldots, Q_t$ and t inputs (X_1, Y_1), (X_2, Y_2), $\ldots$, (X_t, Y_t).
- The goal of protocol Q_i is to solve the input (X_i, Y_i). However, each protocol is given the inputs to other protocols as well. More precisely, in each Q_i, Alice gets $(X_1, X_2, \ldots, X_t)$ as inputs and Bob gets $(Y_1, Y_2, \ldots, Y_t)$ as inputs.[8]
- We say that the protocols ***succeed*** if *at least one* of the protocols Q_i outputs a correct answer.

[7] For example, algebrization is not necessarily closed under *modus ponens*: Although it requires non-algebrizing techniques to prove, say, $\mathsf{NEXP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$, it might still be possible to exhibit a complexity class $\mathcal{C}$ and prove both $\mathcal{C} \subseteq \mathsf{NEXP}$ and $\mathcal{C} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ via algebrizing techniques. In contrast, affine relativization is clearly closed under *modus ponens*.

[8] We remark that in the case of XOR-Missing-String, each X_i and Y_i is already a list of strings, which means Alice and Bob get t lists of strings each.

- For proving algebrization barriers, each Q_i and (X_i, Y_i) should have a different size and correspond to different input lengths of the oracle, but we omit this detail in the informal description here. We refer the reader to Theorem 4.1 for details.

The point of this model is that the protocols are allowed to look at other protocols' inputs and to perform *win-win analyses*: the correctness of Q_i may rely on the *failure* of some other protocol $Q_{i'}$.[9] Indeed, this scenario models a variety of win-win analyses in complexity theory; see e.g., [CHR24, Section 1.4.2] and [LORS24, Section 5.2]. Circuit lower bounds for MA [BFT98, San09] can also be modeled as win-win algebraic query algorithms for Missing-String; see Section A.3.

We show that efficient pseudodeterministic protocols require large communication complexity to solve XOR-Missing-String, even if they receive multiple instances and are allowed to perform win-win analyses in the above sense.

Theorem 1.6 (Informal and simplified version of Theorem 4.1)**.** *In the above model, suppose that each (X_i, Y_i) is a* XOR-Missing-String(n_i, m_i) *instance but the communication complexity of each Q_i is "much less" than m_i. Then there exists a sequence of inputs $\{(X_i, Y_i)\}_{i\in[t]}$ such that every Q_i fails to solve its corresponding instance (X_i, Y_i) pseudodeterministically.*

1.3 Barriers to MA_{E} Circuit Lower Bounds

Finally, we present algebrization barriers to improving the half-exponential lower bound for MA_{E} [BFT98]. While we are unable to fully resolve the algebrizing circuit complexity of MA_{E}, we show that for a natural subclass of MA_{E} which includes the hard language in [BFT98], non-algebrizing techniques are required to go beyond half-exponential bounds.

Recall that a standard algorithm $M^{\widetilde{A}}$ in $\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}}$ defines a hard language if, *relative to this particular oracle* $\widetilde{A}$, $M^{\widetilde{A}}$ satisfies the MA promise and defines a language without small A-oracle circuits; we do not care about the behavior of $M^{\widetilde{B}}$ for other oracles $\widetilde{B}$. In contrast, we say that an MA_{E} machine M is a ***robust*** machine for defining a hard language (or simply "is robust"), if for *any* oracle B and its multilinear extension $\widetilde{B}$ such that $M^{\widetilde{B}}$ satisfies the MA promise, the language computed by $M^{\widetilde{B}}$ does not have small B-oracle circuits. The use of interactive proofs in [BFT98] naturally leads to hard languages defined by robust MA machines; see Appendix A for details.[10]

We use Rob-MA^A (resp. Rob-$\mathsf{MA}_{\mathsf{E}}^A$) to denote the class of languages computed by robust MA (resp. MA_{E}) algorithms with access to oracle A. Our main result is that proving a super-half-exponential bound for languages in E or robust MA_{E} would require non-algebrizing techniques:

Theorem 1.7 (Informal version of Theorem 5.4)**.** *There exists an oracle A_3 and its multilinear extension $\widetilde{A_3}$ such that both $\mathsf{E}^{\widetilde{A_3}}$ and* Rob-$\mathsf{MA}_{\mathsf{E}}^{\widetilde{A_3}}$ *admit half-exponential size A_3-oracle circuits.*

[9] If the possibility of such win-win analyses sounds surprising, it may help to compare it with the following classic puzzle. There are N prisoners, each assigned a (not necessarily distinct) number between 0 and $N-1$. Each prisoner can see others' numbers but not their own. Each prisoner must guess their own number simultaneously. If at least one prisoner guesses correctly, they are all set free; if none of them do, they are all executed.

There is a simple solution that guarantees the prisoners' freedom. The i-th prisoner ($0 \le i \le N-1$) assumes that the total sum of all numbers is congruent to i modulo N, and then infers their own number as

$$(i - \text{sum of other prisoners' numbers}) \bmod N.$$

Exactly one assumption will be correct, so that prisoner will guess their own number correctly.

[10] In fact, the definition of robust MA resembles closer to the class $\mathsf{MA}\cap\mathsf{coMA}$ compared to MA itself. This is natural since *single-valued* FMA*-constructions of hard truth tables* (see, e.g., [CHR24, Section 1.2.1]) give rise to circuit lower bounds for $\mathsf{MA}\cap\mathsf{coMA}$, and indeed the lower bound proved in [BFT98] is for hard languages in $\mathsf{MA}_{\mathsf{E}}\cap\mathsf{coMA}_{\mathsf{E}}$. For ease of notation we use "robust MA" (Rob-MA) instead of "robust $\mathsf{MA}\cap\mathsf{coMA}$" (Rob-$\mathsf{MA}\cap\mathsf{coMA}$).

Why study robust $\mathsf{MA_E}$? We present two reasons for studying the notion of robust $\mathsf{MA_E}$.

Firstly, Theorem 1.7 is tight in the sense that, in the two cases of the win-win argument in [BFT98], the hard language is either in E (i.e., deterministic exponential time), or in robust $\mathsf{MA_E}$. Therefore, a slight adaptation of [BFT98] proves that for every oracle A and its multilinear extension $\widetilde{A}$, $\mathsf{E}^{\widetilde{A}} \cup \text{Rob-}\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}}$ does not have sub-half-exponential size A-oracle circuits. This is shown in Appendix A.

Secondly, while the classes E and $\text{Rob}_{h(n)}\text{-}\mathsf{MA_E}$ may initially appear arbitrary, they in fact capture two fundamental types of failures for algorithms in $\mathsf{MA_E} \cap \mathsf{coMA_E}$:

- For $M \in \text{Rob}_{h(n)}\text{-}\mathsf{MA_E}$, if $M^{\widetilde{A}}$ fails to achieve high circuit complexity, it must be due to an input x on which $M^{\widetilde{A}}$ is semantically incorrect. That is, the corresponding truth table entry is undefined. We call this a failure due to **no positive**.
- In contrast, for $M \in \mathsf{E}$, the value $M^{\widetilde{A}}(x)$ is always well-defined for all x, regardless of the oracle A. If $M^{\widetilde{A}}$ has low circuit complexity, it must be because its truth table is easy for A-oracle circuits. We call this a failure due to a **false positive**.

In general, for an algorithm $M \in \mathsf{MA_E} \cap \mathsf{coMA_E}$, failure to achieve high circuit complexity can be attributed to both types, depending on the oracle. Thus, E and $\text{Rob}_{h(n)}\text{-}\mathsf{MA_E}$ represent two extreme cases within $\mathsf{MA_E} \cap \mathsf{coMA_E}$.

Our proof of Theorem 5.4 entails techniques that can force the two types of failures to occur. In this sense, the proof says something fundamental about $\mathsf{MA_E} \cap \mathsf{coMA_E}$. However, our current methods are limited to handling algorithms that exhibit only a single failure mode. For more general algorithms in $\mathsf{MA_E} \cap \mathsf{coMA_E}$ that can have mixed types of failures, our understanding remains incomplete.

Proof of Theorem 1.7 via a communication lower bound. To prove algebrization barriers to circuit lower bounds for robust $\mathsf{MA_E}$, we need to understand the *robust* MA communication complexity of XOR-Missing-String. Here, an MA protocol P is called *robust* if on every input (X, Y), Merlin (i.e., the prover) cannot convince the protocol to output a non-solution for (X, Y) except with very small probability. Underlying Theorem 1.7 is the following lower bound stating that efficient MA protocols for XOR-Missing-String that are robust can only be correct on a tiny fraction of inputs:

Lemma 1.8. *Let $n \geq 1, 1 \leq m < 2^{n/2}$ be parameters. Let P be a robust FMA communication protocol of complexity C attempting to solve* XOR-Missing-String(n, m). *Then the fraction of inputs that P solves is at most*

$$2^{-\Omega(m/C)+O(C+n)}.$$

Intuitively, since there is no efficient way to verify whether a string is a solution, any protocol that solves a large number of instances of XOR-Missing-String is essentially guessing the answer (similar to the naïve protocol that succeeds with probability $1 - 2^{-O(n)}$), which means that it must make mistakes. Since robust protocols are not allowed to output false positives, they can only be correct on a tiny fraction of inputs.

Although Lemma 1.8 is useful for analyzing the behaviors of robust MA protocols, it does not directly imply any *half-exponential* bounds. Indeed, we need to carefully diagonalize against both $\mathsf{E}^{\widetilde{A}}$ and $\text{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}}$ simultaneously, which leads to a half-exponential bound. We discuss this in more detail in Section 1.4.

1.4 Technical Overview

Next, we briefly overview our proof techniques. In fact, the engine behind all of our communication lower bounds is the following observation about XOR-Missing-String:

Lemma 1.9 (Informal version of Lemma 3.2)**.** *For every large enough rectangle $R = \mathcal{X} \times \mathcal{Y}$ and every answer s, there is a large enough subrectangle $R' = \mathcal{X}' \times \mathcal{Y}'$ ($\mathcal{X}' \subseteq \mathcal{X}$ and $\mathcal{Y}' \subseteq \mathcal{Y}$) such that s is a* wrong answer *for the* XOR-Missing-String *problem on every input $(X, Y) \in R'$.*

With this lemma in hand, it is not hard to prove Theorem 1.4. We characterize a PostBPP communication protocol as a distribution over labeled rectangles (using *approximate majority covers* [Kla03]), where for each input (X, Y), the output of the protocol is given by the label of a randomly selected rectangle that contains (X, Y). Using Lemma 1.9, for every large enough labeled rectangle, there exists a large subrectangle in which the label is never a solution to XOR-Missing-String. Lemma 1.9 thus translates to a lower bound for the success probability of large labeled rectangles (i.e., the probability that the label is a solution, over a random input in the rectangle). This then translates to a lower bound for (weighted) average success probability over a random input, because the distribution consists mostly of large rectangles when the protocol is efficient. This lower bound implies that efficient protocols cannot have very high success probability.

However, more work needs to be done to obtain our other communication lower bounds and algebrization barriers.

BPP communication lower bounds. We prove our pseudodeterministic BPP communication lower bound in the "win-win model" (Theorem 1.6) via a reduction to the PostBPP communication lower bound (Theorem 1.4). We first use an induction on the number of protocols; assume towards a contradiction that Theorem 1.6 holds for any sequence of $t-1$ protocols but does not hold for some sequence of t protocols $Q_1, \ldots, Q_t$. Then the following PostBPP communication protocol solves XOR-Missing-String efficiently: Given an input (X, Y), guess $t-1$ inputs $\{(X_i, Y_i)\}_{i \in [t-1]}$ uniformly at random, set $(X_t, Y_t) := (X, Y)$, and *postselect* on the event that all of the first $t-1$ protocols fail. That is, for every $1 \leq i \leq t-1$, Q_i fails to solve (X_i, Y_i) given $\{(X_i, Y_i)\}_{i \in [t]}$. By our induction hypothesis, this event happens with probability strictly greater than 0. If this happens, then Q_t solves (X_t, Y_t) correctly. This contradicts Theorem 1.4.

For clarity, we have made a crucial simplification in the above description: To obtain algebrization barriers, we also need to prove lower bounds against *list-solvers* for XOR-Missing-String, which are communication protocols that output a *small set* of answers such that one of the answers is correct. Fortunately, it is not hard to adapt the proof of Theorem 1.4 to prove a PostBPP communication lower bound for list solvers; see Section 3.2.

MA communication lower bounds against robust protocols. A robust protocol can be described as a collection of randomized verifiers $V_{w,\pi}$, where $V_{w,\pi}(X, Y) = 1$ means that when the input is (X, Y) and π is given as proof, the protocol accepts w as output. In the formal proof, we apply error reduction on the verifiers, so that when w is not a solution to (X, Y), $V_{w,\pi}(X, Y) = 1$ with extremely small probability.

Thus, if a protocol has large success probability, there must exist some answer string w and some proof π, such that over a random input (X, Y), $\Pr[V_{w,\pi}(X, Y) = 1]$ is large. We interpret this $V_{w,\pi}$ as a distribution over labeled rectangles, and apply Lemma 1.9 to show that conditioned on $V_{w,\pi}(X, Y) = 1$, the verifier makes many mistakes (i.e., $\Pr[V_{w,\pi}(X, Y) = 1 \wedge$

w is not a solution to (X, Y)] is large). This implies that the protocol must make mistakes on some (X, Y), so it cannot be robust.

The half-exponential barrier for robust $\mathsf{MA_E}$. We formulate the problem in terms of refuting communication protocols, where we need to refute a finite set of protocols (deterministic or robust) on each input length. More formally, to construct an algebrized world with circuit upper bound $h(n)$, our oracle encodes one instance (X_i, Y_i) of XOR-Missing-String$(2^i, 2^{h(i)})$ for each input length i. Each protocol being diagonalized is dedicated to solving the one instance that corresponds to its input length (it can nevertheless see the other instances).

We employ an inductive approach to construct the oracle. At step i, we maintain a rectangle[11] R_i of oracles, such that the rectangle is relatively large, and any protocol that works on input length at most i fails to solve its instance on any oracle $A \in R_i$.

The goal of every step is then to slightly shrink the rectangle R_{i-1} into R_i, so that the protocols working on length i are refuted. To achieve this, we employ a combination of two different strategies:

- **Refuting deterministic protocols:** Given a deterministic protocol P working on length i and a large enough rectangle R_i, we can always find a large subrectangle $R' \subseteq R_i$ such that P outputs the same answer on every oracle in R' (this is by definition of deterministic protocols). By Lemma 1.9, there exists a large subrectangle $R'' \subseteq R'$, such that for any oracle $A \in R''$, P does not solve (X_i, Y_i) on A. We then update $R_i \leftarrow R''$ to refute P.
- **Refuting robust protocols:** Given a robust protocol Q working on length i and a large enough rectangle R_i, a random oracle from R_i refutes Q with high probability. This is true as long as the density of R_i is sufficiently larger than the success probability of Q on a uniformly random input (as upper bounded in Lemma 1.8).

On their own, both strategies work extremely well: The first strategy is very similar to the proof of $\mathsf{NEXP}^{\widetilde{A}} \subset \mathsf{P}^A/\mathrm{poly}$ by [AW09], and can be used to obtain much better circuit upper bounds than half-exponential. For the second strategy, since robust protocols are very weak, a randomly selected oracle would refute all robust protocols with high probability.

However, it is unclear how to combine the two strategies. On the one hand, after refuting a deterministic protocol P by the first strategy, the resulting rectangle R_i may be too small for the second strategy to work. On the other hand, although the set of oracles $R' \subseteq R_i$ that refutes a robust protocol Q is large, it may not be a rectangle, so we cannot use it to refute the next deterministic protocol.

To integrate these two strategies, we refute each robust protocol at a carefully chosen time step. Suppose that we refute a robust protocol Q_i working on length i *after* all deterministic protocols working on lengths $< k$ are refuted, where $k = k(i)$ is some function on i. Also, recall that we want to prove an algebrized circuit upper bound of size $h(n)$.

- After refuting the deterministic protocols working on lengths $< k$, we end up with a rectangle that contains roughly a $2^{-2^{O(k)}}$ fraction of all oracles. This is because the deterministic protocols working on lengths $< k$ run in time $2^{O(k)}$. In contrast, Lemma 1.8 implies that Q_i only solves a $\approx 2^{-2^{h(i)}}$ fraction of oracles (since it attempts to solve an instance of XOR-Missing-String$(2^i, 2^{h(i)})$). Therefore, if $2^{-2^{O(k)}} \gg 2^{-2^{h(i)}}$, then we can refute Q_i.

[11] We interpret our oracles as the concatenation of Alice's and Bob's inputs, so it is reasonable to talk about a "rectangle" of oracles.

- On the other hand, after refuting Q_i, we still need to refute the deterministic protocols working on length k. Since these protocols attempt to solve XOR-Missing-String$(2^k, 2^{h(k)})$, to apply Lemma 1.9, this requires our current rectangle R_k to have size at least roughly $2^{-2^{h(k)}}$. Since Q_i is equivalent to a deterministic protocol running in time $2^{2^{O(i)}}$ (by enumerating the random bits and Merlin's proof), after refuting it, we still have a rectangle of density $\approx 2^{-2^{2^{O(i)}}}$. We thus need that $2^{-2^{2^{O(i)}}} \gg 2^{-2^{h(k)}}$.

Overall, we require that $k \ll h(i)$ and $2^i \ll h(k)$. This implies that h must be super-half-exponential.

1.5 Related Works

Prior work on super-polynomial circuit lower bounds. Kannan's seminal work [Kan82] proved a super-polynomial size lower bound for $\Sigma_2\mathsf{EXP}$; since then, a sequence of work has proved circuit lower bounds for various complexity classes such as $\mathsf{ZPEXP}^{\mathsf{NP}}$ [KW98, BCG+96], $\mathsf{S}_2\mathsf{EXP}$ [CCHO05, Cai07], PEXP [Vin05, Aar06], $\mathsf{MA}_{\mathsf{EXP}}$ [BFT98, San09], $\mathsf{ZPEXP}^{\mathrm{MCSP}}$ [IKV18, HLR23], and more [SM02, CLL25]. Such lower bounds are usually proved by *Karp–Lipton* theorems [KL80, BFNW93, CMMW19]: If a large uniform class (usually PH or EXP) admits polynomial-size circuits, then it collapses to a smaller uniform class (such as $\Sigma_2\mathsf{P}$). As explained in [MVW99] and [CHR24, Section 1.4.1], such a strategy naturally yields a *half-exponential* bound. Recently, [CHR24, Li24] proved near-maximum $(2^n/n)$ circuit lower bounds for the classes $\Sigma_2\mathsf{EXP}$, $\mathsf{ZPEXP}^{\mathsf{NP}}$, and $\mathsf{S}_2\mathsf{EXP}$, improving upon previous half-exponential bounds. Near-maximum circuit lower bounds are also known for several classes with *subexponential* amount of advice bits, such as $\mathsf{BPEXP}^{\mathrm{MCSP}}/_{2^{n^\varepsilon}}$ [HLR23] and $\mathsf{AMEXP}/_{2^{n^\varepsilon}}$ [CLL25].

Algebrization. The interactive proof results such as $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92] and $\mathsf{MIP} = \mathsf{NEXP}$ [BFL91] generated much excitement among complexity theorists, as they are the first "truly compelling" ([All90]) non-relativizing results in complexity theory. There has been much discussion on the extent to which these results are non-relativizing, and how to adapt the relativization barrier to accommodate these results [AIV92, For94]. Arguably, the most influential work in this direction is that of Aaronson and Wigderson [AW09] on the algebrization barrier, which nicely captures interactive proof results and arithmetization-based techniques. However, Aaronson and Wigderson's algebrization barrier has its own subtleties (such as not being closed under *modus ponens*, see Footnote 7), which leads to several proposed refinements of its definition [IKK09, AB18]. We also mention the *bounded relativization* barrier, recently proposed by Hirahara, Lu, and Ren [HLR23], which attempts to capture the interactive proof results entirely within the original Baker–Gill–Solovay [BGS75] framework of relativization.

1.6 Open Problems

We think the most important open problem left in our paper is to study the algebrizing circuit complexity of MA_{E}. Can we strengthen Theorem 1.7 to hold for all of MA_{E} instead of just "robust" MA_{E} algorithms? If not, can we prove an exponential size lower bound for MA_{E} (potentially with one bit of advice) using algebrizing techniques?

Another open question is to prove a PP communication lower bound for XOR-Missing-String, which would imply an algebrization barrier to proving almost-everywhere circuit lower bounds for PEXP. Such circuit lower bounds are known in the infinitely-often regime [BFT98, Vin05].

Finally, our work did not examine the regime of fixed-polynomial circuit lower bounds. Using algebrizing techniques, Santhanam [San09] proved that for every constant $k \geq 1$, $\mathsf{MA}/_1 \not\subseteq \mathsf{SIZE}[n^k]$. Aaronson and Wigderson [AW09] speculated that it might be possible to eliminate the advice bit and prove $\mathsf{MA} \not\subseteq \mathsf{SIZE}[n^k]$ using "tried-and-true arithmetization methods," but there has been no progress in this direction. Is there an algebrizing barrier to proving $\mathsf{MA} \not\subseteq \mathsf{SIZE}[n^k]$?

2 Preliminaries

Definition 2.1. A ***search problem*** f over domain $\mathcal{X} \times \mathcal{Y}$ and range $\mathcal{O}$ is defined using a relation $\mathcal{R} \subset (\mathcal{X} \times \mathcal{Y}) \times \mathcal{O}$. For any input $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, the valid solutions for f on (X, Y) are those s such that $(X, Y, s) \in \mathcal{R}$ (we say that ***s is a solution to $f(X, Y)$***). We say that f is a ***total problem***, if for any (X, Y), there is at least one valid solution.

2.1 Complexity Classes

We assume familiarity with basic complexity classes such as P, NP, BPP and their exponential-time versions EXP, NEXP, BPEXP. The reader is encouraged to consult standard textbooks [AB09, Gol08] or the Complexity Zoo[12] for their definition.

Definition 2.2 ($\mathsf{PostBPP}$)**.** A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ is in pr-$\mathsf{PostBPP}$ if there exist two polynomial-time algorithms A, B, taking an input $x \in \{0,1\}^n$ and randomness $r \in \{0,1\}^{\text{poly}(n)}$ (they take the *same* randomness), such that the following holds for every $x \in \Pi_{\text{yes}} \cup \Pi_{\text{no}}$.

- If $x \in \Pi_{\text{yes}}$, then $\Pr_r[A(x, r) = 1 \mid B(x, r) = 1] \geq 2/3$.
- If $x \in \Pi_{\text{no}}$, then $\Pr_r[A(x, r) = 1 \mid B(x, r) = 1] \leq 1/3$.
- $\Pr_r[B(x, r) = 1] > 0$.

We say that the algorithm ***postselects*** on the event that $B(x, r) = 1$.

Definition 2.3 ($\mathsf{MA} \cap \mathsf{coMA}$)**.** A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ is in pr-($\mathsf{MA} \cap \mathsf{coMA}$) if there exist two polynomial-time algorithms (verifiers) V_0, V_1, taking an input $x \in \{0,1\}^n$, a proof $\pi \in \{0,1\}^{\text{poly}(n)}$ and randomness $r \in \{0,1\}^{\text{poly}(n)}$ (they take *different* randomness), such that the following holds for every $x \in \Pi_{\text{yes}} \cup \Pi_{\text{no}}$.

- If $x \in \Pi_{\text{yes}}$, then V_1 accepts x and V_0 rejects x.
- If $x \in \Pi_{\text{no}}$, then V_0 accepts x and V_1 rejects x.

Here, for $k = 0, 1$, we say that

- V_k ***accepts*** x, if there exists a proof π such that $\Pr_r[V_k(x, \pi, r) = 1] \geq 2/3$.
- V_k ***rejects*** x, if for any proof π, we have that $\Pr_r[V_k(x, \pi, r) = 1] \leq 1/3$.

[12] https://complexityzoo.net/, accessed Nov 15, 2025.

2.2 Algebrization

In this section, we present some definitions regarding algebrization barriers. The definitions are in accordance with [AW09], with slight modifications.

Definition 2.4 (Oracle)**.** An ***oracle*** A is a collection of Boolean functions $A_m : \{0,1\}^m \to \{0,1\}$, one for each $m \in \mathbb{N}$. Given a complexity class $\mathcal{C}$, by $\mathcal{C}^A$ we mean the class of languages decidable by a $\mathcal{C}$ machine that can query A_m for any m of its choice.

We say that a separation $\mathcal{C} \not\subset \mathcal{D}$ does not algebrize, if there exist an oracle A and a low-degree extension $\widetilde{A}$ of A, such that $\mathcal{C}^{\widetilde{A}} \subset \mathcal{D}^A$. Throughout this paper, we only consider multilinear extensions, which are the simplest class of low-degree extensions.

Definition 2.5 (Multilinear Extension over Finite Fields)**.** Let $A_m : \{0,1\}^m \to \{0,1\}$ be a Boolean function, and let $\mathbb{F}$ be a finite field. The (unique) multilinear extension of A_m over $\mathbb{F}$ is the linear function $\widetilde{A}_{m,\mathbb{F}} : \mathbb{F}^m \to \mathbb{F}$ that agrees with A_m on $\{0,1\}^m$. Given an oracle $A = (A_m)$, the ***multilinear extension*** $\widetilde{A}$ of A is the collection of multilinear extensions $\widetilde{A}_{m,\mathbb{F}}$, one for each positive integer m and finite field $\mathbb{F}$.

Given a complexity class $\mathcal{C}$, by $\mathcal{C}^{\widetilde{A}}$ we mean the class of languages decidable by a $\mathcal{C}$ machine that can query $\widetilde{A}_{m,\mathbb{F}}$ for any $m, \mathbb{F}$ of its choice. Moreover, we assume that each query to $\widetilde{A}_{m,\mathbb{F}}$ takes $O(m \cdot \log|\mathbb{F}|)$ time.

An important property of multilinear extensions is that algorithms having access to $\widetilde{A}$ can be simulated by communication protocols where Alice and Bob are each given half of A as input. This reduces proving algebrization barriers to proving communication lower bounds. Formally, we have the following theorem, which is a simple corollary of [AW09, Theorem 4.11].

Theorem 2.6. *Let A be an oracle, and let A_0 (resp. A_1) be the subfunction of A obtained by restricting the first bit to 0 (resp. 1). Let M be a deterministic algorithm that has oracle access to $\widetilde{A}$ and runs in time $T(|x|)$ when given x as input. There exists a deterministic communication protocol P in which Alice (resp. Bob) is given x and the function A_0 (resp. A_1) as input, such that P uses $O(T(|x|)^3)$ bits of communication, and agrees with $M(x)$ on any input x and any oracle A.*

We remark that the proof of [AW09, Theorem 4.11] can be generalized to produce randomized communication protocols from randomized algorithms, MA communication protocols from MA algorithms, and so on.

3 An Infinitely-Often Algebrization Barrier for PostBPE

In this section, we prove the following algebrization barrier:

Theorem 1.3 (Restated)**.** *There exists an oracle A and its multilinear extension $\widetilde{A}$ such that*

$$\text{pr-}\mathsf{PostBPE}^{\widetilde{A}} \subseteq \text{i.o.-}\mathsf{SIZE}^A[O(n)].$$

3.1 PostBPP Communication Lower Bounds for XOR-Missing-String

Theorem 1.3 follows from the following communication lower bound for XOR-Missing-String:

Theorem 1.4 (Restated)**.** *Let $n \geq 1$ and $20n \leq m < 2^{n/2}$ be integers. Any PostBPP communication protocol that solves XOR-Missing-String(n, m) with error $\leq 2^{-5n}$ must have communication complexity $\Omega(m)$.*

In this paper, PostBPP communication protocols are defined as follows:[13]

Definition 3.1 (PostBPP communication protocols for search problems)**.** Let f be a search problem over domain $\mathcal{X} \times \mathcal{Y}$ and range $\mathcal{O}$. A **PostBPP** ***communication protocol*** Π for f is defined as follows: Let k be the length of the public randomness used by Π, and let $\{\Pi_r\}_{r\in\{0,1\}^k}$ be a set of deterministic communication protocols, each having communication complexity c. On input $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, protocol Π first samples a uniformly random string $r \in \{0,1\}^k$, then simulates Π_r, whose output can be $\perp$ or any value from $\mathcal{O}$. The communication complexity of Π is defined to be $k + c$.

We say that Π ***solves*** f ***with error*** ε, if for any input $(X, Y) \in \mathcal{X} \times \mathcal{Y}$,

$$\Pr_r\left[\Pi_r(X,Y) \text{ is a solution to } f(X,Y) \,\middle|\, \Pi_r(X,Y) \neq \perp\right] \geq 1 - \varepsilon.$$

We say that Π is ***pseudodeterministic***, if for any input $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, there exists some answer $s \in \mathcal{O}$, such that

$$\Pr\left[\Pi_r(X,Y) = s \,\middle|\, \Pi_r(X,Y) \neq \perp\right] \geq 2/3.$$

The main technical observation for our lower bound is that for any large enough rectangle R, no single answer $s \in \{0,1\}^n$ will be correct for every input $(X, Y) \in R$. In fact, there is always a $2^{-\Theta(n)}$ fraction of inputs in R on which s is not a valid solution.

To use this result in the lower bound, note that a PostBPP communication protocol can be seen as a distribution of labeled rectangles (i.e., approximate majority covers, as defined in Section 3.1.2). If a protocol has small communication complexity, then it must contain many large rectangles, so we can apply the aforementioned lower bound on individual rectangles, which implies that the protocol must have error $2^{-O(n)}$.

3.1.1 A Lower Bound for One Rectangle

In this section, we prove the main technical lemma for the communication lower bound. It is helpful to draw an analogy from the decision tree version of Missing-String: Suppose that the input is a list of m n-bit strings. Whenever the decision tree only queries the input on $m - 1$ bits, there always exists a string in the input list that is not queried. In this case, regardless of the decision tree's answer, the adversary can arbitrarily modify the value of that string and make the decision tree fail.

Here, we show that the XOR-Missing-String problem has a similar property: If, conditioned on the communication history, the rectangle formed by the possible inputs is large (this corresponds to the decision tree making a small number of queries), then regardless of the protocol's answer, there always exists a significant portion of the rectangle, on which the protocol fails.

Lemma 3.2. *Let $n \geq 1, 1 \leq m < 2^{n/2}$ be integers. Let $R \subseteq \{0,1\}^{nm} \times \{0,1\}^{nm}$ be a rectangle (i.e., R is of the form $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^{nm}$) of size at least $2^{2nm-m+2}$. Let s be any n-bit string. Then there exists some R', which is a subrectangle of R and has size at least $2^{-2n-2}|R|$, such that for any $(X, Y) \in R'$, s is not a solution to* XOR-Missing-String *on (X, Y).*

[13] There are many different definitions in the literature (see e.g., [GPW18]). Here, we choose a simple definition that is best suited for proving the barrier.

Proof. Without loss of generality, we only have to prove the lemma for the case where $s = 0^n$. When $s \neq 0^n$, we consider a different rectangle R_s, in which every input string x for Alice in every input list $X \in \mathcal{X}$ is replaced by $x \oplus s$. By the definition of XOR-Missing-String, the lemma holds for R and s if and only if it holds for R_s and 0^n.

To prove the lemma, we need to show that there exists a large enough subrectangle R' in R, in which 0^n is never a solution to XOR-Missing-String. Recall that R is equal to $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a set of Alice's inputs and $\mathcal{Y}$ is a set of Bob's inputs. For any string $s \in \{0,1\}^n$, let $\mathcal{X}_s = \{X \in \mathcal{X}, s \in X\}$ denote the subset of $\mathcal{X}$ that contains the string s; $\mathcal{Y}_s$ is defined similarly.

For any $s \in \{0,1\}^n$, consider the subrectangle $\mathcal{X}_s \times \mathcal{Y}_s \subseteq R$. Since any input list (Alice's or Bob's) in the subrectangle always contains s, 0^n is never a solution in $\mathcal{X}_s \times \mathcal{Y}_s$.

In the remaining, we show that there exists some s such that $\mathcal{X}_s \times \mathcal{Y}_s$ is large enough (i.e., has size $\geq 2^{-2n-2}|R|$) using a counting argument. If we can show this, then the lemma follows by letting $R' = \mathcal{X}_s \times \mathcal{Y}_s$.

Let $cx_s = |\mathcal{X}_s|$ denote the number of occurrences of string s in $\mathcal{X}$. Note that, if some input $X \in \mathcal{X}$ contains multiple copies of s, X is only counted once in cx_s. Let $x_1, \ldots, x_{2^n}$ be all the n-bit strings, sorted in non-increasing order of cx. Similarly define cy_s and $y_1, \ldots, y_{2^n}$. We have the following lower bound on the number of occurrences of x_i:

Claim 3.3. *For any $1 \leq i \leq 2^n$, cx_{x_i} is at least*

$$\frac{|\mathcal{X}| - (i-1)^m}{2^n - i + 1}.$$

Proof. Let $k = \sum_{i' \geq i} cx_{x_{i'}}$, i.e., the sum of occurrences of $x_{\geq i}$.

Consider the inputs X in $\mathcal{X}$ that only contain strings from $x_1, \ldots, x_{i-1}$. The number of such X is at most $(i-1)^m$. Therefore, at least $|\mathcal{X}| - (i-1)^m$ inputs in $\mathcal{X}$ contain at least one string from $x_{\geq i}$. Each of the $|\mathcal{X}| - (i-1)^m$ inputs contribute at least one to the sum k. That is,

$$k = \sum_{i' \geq i} cx_{x_{i'}} \geq |\mathcal{X}| - (i-1)^m.$$

Since x_i occurs at least as frequently as any string in $x_{i+1}, \ldots, x_{2^n}$, we have that $k \leq cx_{x_i} \cdot (2^n - i + 1)$. Therefore,

$$cx_{x_i} \geq \frac{k}{2^n - i + 1} \geq \frac{|\mathcal{X}| - (i-1)^m}{2^n - i + 1}. \qquad \diamond$$

The same bound also applies to cy_{y_i}. Fix any $1 \leq i \leq 2^n$ and consider the strings $x_1, \ldots, x_i$ and $y_1, \ldots, y_{2^n - i + 1}$. By the pigeonhole principle, there must exist some string s that appears in both $\{x_1, \ldots, x_i\}$ and $\{y_1, \ldots, y_{2^n - i + 1}\}$. Using Claim 3.3, we can show that $\mathcal{X}_s \times \mathcal{Y}_s$ is large, that is,

$$\begin{aligned}
|\mathcal{X}_s \times \mathcal{Y}_s| &= cx_s \cdot cy_s \\
&\geq cx_{x_i} \cdot cy_{y_{2^n - i + 1}} && \text{(by definition of the lists } x, y) \\
&\geq \frac{|\mathcal{X}| - (i-1)^m}{2^n - i + 1} \cdot \frac{|\mathcal{Y}| - (2^n - i)^m}{i}. && \text{(by Claim 3.3)}
\end{aligned}$$

Let $A = |\mathcal{X}|, B = |\mathcal{Y}|$. It now remains to show that, for any A, B such that $1 \leq A, B \leq 2^{nm}$ and $A \cdot B \geq 2^{2nm - m + 2}$, there exists some $1 \leq i \leq 2^n$ such that the rectangle chosen above is large enough. That is,

$$\frac{A - (i-1)^m}{2^n - i + 1} \cdot \frac{B - (2^n - i)^m}{i} \geq 2^{-2n-2} \cdot A \cdot B. \tag{1}$$

Let $i^* \geq 1$ be the largest integer such that $2(i^*-1)^m \leq A$. Note that $2 \cdot 2^{nm} > A$, so we must have $i^* \leq 2^n$. In this case, we have that $2(2^n - i^*)^m \leq B$, since if not, then

$$\begin{aligned} & 2(i^*)^m \cdot 2(2^n - i^*)^m > A \cdot B && ((i^*)^m > A \text{ by definition of } i^*) \\ \Rightarrow\ & (i^* \cdot (2^n - i^*))^m > A \cdot B/4 \geq 2^{2nm-m} \\ \Rightarrow\ & i^* \cdot (2^n - i^*) > 2^{2n-1}, \end{aligned}$$

which is impossible since $i \cdot (2^n - i) \leq 2^{2(n-1)}$ for any i. Therefore, by plugging i^* into (1), we have that

$$\begin{aligned} & \frac{A - (i^*-1)^m}{2^n - i^* + 1} \cdot \frac{B - (2^n - i^*)^m}{i^*} \\ \geq\ & \frac{A/2}{2^n - i + 1} \cdot \frac{B/2}{i^*} && (2(i^*-1)^m \leq A \text{ and } 2(2^n - i^*)^m \leq B) \\ \geq\ & \frac{A/2}{2^n} \cdot \frac{B/2}{2^n} = 2^{-2n-2} \cdot A \cdot B. \end{aligned}$$

□

Lemma 3.2 can be extended to the case where, in addition to (X, Y), Alice and Bob also receive some auxiliary input (which is independent of (X, Y)). This variant will be useful in Section 5.

Corollary 3.4. *Let $n \geq 1, 1 \leq m < 2^{n/2}, a \geq 0$ be integers. Let $R \subseteq \{0,1\}^{nm+a} \times \{0,1\}^{nm+a}$ be a rectangle of size at least $2^{2nm+2a-m+2}$ (i.e., R is of the form $X \times Y$ for some $X, Y \subset \{0,1\}^{nm+a}$). Each element in R is interpreted as having the form $(X \circ t_x, Y \circ t_y)$, where (X, Y) is an input to* XOR-Missing-String(n, m)*, and t_x, t_y are two a-bit strings.*

Let s be any n-bit string. There exists some R', which is a subrectangle of R and has size at least $2^{-2n-2}|R|$, such that for any $(X \circ t_x, Y \circ t_y) \in R'$, s is not a solution to XOR-Missing-String *on (X, Y).*

Proof. This corollary cannot be obtained by applying Lemma 3.2 in a black-box way. However, it is provable using the same techniques, i.e., by defining $\mathcal{X}_s = \{s \in X : (X \circ t_x) \in \mathcal{X}\}$ and $\mathcal{Y}_s = \{s \in Y : (Y \circ t_y) \in \mathcal{Y}\}$, and showing that $\mathcal{X}_s \times \mathcal{Y}_s$ is large for some s. The details are omitted. □

3.1.2 Reducing PostBPP Communication to Approximate Majority Covers

To use Lemma 3.2 in the lower bound proof, we use the characterization of PostBPP communication protocols by distributions of labeled rectangles called *approximate majority covers* [Kla03].

Definition 3.5 (approximate majority covers)**.** Let f be a search problem over the domain $\mathcal{X} \times \mathcal{Y}$ and range $\mathcal{O}$. An ***approximate majority cover for f*** is a (multi) set of labeled rectangles $\mathcal{AMC} = \{(R, s)\}$, where $R \subseteq \mathcal{X} \times \mathcal{Y}$ is a rectangle, i.e., R is of the form $A \times B$ for some $A \subseteq \mathcal{X}, B \subseteq \mathcal{Y}$, and $s \in \mathcal{O}$ is the label of R. The ***size*** of $\mathcal{AMC}$ is defined as the number of labeled rectangles.

We say that the approximate majority cover ***solves f with error ε***, if for any $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, we have that

$$\Pr_{(R,s)\sim\mathcal{AMC}}[s \text{ is a solution to } f(X, Y) \mid (X, Y) \in R] \geq 1 - \varepsilon.$$

That is, when we randomly sample from $\mathcal{AMC}$ a labeled rectangle (R, s) that contains (X, Y), the label s is a solution with probability $2/3$.

Claim 3.6. *Let f be a total search problem over domain $\mathcal{X} \times \mathcal{Y}$ and range $\mathcal{O}$. If there exists a* PostBPP *communication protocol Π that solves f with error ε and has complexity C, then there exists an approximate majority cover that solves f with error ε and has size $\leq 2^C$.*

Proof. Assume that Π uses $k \leq C$ bits of randomness. That is, Π is the uniform distribution over 2^k deterministic communication protocols $\{\Pi_r\}$, where each protocol Π_r has complexity $C - k$.

Construct an approximate majority cover $\mathcal{AMC}$ as follows: Initially, $\mathcal{AMC}$ is empty. Each deterministic protocol Π_r partitions the input space into at most 2^{C-k} pairwise disjoint rectangles, where the answers of Π_r in each rectangle are the same. For each such rectangle R of Π_r, suppose the answer of Π_r on R is v, we add (R, v) to $\mathcal{AMC}$ if and only if $v \neq \perp$.

It is clear that the size of $\mathcal{AMC}$ is at most 2^C. Moreover, for any $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, there is a one-to-one correspondence between the rectangles in $\mathcal{AMC}$ containing (X, Y) and the protocols Π_r for which $\Pi_r(X, Y) \neq \perp$. Therefore, the value of $\Pi(X, Y)$ is distributed the same as the label of a uniformly random rectangle in $\mathcal{AMC}$ that contains (X, Y). □

3.1.3 Putting Everything Together

With the components in place, we can now prove Theorem 1.4.

Theorem 1.4 (Restated)**.** *Let $n \geq 1$ and $20n \leq m < 2^{n/2}$ be integers. Any* PostBPP *communication protocol that solves* XOR-Missing-String(n, m) *with error $\leq 2^{-5n}$ must have communication complexity $\Omega(m)$.*

Proof. Let Π be a PostBPP communication protocol that solves XOR-Missing-String(n, m) with error $\leq 2^{-5n}$, and let C denote the complexity of Π. We convert Π into an approximate majority cover $\mathcal{AMC}$ using Claim 3.6. We have that $\mathcal{AMC}$ solves XOR-Missing-String(n, m) with error $\leq 2^{-5n}$ and has size $2^{O(C)}$. We now show that the size of $\mathcal{AMC}$ is at least $2^{\Omega(m)}$, which proves the lemma.

Let $S = \sum_{(R,s)\in\mathcal{AMC}} |R|$ denote the total size of the rectangles. Since $\mathcal{AMC}$ is correct, each input $(X, Y) \in \{0,1\}^{nm} \times \{0,1\}^{nm}$ must be contained in at least one rectangle, which means that $S \geq 2^{2nm}$.

Let

$$C_{\text{correct}} \coloneqq \sum_{(X,Y)\in\{0,1\}^{nm}\times\{0,1\}^{nm}} \sum_{(R,s)\in\mathcal{AMC}, R\ni(X,Y)} [s \text{ is a solution to XOR-Missing-String on } (X, Y)].$$

Since $\mathcal{AMC}$ solves XOR-Missing-String(n, m) with error 2^{-5n}, we have that

$$\begin{aligned}
&C_{\text{correct}} \\
&\geq \sum_{(X,Y)\in\{0,1\}^{nm}\times\{0,1\}^{nm}} (1 - 2^{-5n}) \cdot \sum_{(R,s)\in\mathcal{AMC}, R\ni(X,Y)} 1 \qquad \text{(each } (X, Y) \text{ is correct w.h.p.)} \\
&= (1 - 2^{-5n}) \cdot \sum_{(R,s)\in\mathcal{AMC}} \sum_{(X,Y)\in R} 1 \\
&= (1 - 2^{-5n}) \cdot S.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
&C_{\text{correct}} \\
&= \sum_{(R,s)\in\mathcal{AMC}} \sum_{(X,Y)\in R} [s \text{ is a solution to XOR-Missing-String on } (X,Y)] \\
&= \sum_{\substack{(R,s)\in\mathcal{AMC} \\ |R|\geq 2^{2nm-m+2}}} \sum_{(X,Y)\in R} [s \text{ is a solution to XOR-Missing-String on } (X,Y)] \\
&\quad + \sum_{\substack{(R,s)\in\mathcal{AMC} \\ |R|< 2^{2nm-m+2}}} \sum_{(X,Y)\in R} [s \text{ is a solution to XOR-Missing-String on } (X,Y)].
\end{aligned}$$

It follows from Lemma 3.2 that the first summand is at most

$$\sum_{\substack{(R,s)\in\mathcal{AMC} \\ |R|\geq 2^{2nm-m+2}}} (1-2^{-2n-2})\cdot|R| \leq (1-2^{-4n})\cdot S.$$

Let $|\mathcal{AMC}|$ denote the size of $\mathcal{AMC}$, then the second summand is at most

$$\begin{aligned}
\sum_{\substack{(R,s)\in\mathcal{AMC} \\ |R|< 2^{2nm-m+2}}} |R| &\leq |\mathcal{AMC}|\cdot 2^{2nm-m+2}. \\
&\leq |\mathcal{AMC}|\cdot S/2^{m-2}. && (\text{since } S\geq 2^{2nm})
\end{aligned}$$

If $|\mathcal{AMC}| \leq 2^{m/2}$, then the second summand is at most $2^{-m/2+2}\cdot S$, which is at most $2^{-8n}\cdot S$ since $m \geq 20n$. Summing everything together, we have that

$$C_{\text{correct}} \leq (1-2^{-4n}+2^{-8n})\cdot S,$$

which contradicts the previous bound of $C_{\text{correct}} \geq (1-2^{-5n})\cdot S$. Hence it must be the case that $|\mathcal{AMC}| > 2^{m/2}$. □

Lower bounds for pseudodeterministic protocols. A consequence of Theorem 1.4 is that it also applies to *pseudodeterministic* PostBPP communication protocols for XOR-Missing-String with constant error (say 1/3), as the error probability for such protocols can always be reduced to as small as possible by running it multiple times and outputting the majority answer. In contrast, the error probability of an arbitrary (non-pseudodeterministic) protocol cannot be amplified in general, since it is unclear how to verify whether a string is a valid solution to XOR-Missing-String.

Claim 3.7. *Let $n \geq 1, 1 \leq m < 2^{n/2}$ be integers. Let Π be a pseudodeterministic* PostBPP *communication protocol that solves* XOR-Missing-String(n,m) *with error* 1/3 *and has communication complexity C. Then for any $k \geq 1$, there exists a pseudodeterministic* PostBPP *communication protocol Π_k that solves* XOR-Missing-String(n,m) *with error $2^{-\Theta(k)}$ and has communication complexity $O(C\cdot k)$.*

By setting $k = \Theta(n)$ in the above lemma and applying Theorem 1.4, we have:

Corollary 3.8. *Let $n \geq 1$ and $20n \leq m < 2^{n/2}$ be integers. Any pseudodeterministic* PostBPP *communication protocol that solves* XOR-Missing-String(n,m) *with error probability $\leq 1/3$ requires communication complexity $\Omega(m/n)$.*

3.2 An Extension: XOR-Missing-String Lower Bound for List-Solvers

We prove a strengthened lower bound for XOR-Missing-String against *list-solvers*, which are protocols that output a list of strings, and are considered correct if *at least one* string in the list is a solution. Although this strengthened lower bound is not needed for Theorem 1.3, it is relevant in the context of *almost-everywhere* circuit upper bounds and will be useful in the proof of Theorem 1.5.

Definition 3.9. Let f be a search problem over domain $\mathcal{X} \times \mathcal{Y}$ and range $\mathcal{O}$. We say that a communication protocol Π is a ***list-solver*** for f, if it outputs a list of answers (i.e., its output is in $\mathcal{O}^k$ for some k).

For a PostBPP list-solver Π, we say that it ***list-solves f with error ε***, if for any input $(X, Y) \in \mathcal{X} \times \mathcal{Y}$,

$$\Pr[\text{one of the answers in } \Pi(X,Y) \text{ solves } (X,Y) \mid \Pi(X,Y) \neq \perp] \geq 1 - \varepsilon.$$

Lemma 3.10. *Let $n \geq 1, 1 \leq m < 2^{n/2}, k$ be parameters, where $m \geq 20nk$. Let Q be a* PostBPP *list-solver of* XOR-Missing-String(n, m) *that outputs k candidate strings. If Q list-solves* XOR-Missing-String(n, m) *with error $\leq 2^{-5nk}$, then the complexity of Q is at least $\Omega(m)$.*

It is natural to require that the error probability is at least $2^{-O(nk)}$, since this bound is achieved by a trivial protocol that outputs k random strings.

The proof is similar to Theorem 1.4: We use the lower bound on rectangles (Lemma 3.2) to show that, for large enough rectangles, even when the protocol outputs k candidate strings, there still exists a significant fraction of inputs in the rectangle on which all strings fail. We then use approximate majority covers to translate this result into a communication lower bound.

Proof. We start by proving a stronger version of Lemma 3.2.

Claim 3.11. *Let $n \geq 1, 1 \leq m < 2^{n/2}, k$ be parameters. Let $R \subseteq \{0,1\}^{nm} \times \{0,1\}^{nm}$ be a rectangle of size at least $2^{2nm-m+2+(2n+2)\cdot k}$. Let $s_1, \ldots, s_k$ be any set of k n-bit strings. Then there exists some R', which is a subrectangle of R and has size at least $2^{-(2n+2)\cdot k}|R|$, such that for any $(X, Y) \in R'$, none of $s_1, \ldots, s_k$ is a solution to* XOR-Missing-String *on (X, Y).*

Proof. The proof is by repeatedly applying Lemma 3.2. Given R and $s_1, \ldots, s_k$, we first apply Lemma 3.2 on R and s_1 to construct a subrectangle $R_1 \subset R$, which has size at least $2^{-2n-2}|R|$, on which s_1 is never a solution. Next, since $|R_1| \geq 2^{-2n-2}|R| \geq 2^{2nm-m+2}$, we can apply Lemma 3.2 on R_1 and s_2, and obtain another subrectangle $R_2 \subset R_1$, on which s_2 is never the solution. This process can be repeated, and the end result is a subrectangle R_k that refutes every s_i. ◇

Next, we convert Q into an approximate majority cover.

Formally, define the k-fold XOR-Missing-String(n, m) to be the search problem where, given an instance (X, Y) of XOR-Missing-String(n, m), the task is to output k strings so that at least one of them solves (X, Y). Since Q is a protocol for this problem, we can use Claim 3.6 to obtain an approximate majority cover $\mathcal{AMC}$, such that $\mathcal{AMC}$ has size at most 2^C (where C is the complexity of Q), and the label corresponding to each rectangle is a list of k strings. $\mathcal{AMC}$ satisfies that, for any $(X, Y) \in \{0,1\}^{nm} \times \{0,1\}^{nm}$,

$$\Pr_{(R, s_1, \ldots, s_k) \sim \mathcal{AMC}}[\text{one of } s_1, \ldots, s_k \text{ solves } (X,Y) \mid (X,Y) \in R] \geq 1 - 2^{-5nk}. \tag{2}$$

Finally, we show that $\mathcal{AMC}$ has size at least $2^{\Omega(m)}$. Let

$$C_{\text{correct}} = \sum_{(X,Y)} \sum_{(R,s_1,\ldots,s_k)\in\mathcal{AMC},(X,Y)\in R} [\text{one of } s_1,\ldots,s_k \text{ solves } (X,Y)].$$

Let $S = \sum_{(R,s_1,\ldots,s_k)\in\mathcal{AMS}} |R|$ denote the total size. The following computations are similar to the proof of Theorem 1.4, where similar steps are simplified.

By (2), we have that

$$C_{\text{correct}} \geq (1 - 2^{-5nk}) \cdot S.$$

On the other hand, using Claim 3.11, we have that

$$\begin{aligned} C_{\text{correct}} &\leq \sum_{R\in\mathcal{AMC}:|R|\geq 2^{2nm-m+2+(2n+2)\cdot k}} (1 - 2^{-(2n+2)\cdot k}) \cdot |R| + \sum_{R\in\mathcal{AMC}:|R|\leq 2^{2nm-m+2+(2n+2)\cdot k}} |R| \\ &\leq (1 - 2^{-(2n+2)\cdot k}) \cdot S + 2^C \cdot 2^{2nm-m+2+(2n+2)\cdot k}. \end{aligned}$$

Since $\mathcal{AMC}$ solves every input, we must have that $S \geq 2^{2nm}$. Therefore, the second term is at most $2^C \cdot 2^{-m+2+(2n+2)\cdot k} \cdot S$, which is $2^{-\Omega(m)} \cdot S$ since $m \geq 20nk$. We thus have that

$$(1 - 2^{(2n+2)\cdot k}) \cdot S + 2^C \cdot 2^{-\Omega(m)} \cdot S \geq (1 - 2^{-5nk}) \cdot S.$$

For this to hold, it must be the case that $C = \Omega(m)$. □

3.3 Proving the Barrier

Theorem 1.3 (Restated)**.** *There exists an oracle A and its multilinear extension $\widetilde{A}$ such that*

$$\text{pr-}\mathsf{PostBPE}^{\widetilde{A}} \subseteq \text{i. o.-}\mathsf{SIZE}^{A}[O(n)].$$

Proof of Theorem 1.3. Let $M_1, M_2, \ldots$ be a syntactic enumeration of PostBPE algorithms each running in time 2^n (that is, each M_i may or may not satisfy the PostBPP promise). If we can prove a barrier for all such algorithms, then we can also prove a barrier for all algorithms running in time $2^{O(n)}$ by a padding argument. We assume that each machine M_i appears infinitely many times in the list $(M_i)_{i\in\mathbb{N}}$.

Let n_1 be a large enough constant and set $n_i = 4^{n_{i-1}}$ for every $i > 1$. Let $m_i = n_i^{20}$ for every $i \geq 1$. Recall that logarithms are always base 2. The oracle A that we construct will have the following structure: On input strings whose lengths are not of the form $2\log m_i$, the value of A will always be zero. For every $i \geq 1$, the truth table of A on input length $2\log m_i$ will encode an instance (X_i, Y_i) of XOR-Missing-String(n_i, m_i). More specifically, the truth table of $A \cap \{0,1\}^{2\log m_i}$ restricted to inputs whose first bit is 0 (resp. 1) will be equal to X_i (resp. Y_i) padded with zeros. Note that the length of (X_i, Y_i) is $2m_i \cdot n_i \leq 2^{2\log m_i}$.

For every $i \geq 1$, the instance (X_i, Y_i) is designed to diagonalize against $M_i^{\widetilde{A}}$. More precisely, we want the following property to hold: let $\mathcal{GOOD}_i$ be the set of inputs $x \in \{0,1\}^{\log n_i}$ such that the execution of $M_i^{\widetilde{A}}(x)$ satisfies the semantics of PostBPP, then there exists a *non-solution* $\alpha \in \{0,1\}^{n_i}$ of the XOR-Missing-String instance (X_i, Y_i) such that $\alpha_x = M_i^{\widetilde{A}}(x)$ for every $x \in \mathcal{GOOD}_i$. Note that $\alpha = (X_i)_a \oplus (Y_i)_b$ for some $a, b \in [m_i]$, hence by hardwiring a and b we can construct an A-oracle circuit of size $O(\log m_i) \leq O(\log n_i)$ whose truth table is equal to α, and it follows that the

pr-PostBPE problem defined by $M_i^{\widetilde{A}}$ on input length $\log n_i$ can be computed by linear-size A-oracle circuits. Therefore, it suffices to satisfy the aforementioned property.

Note that, when considering the truth table of $M_i^{\widetilde{A}}$ on input length $\log n_i$, the algorithm $M_i^{\widetilde{A}}$ can only access $\widetilde{A}$ on inputs of length $\leq n_i$. Therefore, we only need to show that $M_i^{\widetilde{A}_{\leq n_i}}$ fails to solve (X_i, Y_i). Since $2\log m_{i+1} > n_i$, this implies that the algorithm only sees $(X_{\leq i}, Y_{\leq i})$.

We construct our oracle inductively. Suppose that we have constructed (X_j, Y_j) for every $j < i$, and we need to construct (X_i, Y_i). Let P_i be the following PostBPP communication protocol that tries to solve XOR-Missing-String(n_i, m_i). Given an instance (X, Y), define an oracle A such that $(X_{<i}, Y_{<i})$ are the previously fixed values and $(X_i, Y_i) = (X, Y)$. Then, Alice and Bob output the "truth table" of $M_i^{\widetilde{A}_{\leq n_i}}$ on inputs of length $\log n_i$, denoted as $\mathbf{tt} \in \{0,1\}^{n_i}$: for every $x \in \{0,1\}^{\log n_i}$, Alice and Bob simulate the computation of $M_i^{\widetilde{A}_{\leq n_i}}(x)$ using (the PostBPP version of) Theorem 2.6 repeatedly for $(1000n_i + 1)$ times, and output the majority outcome as the x-th bit of $\mathbf{tt}$. (We use boldface to emphasize that $\mathbf{tt}$ is a random variable). Let $\mathcal{GOOD}_i(X, Y)$ denote the set of inputs $x \in \{0,1\}^{\log n_i}$ such that the computation of $M_i^{\widetilde{A}_{\leq n_i}}(x)$ satisfies PostBPP promise, and let $f_i : \mathcal{GOOD}_i(X, Y) \to \{0,1\}$ denote the promise problem defined by $M_i^{\widetilde{A}_{\leq n_i}}$:

$$\mathcal{GOOD}_i^b(X,Y) = \left\{ x \in \{0,1\}^{\log n_i} : \Pr\left[M_i^{\widetilde{A}_{\leq n_i}}(x) = b \;\middle|\; M_i^{\widetilde{A}_{\leq n_i}}(x) \neq \perp \right] \geq 2/3 \right\},$$

$$\mathcal{GOOD}_i(X,Y) = \mathcal{GOOD}_i^0(X,Y) \cup \mathcal{GOOD}_i^1(X,Y),$$

$$f_i(x) = \begin{cases} 0 & \text{if } x \in \mathcal{GOOD}_i^0(X,Y); \\ 1 & \text{if } x \in \mathcal{GOOD}_i^1(X,Y). \end{cases}$$

We say that $\mathbf{tt}$ is *consistent* with f_i if for every $x \in \mathcal{GOOD}_i(X, Y)$, $\mathbf{tt}_x = f_i(x)$. Since the computation of $M_i^{\widetilde{A}_{\leq n_i}}(x)$ is repeated $(1000n_i + 1)$ times, the probability that $\mathbf{tt}$ is consistent with f_i is at least $1 - 2^{-10n_i}$. On the other hand, note that the behavior of P_i is the same as some PostBPP algorithm that has oracle access to $\widetilde{A}$ and runs in time $O(n_i^3)$ (since it considers n_i possible inputs, simulates M_i for $O(n_i)$ times on each of them, and each simulation takes $O(n_i)$ time), hence Theorem 2.6 implies that P_i can be implemented by a PostBPP communication protocol of complexity $O(n_i^9) = o(m_i)$. It follows from Theorem 1.4 that P_i cannot solve XOR-Missing-String(n_i, m_i) with success probability more than $1 - 2^{-5n_i}$; in other words, there exists some (X, Y) such that $\mathbf{tt}$ is a non-solution of (X, Y) w.p. at least 2^{-5n_i}. By a union bound, there is a non-zero probability that both of the following hold simultaneously: $\mathbf{tt}$ is consistent with f_i and $\mathbf{tt}$ is a non-solution of (X, Y). This means that there is a non-solution $\alpha \in \{0,1\}^{n_i}$ of (X, Y) such that for every $x \in \mathcal{GOOD}_i(X, Y)$, $M_i^{\widetilde{A}_{\leq n_i}}(x) = \alpha_x$. Setting $(X_i, Y_i) = (X, Y)$, this establishes the property we want for i, and we can continue our construction for $i + 1$. □

4 Alternative Almost-Everywhere Algebrization Barrier for BPE

In [AW09], the authors proved almost-everywhere algebrization barriers for many common complexity classes such as NEXP and BPEXP. However, their barriers were constructed using multiquadratic extensions, which are more difficult to manipulate than multilinear extensions, and are arguably less clean. Barriers based on multiquadratic extensions also fail to imply *affine relativization* barriers in the sense of [AB18]. It was unknown whether these barriers can be proven using only communication complexity. In this section, we use the techniques developed for

XOR-Missing-String to show that these almost-everywhere algebrization barriers can indeed be proven using communication complexity only.

Theorem 1.5 (Restated)**.** *There exists an oracle A and its multilinear extension $\widetilde{A}$ such that*

$$\mathsf{BPE}^{\widetilde{A}} \subseteq \mathsf{SIZE}^{A}[O(n)].$$

4.1 Reducing to Communication Lower Bounds for XOR-Missing-String

We first show that certain communication lower bounds suffice for proving our barrier result. Roughly speaking, we consider a setting where there are many inputs $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell$ and many protocols $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell$, every protocol $\mathcal{P}_i$ receives every input $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$, and the goal of $\mathcal{P}_i$ is to solve the problem on input $\mathcal{I}_i$. We want to prove lower bounds of the following form: there is a sequence of bad inputs $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$ such that *no protocol $\mathcal{P}_i$ could solve $\mathcal{I}_i$ correctly, even after seeing the inputs of other protocols.* This setting captures win-win analyses: for example, if $\mathcal{P}_i$ fails to solve $\mathcal{I}_i$, then $\mathcal{I}_i$ might contain useful information that helps $\mathcal{P}_j$ solve $\mathcal{I}_j$. In fact, this is indeed what happened in many win-win analyses in complexity theory: see e.g., [CHR24, Section 1.4.2], [LORS24, Section 5.2], and Section A.3 of this paper.

We set the following parameters. Let $n_1 = 2^C$ for some sufficiently large constant C and $n_i = n_{i-1}^4$ for every $i > 1$. For every $i \geq 1$, let $m_i = n_i^{100}$, the i-th input will be an instance of XOR-Missing-String(n_i, m_i). We also define a sequence of error thresholds $\{p_i\}_{i\in\mathbb{N}}$: let $p_1 = 0.6$ and $p_i = p_{i-1} + 1/(10i^2)$ for every $i > 1$, then $p_i < 0.8$ for every i. Looking ahead, it will be useful in the proof of Theorem 4.1 that there is a non-trivial gap (of $1/(10i^2)$) between each p_i and p_{i+1}.

We need the following communication lower bound:

Theorem 4.1. *Let $t \geq 1$ be an integer and $\{Q_{i,j}\}_{1\leq j\leq i\leq t}$ be a set of randomized communication protocols satisfying the following:*

- *In each $Q_{i,j}$, Alice receives $\vec{X} := (X_1, \ldots, X_t)$ and Bob receives $\vec{Y} := (Y_1, \ldots, Y_t)$, where each (X_i, Y_i) is an instance of* XOR-Missing-String(n_i, m_i).
- *The communication complexity of each $Q_{i,j}$ is at most n_i^4 and each $Q_{i,j}$ outputs a string of length n_i.*

Then there exists a sequence of inputs $\{(X_i, Y_i)\}_{i\leq t}$ such that every $Q_{i,j}$ fails to solve the instance (X_i, Y_i). More formally, for every $1 \leq j \leq i \leq t$:

- *either no string is outputted by $Q_{i,j}(\vec{X}, \vec{Y})$ with probability $> p_t$; or*
- *the (unique) string outputted with probability $> p_t$ is not a solution to* XOR-Missing-String *on the input (X_i, Y_i).*

Proof of Theorem 1.5 from Theorem 4.1. Similar to the proof of Theorem 1.3, our oracle encodes a sequence of XOR-Missing-String instances. An infinite sequence of instances $\{(X_i, Y_i)\}_{i\in\mathbb{N}}$ where each (X_i, Y_i) is an instance of XOR-Missing-String(n_i, m_i) corresponds to an oracle A defined as follows:

- For every $i \geq 1$, the truth table of A on input length $2\log m_i$ encodes the instance (X_i, Y_i). More precisely, the truth table of $A \cap \{0,1\}^{2\log m_i}$ restricted to inputs whose first bit is 0 (resp. 1) is equal to X_i (resp. Y_i) padded with zeros. Note that the length of (X_i, Y_i) is $2m_i \cdot n_i \leq 2^{2\log m_i}$.

- If the input length is not of the form $2 \log m_i$, then A always returns 0.

Oracles of this form are called ***well-structured oracle***. When the oracle A corresponds to a finite sequence of t instances $\{(X_i, Y_i)\}_{i \le t}$ (that is, (X_i, Y_i) is the all-zero string for every $i > t$), we say that A is a ***truncated oracle at level t***. Note that we consider oracles that are much "denser" than those considered in Theorem 1.3, in the sense that the lengths of adjacent XOR-Missing-String instances encoded in the oracle are much closer to each other. (Recall that we set $n_i = n_{i-1}^4$ in this section but we set $n_i = 4^{n_{i-1}}$ in Theorem 1.3.)

Converting algorithms to communication protocols. Let $M_1, M_2, \ldots$ be a syntactic enumeration of BPE algorithms, each running in time 2^n. For any integer $i \ge 1, j \ge 1$, let $P_{i,j}$ be the communication protocol that outputs the truth table of $M_j^{\widetilde{A}}$ on inputs of length less than $\log n_i$:

- Alice (resp. Bob) receives as inputs the subfunction A_0 (resp. A_1) of a well-structured oracle A, where A_0 (resp. A_1) denotes the subfunction of A where the first bit is restricted to 0 (resp. 1). Note that since M_j runs in $2^{\log n_i} = n_i$ time, for some large enough $b_i := O(\log n_i)$, it only has access to the instances (X_j, Y_j) for $j \le b_i$. Hence one can view $P_{i,j}$ as a communication protocol where Alice (resp. Bob) receives $(X_1, \ldots, X_{b_i})$ (resp. $(Y_1, \ldots, Y_{b_i})$) as inputs.

- Then, for every $\ell < \log n_i$, Alice and Bob simulate the execution of $M_j^{\widetilde{A}}$ on inputs of length ℓ and obtains a truth table tt_ℓ. They output the concatenation of tt_ℓ for every $\ell < \log n_i$; this is a bit-string of length $n_i - 1$, and we append a bit 0 at the end to make the output length equal to n_i.

- On each input of $M_j^{\widetilde{A}}$, Alice and Bob simulate the execution of $M_j^{\widetilde{A}}$ for $\Theta(n_i)$ times and output the majority answer. This means that when $M_j^{\widetilde{A}}$ satisfies the semantics of a BPP algorithm, $P_{i,j}$ successfully computes the truth table with probability $\ge 1 - 2^{-\Omega(n_i)}$.

Using (a generalization of) Theorem 2.6, $P_{i,j}$ can be implemented by a randomized communication protocol with complexity $O(n_i^3) < n_i^4$ if n_i is large enough.

Constructing the oracle. For each $k \in \mathbb{N}$, define S_k to be the set of truncated oracles at level b_k (i.e., instance sequences $\{(X_i, Y_i)\}_{i \le b_k}$) such that for every $1 \le j \le i \le k$, the protocol $P_{i,j}$ fails to solve (X_i, Y_i) with probability threshold 0.8. Using Theorem 4.1, we can show that

Claim 4.2. *For any $k \ge 1$, S_k is nonempty.*

Proof. This is a consequence of Theorem 4.1. More specifically, we use b_k as the parameter t in Theorem 4.1, let $Q_{i,j} = P_{i,j}$ for every $i \le k$, and $Q_{i,j}$ be a trivial protocol for $i > k$. Note that each $Q_{i,j}$ is a communication protocol with complexity $\le n_i^4$. Hence, by invoking Theorem 4.1, we obtain a truncated oracle A at level b_k, on which every protocol $P_{i,j}$ fails with probability threshold $p_{b_k} < 0.8$. It follows that $A \in S_k$. ◇

Let $t_1 \le t_2$. For instance sequences (i.e., truncated oracles) $A^{(1)} = \{(X_i^{(1)}, Y_i^{(1)})\}_{i \le t_1}$ and $A^{(2)} = \{(X_i^{(2)}, Y_i^{(2)})\}_{i \le t_2}$, we say that $A^{(1)}$ is a ***prefix*** of $A^{(2)}$ if $(X_i^{(1)}, Y_i^{(1)}) = (X_i^{(2)}, Y_i^{(2)})$ for every $i \le t_1$; that is, they agree on the first t_1 instances. We say that a truncated oracle $A \in S_k$ ***extends infinitely*** if there exist infinitely many oracles in $\bigcup_{k \ge i} S_k$ for which A is a prefix. The following two properties of $\{S_k\}$ are easy to see:

- For any $A \in S_i$ and any $j < i$, there exists an oracle in S_j that is a prefix of A.

- The empty oracle (corresponding to the empty sequence of instances) extends infinitely (this follows from Claim 4.2).

We construct a sequence $\{(X_i, Y_i)\}_{i\in\mathbb{N}}$ as follows. Suppose we have constructed a finite sequence $A_\ell = \{(X_i, Y_i)\}_{i\le\ell}$, we maintain the invariant that A_ℓ extends infinitely. We start from $\ell = 0$ and A_ℓ being the empty oracle. To construct $(X_{\ell+1}, Y_{\ell+1})$, we simply find any $(X_{\ell+1}, Y_{\ell+1})$ that would maintain our invariant that $\{(X_i, Y_i)\}_{i\le\ell+1}$ extends infinitely. Since there are only finitely many options for $(X_{\ell+1}, Y_{\ell+1})$, such a choice of $(X_{\ell+1}, Y_{\ell+1})$ always exists.

Let A denote the well-structured oracle corresponding to the sequence $\{(X_i, Y_i)\}_{i\in\mathbb{N}}$ and we now prove that $\mathsf{BPE}^{\widetilde{A}} \subseteq \mathsf{SIZE}^A[O(n)]$. Consider a machine $M_j^{\widetilde{A}}$ and a sufficiently large input length n. Let i be the integer such that $n \in [\log n_{i-1}, \log n_i)$. In this case, we have that $n \ge (\log n_i)/4$. Since n is sufficiently large, we may assume that $i \ge j$. Note that the behavior of $P_{i,j}$ on A is the same as its behavior on $A_{b_i} = \{(X_{i'}, Y_{i'})\}_{i'\le b_i}$ (as M_j on input length $\log n_i$ can only access these inputs). Since $A_{b_i} \in S_i$, we have that $P_{i,j}$ fails to solve the instance (X_i, Y_i) with probability threshold 0.8. There are two cases:

- Either $M_j^{\widetilde{A}}$ does not satisfy the BPP promise. In this case, we are done.
- Or $M_j^{\widetilde{A}}$ satisfies the BPP promise but the unique string outputted by $P_{i,j}$ is a string of the form $x \oplus y$ where $x \in X_i$ and $y \in Y_i$. In this case, there is an A-oracle circuit of size $O(\log m_i) = O(\log n_i)$ that outputs the truth table of $M_j^{\widetilde{A}}$ on inputs of length $< \log n_i$. From this, we can construct a circuit which agrees with $M_j^{\widetilde{A}}$ on inputs of length n and has size $O(\log m_i) = O(n)$. □

4.2 Proof of Theorem 4.1

We prove Theorem 4.1 by induction. The base case where $t = 1$ follows from Corollary 3.8. Now assume that for some $t > 1$, Theorem 4.1 holds for $t-1$ but not for t, and $\{Q_{i,j}\}_{1\le j\le i\le t}$ is a set of protocols that witnesses the failure of Theorem 4.1. We show that $\{Q_{i,j}\}$ can be transformed into a single $\mathsf{PostBPP}$ protocol Q for list-solving XOR-Missing-String(n_t, m_t) that is impossibly efficient in the sense that it contradicts our communication lower bound (Lemma 3.10).

Let (X_t, Y_t) be an input for XOR-Missing-String(n_t, m_t). For $1 \le j \le i \le t$, let $Q'_{i,j}$ be the protocol that takes as input a truncated oracle at level $t-1$, combines it with (X_t, Y_t) to form a truncated oracle at level t, and then simulates the execution of $Q_{i,j}$ on the combined oracle. Consider the protocols $Q'_{i,j}$ for $1 \le j \le i \le t-1$. Since Theorem 4.1 holds for $t-1$, there must exist a truncated oracle A at level $t-1$ on which all these protocols fail. That is, when we combine this oracle A with (X_t, Y_t) to form an oracle A' and run the original protocols $\{Q_{i,j}\}$ on A', every protocols $Q_{i,j}$ with $i \le t-1$ would fail. However, one of the protocols $Q_{i,j}$ must succeed on A'. In this case, the protocol that succeeds must have $i = t$, which means that its output solves (X_t, Y_t).

The above discussion naturally leads to the following $\mathsf{PostBPP}$ communication protocol for list-solving XOR-Missing-String(n_t, m_t): randomly guess a truncated oracle A at level $t-1$, use postselection to ensure that $Q'_{i,j}$ fails for every $1 \le j \le i \le t-1$, and simulate the protocols $Q_{j,t}$ to list-solve the instance (X_t, Y_t). We remark that the combined protocol Q needs to use *postselection* even though the original protocols $Q_{i,j}$ do not; nevertheless, we can still reach a contradiction as Lemma 3.10 applies to $\mathsf{PostBPP}$ communication protocols as well.

See Algorithm 1 for a formal description of the $\mathsf{PostBPP}$ communication protocol Q.

Let K denote the number of bits encoded in a truncated oracle of size $t-1$, formally, $K = \sum_{i<t} 2m_i n_i$. Note that $K = O(t \cdot n_{t-1} \cdot m_{t-1}) = O(n_t^{30}) \ll m_t$.

Algorithm 1: The protocol Q

Input: Alice gets X_t and Bob gets Y_t, where (X_t, Y_t) is an input to XOR-Missing-String(n_t, m_t).

Output: A list of t n_t-bit strings, or $\perp$.

for $i \leftarrow 1$ **to** $t-1$ **do**
  $(X_i, Y_i) \leftarrow$ uniformly random input of XOR-Missing-String(n_i, m_i);
$A \leftarrow$ the truncated oracle at level t that encodes $(X_{\le t}, Y_{\le t})$;
$p \leftarrow (p_{t-1} + p_t)/2$;
for $i \leftarrow 1$ **to** $t-1$ **do**
  for $j \leftarrow 1$ **to** i **do** `// Check whether` $Q_{i,j}$ `fails`
    Simulate the execution of $Q_{i,j}$ on input A,
      repeat for $K \cdot t^{10}$ times using independent randomness;
    $s \leftarrow$ the empirical majority answer;
    $p_{\mathsf{emp}} \leftarrow$ empirical probability of outputting s;
    if $p_{\mathsf{emp}} > p$ *and* s *is a solution to* (X_i, Y_i) **then**
      return $\perp$;
for $j \leftarrow 1$ **to** t **do** `// If all protocols` $Q_{<t,j}$ `fail, run` $Q_{t,\le t}$ `for answers`
  Simulate the execution of $Q_{t,j}$ on input A,
    repeat for $(n_t)^2$ times using independent randomness;
  $s_j \leftarrow$ the empirical majority answer;
return $\langle s_1, \ldots, s_t \rangle$.

Complexity of Q. Recall that the complexity of a PostBPP communication protocol is defined as $k + c$, where k is the length of the public randomness and c is the number of bits communicated (Definition 3.1). The complexity of Q consists of the following parts:

- Sampling $(X_{<t}, Y_{<t})$: This costs at most $K = O(n_t^{30})$ random bits.
- Simulating the protocols $Q_{i,j}$ for each $1 \le j \le i \le t$: This requires communicating at most $t^2 \cdot (n_t)^4 \cdot K \cdot t^{10} = O(n_t^{46})$ bits.

Hence, the complexity of Q is $O(n_t^{46}) \ll m_t$.

We remark that Q also needs to check whether the outputs of $Q_{i,j}$ is a solution of (X_i, Y_i) for each $1 \le j \le i \le t-1$. However, since $(X_{<t}, Y_{<t})$ is sampled using public randomness, this costs no communication.

Success probability of Q. For a fixed input (X_t, Y_t), let E_{t-1} denote the following event: for the oracle A constructed in Algorithm 1, all the protocols $Q_{i,j}$ with $1 \le j \le i \le t-1$ fail on A with probability threshold p_{t-1}. Let E_t denote the event that all these protocols (i.e., $\{Q_{i,j}\}_{1 \le j \le i \le t-1}$) fail with probability threshold p_t. Note that E_{t-1} implies E_t, which means that $\Pr[E_{t-1}] \le \Pr[E_t]$.

In the following, we lower bound $\Pr[Q \text{ is correct} \mid Q(X, Y) \neq \perp]$ by considering the behavior of Q under three events:

1. **Bad case:** Conditioned on $\neg E_t$, Q outputs $\perp$ with high probability (Claim 4.4). Therefore, this case only negligibly affects $\Pr[Q \text{ is correct} \mid Q(X, Y) \neq \perp]$.

2. **Good case:** Conditioned E_{t-1}, with high probability, Q does not output $\perp$ (Claim 4.4), and is correct (Claim 4.5). We also show that $\Pr[E_{t-1}]$ is large (Claim 4.3), so that this case has a big contribution to $\Pr[Q \text{ is correct} \mid Q(X,Y) \neq \perp]$.

3. **Intermediate case:** Conditioned on $E_t \wedge \neg E_{t-1}$, Q behaves abnormally, in that we cannot bound $\Pr[Q(X,Y) \neq \perp \mid E_t \wedge \neg E_{t-1}]$. However, we know that conditioned on $E_t \wedge \neg E_{t-1}$ and $Q(X,Y) \neq \perp$, Q is correct with high probability (Claim 4.5). Therefore, regardless of $\Pr[Q(X,Y) \neq \perp \mid E_t \wedge \neg E_{t-1}]$, this case does not hurt $\Pr[Q \text{ is correct} \mid Q(X,Y) \neq \perp]$.

Let S denote the event that, when running the simulation, one of the output strings $s_1, \ldots, s_t$ solves (X_t, Y_t). We define S in such a way that S is independent from whether $Q(X_t, Y_t)$ outputs $\perp$. That is, it is possible that although $Q(X_t, Y_t)$ outputs $\perp$, had Q performed the simulation, it would have solved (X_t, Y_t). In this case, we say that S holds.

We start by proving that the good case happens with significant probability:

Claim 4.3. $\Pr[E_{t-1}] \geq 2^{-K}$.

Proof. Consider the protocols $\{Q'_{i,j}\}$ $(1 \leq j \leq i < t)$, where $Q'_{i,j}$ takes a truncated oracle A' of size $t-1$, combines it with (X_t, Y_t), and then simulates $Q_{i,j}$ on A. Since Theorem 4.1 holds for $t-1$, applying it on $\{Q'_{i,j}\}$ implies that there exists at least one truncated oracle A' of size $t-1$, on which all $Q'_{i,j}$ fail with probability threshold p_{t-1}. ◇

Next, we prove bounds for $\Pr[Q(X_t, Y_t) \neq \perp]$ in the good and bad cases:

Claim 4.4. *The following items hold:*

- ***Bad case:*** $\Pr[Q(X_t, Y_t) \neq \perp \mid \neg E_t] \leq 2^{-2K}$.
- ***Good case:*** $\Pr[Q(X_t, Y_t) \neq \perp \mid E_{t-1}] \geq 1 - 2^{-2K}$.

Proof. If E_t does not happen, then there exists $1 \leq j \leq i < t$ such that when running $Q_{i,j}$ on A, there exists a (unique) string s outputted with probability $> p_t$, and s is a valid solution to (X_i, Y_i).

Using a straightforward Chernoff bound, when running Lines 5-12 in Algorithm 1, with probability $1 - 2^{-2K}$, the empirical probability p_{emp} of outputting s will be greater than $p_t \cdot (1 - 0.01/t^2) > (p_t + p_{t-1})/2$. This means that Q outputs $\perp$ with probability $\geq 1 - 2^{-2K}$.

The second bullet can be proved similarly. ◇

Finally, in the good and intermediate case, we show that S holds with high probability.

Claim 4.5. *In Algorithm 1, for a truncated oracle A at level $t-1$, if E_t holds for A, we have that*

$$\Pr[S \mid A \text{ is sampled}] \geq 1 - 2^{-\Omega((n_t)^2)}.$$

Proof. By our assumption on t, for any possible A, at least one of the protocols $Q_{i,j}$ $(1 \leq j \leq i \leq t)$ does not fail with probability threshold p_t. If E_t happens, then all the protocols $Q_{i,j}$ with $i < t$ (i.e., those that operate on smaller input lengths) fail on A with probability threshold p_t. Therefore, at least one of the protocols $Q_{t,j}$ succeeds on A. In this case, when we simulate $Q_{t,j}$ in Q, with probability $1 - 2^{-\Omega(n_t^2)}$, the empirical majority answer s_j is indeed the majority answer of $Q_{t,j}$, which is a valid solution for (X_t, Y_t). ◇

Combining the previous results proves that Q succeeds with high probability.

Claim 4.6. *When running the combined protocol Q on any input (X_t, Y_t),*

$$\Pr[S \mid Q(X_t, Y_t) \neq \bot] \geq 1 - 2^{-\Omega(n_t^2)}.$$

Proof. Let $G = E_{t-1} \vee \neg E_t$ (note that E_{t-1} and $\neg E_t$ are disjoint). We have that

$$\begin{aligned} &\Pr[S \mid Q(X_t, Y_t) \neq \bot] \\ \geq\ &\min\{\Pr[S \mid Q(X_t, Y_t) \neq \bot \wedge G], \Pr[S \mid Q(X_t, Y_t) \neq \bot \wedge \neg G]\}. \end{aligned} \tag{3}$$

The second term is at least

$$\min_{A: G \text{ does not hold for the oracle } A} \Pr[S \mid Q(X_t, Y_t) \neq \bot \wedge A \text{ is sampled}],$$

which is at least $1 - 2^{-\Omega((n_t)^2)}$ by Claim 4.5 (note that $\neg G = \neg E_{t-1} \wedge E_t$, which implies that E_t holds for A).

For the first term, we have that

$$\begin{aligned} &\Pr[S \mid Q(X_t, Y_t) \neq \bot \wedge G] \\ \geq\ &\frac{\Pr[S \wedge Q(X_t, Y_t) \neq \bot \mid G]}{\Pr[Q(X_t, Y_t) \neq \bot \mid G]} \\ \geq\ &\frac{\Pr[S \wedge Q(X_t, Y_t) \neq \bot \wedge G]}{\Pr[Q(X_t, Y_t) \neq \bot \mid E_{t-1}] \cdot \Pr[E_{t-1} \mid G] + \Pr[Q(X_t, Y_t) \neq \bot \mid \neg E_t] \cdot \Pr[\neg E_t \mid G]} \\ \geq\ &\frac{\Pr[S \wedge Q(X_t, Y_t) \neq \bot \wedge E_{t-1}]}{\Pr[Q(X_t, Y_t) \neq \bot \mid E_{t-1}] \cdot \Pr[E_{t-1}] + \Pr[Q(X_t, Y_t) \neq \bot \mid \neg E_t]} \qquad (E_{t-1} \text{ is contained in } G) \\ \geq\ &\frac{\Pr[S \mid E_{t-1}] \cdot \Pr[Q(X_t, Y_t) \neq \bot \mid E_{t-1}] \cdot \Pr[E_{t-1}]}{\Pr[Q(X_t, Y_t) \neq \bot \mid E_{t-1}] \cdot \Pr[E_{t-1}] + \Pr[Q(X_t, Y_t) \neq \bot \mid \neg E_t]}, \end{aligned}$$

where the last line uses the fact that S and $Q(X_t, Y_t) \neq \bot$ are independent.

Using Claim 4.4 and Claim 4.5, we have that

$$\Pr[S \mid Q(X_t, Y_t) \neq \bot \wedge G] \geq \frac{(1 - 2^{-\Omega(n_t^2)}) \cdot (1 - 2^{-2K}) \cdot \Pr[E_{t-1}]}{\Pr[E_{t-1}] + 2^{-2K}}.$$

Since $\Pr[E_{t-1}] \geq 2^{-K}$ by Claim 4.3, we have that

$$\Pr[S \mid Q(X_t, Y_t) \neq \bot \mid G] \geq \frac{(1 - 2^{-\Omega(n_t^2)}) \cdot (1 - 2^{-2K}) \cdot 2^{-K}}{2^{-K} + 2^{-2K}} \geq 1 - 2^{-\Omega(n_t^2)}.$$

Therefore, both terms in (3) are at least $1 - 2^{-\Omega(n_t^2)}$. ◇

Putting everything together. Finally, we prove Theorem 4.1:

Proof of Theorem 4.1. The base case of $t = 1$ follows from Corollary 3.8.

For $t > 1$, assume that the lemma holds for $t - 1$ but not for t. This implies that Algorithm 1 is a PostBPP communication protocol Q that:

- takes an input of XOR-Missing-String(n_t, m_t), and outputs t n_t-bit strings,
- has complexity $\leq O(n_t^{46}) = o(m_t)$, and
- succeeds with probability $\geq 1 - 2^{-\Theta((n_t)^2)} \geq 1 - 2^{-\omega(n_t \cdot t)}$ on any input.

Such a protocol is impossible by Lemma 3.10. □

5 Almost-Everywhere Algebrization Barrier for Robust $\mathsf{MA_E}$

5.1 Definitions

Definition 5.1. An $\mathsf{MA} \cap \mathsf{coMA}$ ***algorithm*** M is associated with two verifiers V_0, V_1, which are randomized algorithms that take an input x and a proof π from Merlin. We say that

- A verifier V_k ***accepts*** x, if there exists a proof π such that $\Pr[V_k(x,\pi)=1] \geq 2/3$.
- A verifier V_k ***rejects*** x, if for any proof π, we have that $\Pr[V_k(x,\pi)=1] \leq 1/3$.

We define the notion of ***robust*** algorithms. Intuitively, an MA algorithm is robust if its truth table is never easy for A-oracle circuits. We do however allow an algorithm to be semantically incorrect on some oracle, which is not considered as an error.

Formally, we say that M is ***robust w.r.t. $h(n)$ on length n*** if, for *any* oracle A and any 2^n-bit string w that has $h(n)$ size A-oracle circuits, there exists at least one input x of length n, such that $V_{w_x}^{\widetilde{A}}$ rejects x. That is, Merlin cannot convince Arthur to output the truth table w.

The following definitions characterize FMA communication protocols. Such protocols are able to compute the truth tables of $\mathsf{MA} \cap \mathsf{coMA}$ algorithms.

Definition 5.2. An FMA ***communication protocol*** P for a search problem f (over domain $\mathcal{X} \times \mathcal{Y}$ and range $\mathcal{O}$) is defined as a set of randomized protocols $P_{w,\pi}$, for each pair of answer $w \in \mathcal{O}$ and proof π from Merlin. Let k denote the length of the proof. That is, $\pi \in \{0,1\}^k$.

We say that P ***solves (X,Y)***, if

- For some w, π where w is a solution to (X,Y), $\Pr[P_{w,\pi}(X,Y)=1] \geq 2/3$.
- For any w, π where w is *not* a solution to (X,Y), $\Pr[P_{w,\pi}(X,Y)=1] \leq 1/3$.

The ***complexity*** of P is defined as $k+C$, where C is the maximum complexity of any randomized protocol $P_{w,\pi}$. Here, the complexity of a randomized protocol is defined as the maximum number of bits communicated on any input, plus the number of random bits used by the protocol. Note that we only allow public randomness (which is available to both Alice and Bob but not Merlin).

A protocol P is ***robust***, if for any (X,Y) and for any w, π where w is *not* a solution to (X,Y), $\Pr[P_{w,\pi}(X,Y)=1] \leq 1/3$.

5.2 XOR-Missing-String Lower Bounds Against Robust Protocols

The main engine of our barrier is the following lemma, which shows that robust protocols are much weaker than general protocols.

Lemma 1.8 (Restated)**.** *Let $n \geq 1, 1 \leq m < 2^{n/2}$ be parameters. Let P be a robust* FMA *communication protocol of complexity C attempting to solve* XOR-Missing-String(n,m)*. Then the fraction of inputs that P solves is at most*

$$2^{-\Omega(m/C)+O(C+n)}.$$

Proof. We only prove the lemma for the case where $C < m/2$, since otherwise the bound is straightforward.

As an FMA communication protocol, P can be viewed as a collection of at most $2^n \cdot 2^C$ randomized protocols, where each protocol $P_{w,\pi}$ corresponds to a pair (w,π) of candidate answer and

proof, and $P_{w,\pi}$ outputs 1 if and only if Arthur accepts Merlin's proof π, which proves that w is a solution.

Let $P'_{w,\pi}$ denote an error-reduced version of $P_{w,\pi}$. Specifically, $P'_{w,\pi}$ runs $P_{w,\pi}$ repeatedly for $\Theta(m/C)$ times, each time using independent randomness, and then outputs the majority answer. We choose the number of repetitions carefully, so that

1. $P'_{w,\pi}$ has complexity $< m/2$.
2. On input (X, Y), if w is not a solution, then
$$\Pr[P'_{w,\pi}(X, Y) = 1] \leq 2^{-\Omega(m/C)}. \tag{4}$$
3. If P solves XOR-Missing-String on (X, Y), then for some (w, π) where w is a solution to (X, Y), we have that
$$\Pr[P'_{w,\pi}(X, Y) = 1] \geq 1 - 2^{-\Omega(m/C)}.$$

Item 3 means that the fraction of inputs solvable by P is at most

$$(1 + 2^{-\Theta(m/C)}) \cdot \sum_{w,\pi} \Pr_{(X,Y)}[P'_{w,\pi}(X, Y) = 1],$$

where the probability is over the choice of (X, Y) and the internal randomness of $P'_{w,\pi}$. To bound this value, we think of $P'_{w,\pi}$ as a distribution of deterministic protocols. For each deterministic protocol, we have the following claim, which is a relation between the probabilities of true and false positives.

Claim 5.3. *Let w be an n-bit string, and let P be a deterministic protocol of complexity C, which outputs 0 or 1. Let*

$$p = \Pr_{(X,Y)}[P(X, Y) = 1],$$
$$p_{wrong} = \Pr_{(X,Y)}[P(X, Y) = 1 \wedge w \textit{ does not solve } (X, Y)].$$

We have that

$$p \leq 2^{-m+2} \cdot 2^C + p_{wrong} \cdot 2^{2n+2}.$$

Proof. P can be characterized as a collection of at most 2^C rectangles that are pairwise disjoint and cover the entire input space, such that the output of P is the same for inputs that belong to the same rectangle. We say that a rectangle is *large* if it has size $\geq 2^{2nm-m+2}$, and say that a rectangle is *small* otherwise.

We fix one large rectangle, and consider its contribution to p and p_{wrong}. Using Lemma 3.2, when a rectangle R is large, there must exist a $2^{-(2n+2)}$ fraction of input in R, on which w is not a solution. Therefore, if P outputs 1 on R, then R contributes $|R|/2^{2nm}$ to p, and contributes at least $|R| \cdot 2^{-(2n+2)}/2^{2nm}$ to p_{wrong}. We therefore have the following relation between p and p_{wrong}:

$$p \leq 2^{-m+2} \cdot 2^C + p_{\text{wrong}} \cdot 2^{2n+2},$$

where the first term upper bounds the total size of small rectangles. ◇

For each $P'_{w,\pi}$, by applying Claim 5.3 on each deterministic protocol in its distribution, we have that

$$\Pr_{(X,Y)}[P'_{w,\pi}(X,Y)=1] \le 2^{-m+2}\cdot 2^C + \Pr_{(X,Y)}[P'_{w,\pi}(X,Y)=1 \wedge w \text{ does not solve } (X,Y)]\cdot 2^{2n+2}. \tag{5}$$

By (4), whenever w does not solve (X,Y), we have that $\Pr[P'_{w,\pi}(X,Y)=1] \le 2^{-\Omega(m/C)}$, where the probability is over the internal randomness of $P'_{w,\pi}$. Summing over all (X,Y), we have that

$$\Pr_{(X,Y)}[P'_{w,\pi}(X,Y)=1 \wedge w \text{ does not solve } (X,Y)] \le 2^{-\Omega(m/C)}.$$

Plugging this into (5), we thus have that

$$\Pr_{(X,Y)}[P'_{w,\pi}(X,Y)=1] \le 2^{-\Omega(m/C)+O(C+n)}.$$

Summing this over for every $P'_{w,\pi}$, we have that (note that the number of pairs w,π is at most $2^{O(C+n)}$)

$$(1+2^{-\Theta(m/C)})\cdot \sum_{w,\pi} \Pr_{(X,Y)}[P'_{w,\pi}(X,Y)=1] \le 2^{-\Omega(m/C)+O(C+n)}. \qquad \square$$

5.3 The Barrier

We say that a function $h(n)$ is ***super-half-exponential***, if h is strictly monotonically increasing, $h(n)=\omega(n)$, and $h(h(n)/n) \ge 2^n$. In the rest of this section, let $h(n)$ be any fixed super-half-exponential function.

Let $\mathrm{Rob}_{h(n)}$-$\mathsf{MA_E}$ denote the class of languages decidable by algorithms that are in $\mathsf{MA_E}\cap\mathsf{coMA_E}$ and are robust w.r.t. $h(n)$ on every input length. We present the following barrier result.

Theorem 5.4. *There exists an oracle A such that $(\mathsf{E}^{\widetilde{A}} \cup (\mathrm{Rob}_{h(n)}\text{-}\mathsf{MA_E})^{\widetilde{A}}) \subseteq \mathsf{SIZE}^A[h(n)]$, where $\widetilde{A}$ denotes the multilinear extension of A.*

5.3.1 Oracle Structure

In this proof, we say that an oracle A is ***well-structured***, if it encodes one instance of XOR-Missing-String for each input length i. Formally, for every $i\in\mathbb{N}$, the truth table of A on inputs of length $h(i)$ encodes an instance (X_i,Y_i) of XOR-Missing-String$(n_i=2^i, m_i=2^{h(i)/10})$ (note that this instance can be described in $2n_im_i \le 2^{h(i)}$ bits), where X_i (resp. Y_i) is encoded in the truth table for A on inputs that start with 0 (resp. 1). When querying A on an input of length other than $h(i)$ for some i, the oracle returns 0.

These parameters are chosen such that, if an algorithm $M^{\widetilde{A}}$ is semantically correct and does not solve (X_i,Y_i) (i.e., its truth table on inputs of length $=i$ is not a solution), then there exists an A-oracle circuit of size less than $h(i)$ that computes the truth table of $M^{\widetilde{A}}$ on inputs of length i. Indeed, the truth table must be of the form $x\oplus y$, where $x\in X_i$ and $y\in Y_i$. Such strings can be computed by A-oracle circuits of size $2\log m_i + O(i) < h(i)$, by hardcoding the index of x and y in X_i and Y_i respectively.

5.3.2 Converting Algorithms to Protocols

Similar to the previous barriers, we start by enumerating the algorithms in E and in $\mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}$. The two types of algorithms are enumerated separately. For E, let $M_1, M_2, \ldots$ be an enumeration of deterministic algorithms in $\mathsf{DTIME}[2^n]$. Let $P_{i,j}$ be a deterministic protocol that outputs the truth table of $M_j^{\widetilde{A}}$ on inputs of length i. Note that $P_{i,j}$ runs in time $2^{O(i)}$. We aim to ensure that, for any sufficiently large i and $1 \leq j \leq i$, $P_{i,j}$ does not solve (X_i, Y_i).

For $\mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}$, let $(V'_{j,0}, V'_{j,1})_{j\in\mathbb{N}}$ be a syntactic enumeration of pairs of verifiers, where each verifier is a randomized algorithm that receives an input x and a proof π and runs in time $2^{|x|}$ (in particular, the length of π is at most $2^{|x|}$). Let M'_j be the $\mathsf{MA} \cap \mathsf{coMA}$ algorithm associated with $(V'_{j,0}, V'_{j,1})$. For $i \in \mathbb{N}$ and $1 \leq j \leq i$, let $Q_{i,j}$ be the FMA protocol of complexity $2^{O(i)}$ that outputs the truth table of $(M'_j)^{\widetilde{A}}$ on inputs of length i, in the case that the algorithm is robust. More formally, $Q_{i,j}$ is defined as follows.

- If $(M'_j)^{(-)}$ is not robust w.r.t. $h(i)$ on inputs of length i, then $Q_{i,j}$ rejects every pair (w, π) of candidate answer and proof (i.e., it is the trivial robust protocol).
- Otherwise, $Q_{i,j}$ simulates the execution of $(M'_j)^{\widetilde{A}}$ on inputs of length i (repeatedly for $2^{O(i)}$ times on each input) and outputs its truth table.

  More specifically, $Q_{i,j}$ corresponds to the following set of verifiers $\{V_{w,\pi}\}$: $V_{w,\pi}$ interprets π as a concatenation of proofs $\{\pi_x : x \in \{0,1\}^i\}$, one for each input of length i. For each input x of length i, $V_{w,\pi}$ checks whether the x-th bit of w (denoted as w_x) is the output of $(M'_j)^{\widetilde{A}}(x)$, by simulating the execution of $(M'_j)^{\widetilde{A}}(x)$ for 2^i times. In each simulation, $V_{w,\pi}$ checks whether $(M'_j)^{\widetilde{A}}(x) = w_x$ when given the proof π_x. If for every x, the majority of simulations accept, then $V_{w,\pi}$ outputs 1; otherwise, it outputs 0.

  Each $V_{w,\pi}$ has complexity $2^{O(i)}$, and the length of π is also $2^{O(i)}$. Therefore, the complexity of $Q_{i,j}$ is $2^{O(i)}$. Moreover, $Q_{i,j}$ is a robust protocol for (X_i, Y_i). This is because M'_j is robust, which means that for any non-solution w of (X_i, Y_i), there exists some x such that $(V'_{j,s_x})^{\widetilde{A}}$ rejects x. Therefore, when $V_{w,\pi}$ checks x, the majority of simulations will reject.

In the following construction, we aim to ensure that, for any sufficiently large i and $1 \leq j \leq i$, $Q_{i,j}$ does not solve (X_i, Y_i). If this holds, then for any algorithm M'_j in $\mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}$, there exists a family of A-oracle circuits of size $h(i)$ that computes the truth table of $(M'_j)^{\widetilde{A}}$.

Note that for every $i \in \mathbb{N}$ and $1 \leq j \leq i$, since $P_{i,j}$ and $Q_{i,j}$ both simulate algorithms that run in time 2^i, both protocols only access the oracle A on inputs of length $\leq 2^i$. In other words, when $2^i < h(i')$, these protocols cannot access $(X_{i'}, Y_{i'})$.

5.3.3 The Inductive Guarantee

We construct the oracle A inductively, where at step i, we maintain a rectangle R_i of well-structured oracles (i.e., $R_i = \mathcal{A}_i \times \mathcal{B}_i$ where $\mathcal{A}_i$ is a subset of Alice's inputs and $\mathcal{B}_i$ is a subset of Bob's inputs). The goal is to refute all protocols by shrinking the rectangle, where we hope to guarantee that for any i, the protocols $P_{\leq i,j}, Q_{\leq i,j}$ fail on any oracle in R_i (in reality, we achieve a weaker guarantee; see below).

The following constant parameters are used in the construction. The big-O notations will not hide any dependence on these constants.

- Let $c_1 \geq 1$ be a constant such that any protocol $P_{i,j}$ or $Q_{i,j}$ have complexity at most $2^{c_1 i} = n_i^{c_1}$.
- Let $c_2 \geq 1$ be a constant such that, for any $i \geq i_0$ and any robust protocol Q of complexity $n_i^{c_1}$ attempting to solve XOR-Missing-String(n_i, m_i), Q solves at most $2^{-c_2 m_i / n_i^{c_1}}$ fraction of all inputs. Such a constant exists by Lemma 1.8.
- Let c_3, c_4 be constants such that $c_1 \ll c_3 \ll c_4$.
- Let i_0 be a sufficiently large constant that depends on c_1 through c_4. We only refute the protocols $P_{i,j}, Q_{i,j}$ where $i \geq i_0$.

We ensure the following properties for R_i. Intuitively, $R_1 \supset R_2 \supset \ldots$ is a sequence of rectangles representing increasingly stronger restrictions on the oracle A, where R_i (ideally) refutes all protocols $P_{i',j}, Q_{i',j}$ where $i_0 \leq i' \leq i$.

1. **Containment:** For any $i > 1$, R_i is a subrectangle of R_{i-1}.
2. **Measurability:** To decide whether an oracle A is in R_i, we only need to look at the first $h(i)$ instances $(X_{\leq h(i)}, Y_{\leq h(i)})$ of A.

   Although R_i is a set of uncountably many oracles, measurability implies that we can treat it as a set of finite oracles (up to length $h(i)$). Hence, we can reasonably talk about the "size" of R_i, as well as the "uniform distribution" over R_i.
3. **Common prefix:** All the oracles in R_i agree on the first i instances $(X_{\leq i}, Y_{\leq i})$. We denote this common prefix as $(X_1^{\text{fixed}}, Y_1^{\text{fixed}}), \ldots, (X_i^{\text{fixed}}, Y_i^{\text{fixed}})$. Since for each $i \geq 1$, R_{i+1} is a subrectangle of R_i, we have that $(X_1^{\text{fixed}}, Y_1^{\text{fixed}}), \ldots, (X_i^{\text{fixed}}, Y_i^{\text{fixed}})$ is also a prefix of every $R_{i'}$ $(i' > i)$.
4. **Largeness:** R_i contains at least a $2^{-n_i^{c_3}}$ fraction of oracles that agree on $(X_{\leq i}^{\text{fixed}}, Y_{\leq i}^{\text{fixed}})$. Formally, for a random well-structured oracle A,

$$\Pr_A[A \in R_i \mid A \text{ agrees with } (X_{\leq i}^{\text{fixed}}, Y_{\leq i}^{\text{fixed}})] \geq 2^{-n_i^{c_3}}.$$

5. **Refuting deterministic protocols:** For $i_0 \leq i' \leq i$, $1 \leq j \leq i'$ and any oracle $A \in R_i$, $P_{i',j}$ does not solve $(X_{i'}, Y_{i'})$ on A.
6. **Refuting robust protocols:** For $i_0 \leq i' \leq i$, $1 \leq j \leq i'$:
   (a) **Small protocols:** If $2^{i'} \leq h(i)$, then for any oracle $A \in R_i$, $Q_{i',j}$ does not solve $(X_{i'}, Y_{i'})$ on A. (In this case, $Q_{i',j}$ only has access to the first i instances in A, which are equal to $\{(X_{i''}^{\text{fixed}}, Y_{i''}^{\text{fixed}})\}_{i'' \leq i}$.)
   (b) **Large protocols:** If $2^{i'} > h(i)$, then

$$\Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \mid A \in R_i] \leq 2^{-c_2 m_{i'} / n_{i'}^{c_1} + n_i^{c_4}}.$$

   We provide some intuition about this requirement, in particular about the right-hand side $2^{-c_2 m_{i'} / n_{i'}^{c_1} + n_i^{c_4}}$. The first term $c_2 m_{i'} / n_{i'}^{c_1}$ is the exponent in the upper bound of Lemma 1.8, which is the probability that $Q_{i',j}$ solves $(X_{i'}, Y_{i'})$ on a random oracle. Since R_i is only a subset of all oracles, the probability of $Q_{i',j}$ solving a random oracle in R_i might be larger, and this probability could increase as we reduce the size of R_i. This behavior is captured by the second term $n_i^{c_4}$ in the exponent.

Why half-exponential? The probability bound of Property 6b is the reason that our proof requires $h(n)$ to be super-half-exponential. In order for (6b) to be nontrivial, we require that $2^{-c_2 m_{i'}/n_{i'}^{c_1}+n_i^{c_4}} < 1$ for any $i', i \geq i_0$ such that $2^{i'} > h(i)$. Recall that $m_i = 2^{h(i)/10}$ and $n_i = 2^i$, this requirement loosely translates to $h(i') \geq i$ whenever $2^{i'} > h(i)$ (up to some polynomial). Assuming that h is monotone, $h(i') \geq i$ is equivalent to $h(h(i')) \geq h(i)$, and our requirement becomes $h(h(i')) > 2^{i'}$ for any i', i.e., h must be super-half-exponential.

Formally, given that h is super-half-exponential, we can show that:

Claim 5.5. *For $i_0 \leq i' \leq i$ and $1 \leq j \leq i'$ such that $h(i-1) < 2^{i'} \leq h(i)$, we have that $2^{-c_2 m_{i'}/n_{i'}^{c_1}+n_i^{c_4}} < 1$.*

Proof. The above inequality is equivalent to

$$m_{i'} > n_{i'}^{c_1} \cdot n_i^{c_4}/c_2.$$

Recall that $n_{i'} = 2^{i'}$ and $m_{i'} = 2^{h(i')/10}$. Taking logarithm on both sides, this is equivalent to showing that

$$h(i')/10 > c_1 \cdot i' + c_4 \cdot i + O(1) = c_4 \cdot i \cdot (1 + o(1)).$$

Suppose that this does not hold. Then we have

$$h(i')/O(c_4) \leq i - 1.$$

Since h is monotone, we have that

$$h(h(i')/O(c_4)) \leq h(i-1).$$

However, since h is super-half-exponential, we have that $h(h(i')/i') \geq 2^{i'}$. This contradicts $2^{i'} > h(i-1)$. □

If we can construct such a sequence of rectangles, then the oracle $A^{\text{fixed}} = (X_i^{\text{fixed}}, Y_i^{\text{fixed}})$ (formally, $A^{\text{fixed}}(x) = 1$ if and only if $A'(x) = 1$ for every $A' \in R_n$, where n is sufficiently large) proves the theorem, because it lies in every rectangle R_n, and thus refutes every protocol.

5.3.4 The Construction

When $i < i_0$, we let $(X_{\leq i}^{\text{fixed}}, Y_{\leq i}^{\text{fixed}})$ be an arbitrary prefix, and let R_i be the set of all oracles that agree with $(X_{\leq i}^{\text{fixed}}, Y_{\leq i}^{\text{fixed}})$.

When $i \geq i_0$, suppose that we have constructed R_{i-1}. To construct R_i, we start by letting $R_i^0 \leftarrow R_{i-1}$, then gradually shrink R_i^0 to ensure that the above properties are satisfied.

Step 1: Refuting the deterministic protocols. We start by finding a large subrectangle in R_i^0 on which all deterministic protocols fail. Note that we only have to consider protocols $P_{i,j}$ for each $1 \leq j \leq i$, since the other protocols $P_{<i,j}$ already fail on every oracle in R_{i-1}.

For each $1 \leq j \leq i$, since the deterministic protocol $P_{i,j}$ runs in time $n_i^{c_1}$, there exists a subrectangle R' of R_i^0, such that R' contains at least $2^{-n_i^{c_1}}$ fraction of R_i^0, and $P_{i,j}$ outputs the same string for any oracle in R'.

Next, we use Corollary 3.4 to find a subrectangle R'' of R', such that R'' contains at least $2^{-O(n_i)}$ fraction of R', and that for any oracle $A \in R''$, $P_{i,j}$ does not solve (X_i, Y_i) on A.

Corollary 3.4 (Restated)**.** *Let $n \geq 1, 1 \leq m < 2^{n/2}, a \geq 0$ be integers. Let $R \subseteq \{0,1\}^{nm+a} \times \{0,1\}^{nm+a}$ be a rectangle of size at least $2^{2nm+2a-m+2}$ (i.e., R is of the form $X \times Y$ for some $X, Y \subset \{0,1\}^{nm+a}$). Each element in R is interpreted as having the form $(X \circ t_x, Y \circ t_y)$, where (X, Y) is an input to* XOR-Missing-String(n, m)*, and t_x, t_y are two a-bit strings.*

Let s be any n-bit string. There exists some R', which is a subrectangle of R and has size at least $2^{-2n-2}|R|$, such that for any $(X \circ t_x, Y \circ t_y) \in R'$, s is not a solution to XOR-Missing-String *on (X, Y).*

Note that Corollary 3.4 is not naïvely applicable to R', because the oracles in R' are infinitely long. However, by the property of measurability (2), we may pretend that the oracles in R' only encode finitely many instances (X_*, Y_*). It is then possible to interpret the instances $(X_{\neq i}, Y_{\neq i})$ as the auxiliary input and apply Corollary 3.4. Finally, we update $R_i^0 \leftarrow R''$ to refute $P_{i,j}$.

After considering all $1 \leq j \leq i$, we have that, for any $1 \leq j \leq i$ and any oracle $A \in R_i^0$, $P_{i,j}$ does not solve (X_i, Y_i) on A. Let R_i^1 denote the resulting rectangle after refuting all $P_{i,j}$. Note that the size of R_i^1 is at least a $2^{-i \cdot O(n_i^{c_1})} \geq 2^{-O(n_i^{c_1})}$ fraction of the size of R_{i-1}. After this, the current rectangle R_i^1 satisfies all properties except 3 and 6:

- Containment is satisfied because we only shrink the rectangle.
- Measurability is satisfied because it was satisfied by R_{i-1}, and we only look at the first $h(i)$ instances of A to decide the answer of each $P_{i,j}$. Also, when applying Corollary 3.4, it suffices to check the value of (X_i, Y_i). To see this, recall that in Corollary 3.4, the subrectangle is constructed by fixing some n_i-bit string s, and restricting the rectangle to those oracles for which both X_i and Y_i contain s.
- R_i^1 is still large because we only shrink it by a factor of $2^{-n_i^{O(c_1)}}$:

$$\Pr_A[A \in R_i^1 \mid A \text{ agrees with } (X_{\leq i-1}^{\text{fixed}}, Y_{\leq i-1}^{\text{fixed}})] \geq 2^{-n_{i-1}^{c_3} - n_i^{O(c_1)}} = 2^{-O(n_{i-1}^{c_3})}. \tag{6}$$

  Note that this is not exactly the same as Largeness (Property 4) because we have not yet fixed $(X_i^{\text{fixed}}, Y_i^{\text{fixed}})$.
- R_i^1 refutes all deterministic protocols because R_{i-1} already refutes all $P_{i',j}$ where $i_0 \leq i' \leq i-1$, and we have just ensured that R_i^1 refutes all $P_{i,j}$.

As for the robust protocols, we now have the following guarantee: For $i_0 \leq i' \leq i$, $1 \leq j \leq i'$,

(a) **Small protocols:** If $2^{i'} \leq h(i-1)$, then for any oracle $A \in R_i^1$, $Q_{i',j}$ does not solve $(X_{i'}, Y_{i'})$ on A. This holds by transitivity from R_{i-1}.

(b) **Large protocols:** If $2^{i'} > h(i-1)$, then

$$\Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \mid A \in R_i^1] \leq 2^{-c_2 m_{i'}/n_{i'}^{c_1} + O(n_{i-1}^{c_4})}. \tag{7}$$

For $i' < i$, this holds because R_{i-1} satisfies Property 6 on $i-1$, i.e., it refutes the protocol $Q_{i',j}$ if $i' < i$. Since we only shrink R_i^1 by a $2^{-n_i^{O(c_1)}}$ factor, the success probability of $Q_{i',j}$ only increases by a $2^{n_i^{O(c_1)}} = 2^{O(n_{i-1}^{c_4})}$ factor.

For $i' = i$, our bound comes from applying Lemma 1.8 to the set of all oracles (that agree with $(X^{\text{fixed}}_{\le i-1}, Y^{\text{fixed}}_{\le i-1})$). Note that

$$\Pr_A[Q_{i,j} \text{ solves } (X_i, Y_i) \text{ on } A \mid A \text{ agrees with } (X^{\text{fixed}}_{\le i-1}, Y^{\text{fixed}}_{\le i-1})] \le 2^{-c_2 m_i / n_i^{c_1}}, \tag{8}$$

which is a corollary of Lemma 1.8. Indeed, if (8) does not hold, then we can fix an infinite sequence $(X^{\text{bad}}_{>i}, Y^{\text{bad}}_{>i})$ of instances, such that

$$\Pr_A[Q_{i,j} \text{ solves } (X_i, Y_i) \text{ on } A \mid A \text{ agrees with } (X^{\text{fixed}}_{\le i-1}, Y^{\text{fixed}}_{\le i-1}) \text{ and } (X^{\text{bad}}_{>i}, Y^{\text{bad}}_{>i})] > 2^{-c_2 m_i / n_i^{c_1}}.$$

Using this, we can construct a robust protocol Q that breaks the bound of Lemma 1.8: On input (X_i, Y_i), Q combines it with $(X^{\text{fixed}}_{\le i-1}, Y^{\text{fixed}}_{\le i-1})$ and $(X^{\text{bad}}_{>i}, Y^{\text{bad}}_{>i})$ to form a well-structured oracle A, then simulates $Q_{i,j}$ on A. The success probability of Q is exactly the above probability, which contradicts Lemma 1.8.

Combining (8) and (6), we have

$$\Pr_A[Q_{i,j} \text{ solves } (X_i, Y_i) \text{ on } A \mid A \in R_i^1] \le 2^{-c_2 m_i / n_i^{c_1} + O(n_{i-1}^{c_3})}$$

for every $1 \le j \le i$.

Step 2: Extending the common prefix. Next, we find an instance $(X_i^{\text{fixed}}, Y_i^{\text{fixed}})$, and let R_i be the restriction of R_i^1 to the set of oracles that agree with $(X^{\text{fixed}}_{\le i}, Y^{\text{fixed}}_{\le i})$. In the following, we show that there exists a choice of $(X_i^{\text{fixed}}, Y_i^{\text{fixed}})$ such that every property is satisfied. The proof is by showing that a random instance (X_i, Y_i) satisfies every property with nonzero probability.

As the other properties are straightforward, we only show that Properties 4 and 6 are satisfied. More precisely, we show that:

Lemma 5.6. *There exists a choice of $(X_i^{fixed}, Y_i^{fixed})$, such that when R_i is set to be the subset of oracles in R_i^1 that agree with $(X_i^{fixed}, Y_i^{fixed})$, we have that*

- ***Largeness:***

$$\Pr_A[A \in R_i \mid A \text{ agrees with } (X^{fixed}_{\le i}, Y^{fixed}_{\le i})] \ge 2^{-n_i^{c_3}}.$$

- ***Refuting robust protocols:*** *For i', j such that $i_0 \le i' \le i$, $1 \le j \le i'$, we have that*
  - *(a)* ***Small protocols:*** *If $2^{i'} \le h(i-1)$, then for any oracle $A \in R_i$, $Q_{i',j}$ does not solve $(X_{i'}, Y_{i'})$ on A.*
  - *(b)* ***Large protocols:*** *If $2^{i'} > h(i-1)$, then*

$$\Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \mid A \in R_i] \le 2^{-c_2 m_{i'} / n_{i'}^{c_1} + n_i^{c_4}}.$$

Note that the property stated for robust protocols is different from Property 6, in that the large protocols are defined to be those that satisfy $2^{i'} > h(i-1)$ instead of $2^{i'} > h(i)$. However, as we will show in Claim 5.9, Lemma 5.6 implies Property 6.

Proof. We first show that largeness is satisfied with nontrivial probability, then show that conditioned on largeness being satisfied, each robust protocol is refuted with high probability.

Largeness. For an instance (X_i, Y_i), let $p_{(X_i,Y_i)}$ denote

$$\Pr_A[A \in R_i^1 \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}}) \wedge A \text{ agrees with } (X_i, Y_i)],$$

i.e., the fraction of R_i^1 among the oracles that agree with $(X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})$ and (X_i, Y_i). In order to satisfy Property 4, we must choose an instance (X_i, Y_i) such that $p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}$. Let

$$p_{\text{large}} = \Pr_{(X_i,Y_i)}[p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}].$$

Intuitively, since $\Pr[A \in R_i^1 \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})]$ is guaranteed to be large, $p_{(X_i,Y_i)}$ should be large on average. By applying a simple Markov bound, we can show that $p_{(X_i,Y_i)}$ is also large with decent probability:

Claim 5.7. *$p_{large} \ge 2^{-n_i^{c_3}}$.*

Proof. Using the notation $p_{(X_i,Y_i)}$, (6) directly translates to

$$\mathbb{E}_{(X_i,Y_i)}[p_{(X_i,Y_i)}] \ge 2^{-O(n_{i-1}^{c_3})}.$$

Since $\Pr_{(X_i,Y_i)}[p_{(X_i,Y_i)} \le 1] = 1$ by definition, we have that

$$\begin{aligned}\mathbb{E}_{(X_i,Y_i)}[p_{(X_i,Y_i)}] &\le p_{\text{large}} \cdot 1 + (1 - p_{\text{large}}) \cdot 2^{-n_i^{c_3}} \\ &\le p_{\text{large}} + 2^{-n_i^{c_3}}.\end{aligned}$$

The claim holds because c_3 is sufficiently large (note that $n_i = 2^i$). ◇

Refuting robust protocols. Next, we apply a union bound over the robust protocols, and show that conditioned on $p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}$, with high probability, every protocol satisfies the property stated in the lemma.

Let $i_0 \le i' \le i$ and $1 \le j \le i'$. In the small case (i.e., $2^{i'} \le h(i-1)$), $Q_{i',j}$ does not solve $(X_{i'}, Y_{i'})$ on any oracle in R_i, and this property is preserved regardless of $(X_i^{\text{fixed}}, Y_i^{\text{fixed}})$. Therefore, we only need to consider the case where $2^{i'} > h(i-1)$. For any $Q_{i',j}$ and (X_i, Y_i), let $q_{i',j,(X_i,Y_i)}$ denote

$$\Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \mid A \in R_i^1 \wedge A \text{ agrees with } (X_i, Y_i)],$$

i.e., the fraction of oracles in R_i that $Q_{i',j}$ solves, if $(X_i^{\text{fixed}}, Y_i^{\text{fixed}})$ is chosen to be (X_i, Y_i). We show that for a random instance (X_i, Y_i), conditioned on $p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}$, $q_{i',j,(X_i,Y_i)}$ is small with high probability. Again, the proof holds by a simple Markov bound, where we use the fact that $Q_{i',j}$ only solves a small fraction of oracles in R_i^1.

Claim 5.8. *Let $i_0 \le i' \le i$, $1 \le j \le i'$, be such that $2^{i'} > h(i-1)$. We have that*

$$\Pr_{(X_i,Y_i)}[q_{i',j,(X_i,Y_i)} > 2^{-c_2 m_{i'}/n_{i'}^{c_1} + n_i^{c_4}} \mid p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}] \le 2^{-i}.$$

Proof. The proof is by investigating the value $\mathbb{E}_{(X_i,Y_i)}[q_{i',j,(X_i,Y_i)} \cdot p_{(X_i,Y_i)}]$, which is the probability that $Q_{i',j}$ solves a random oracle in R_i^1 (conditioned on the common prefix). Indeed, we can show that this is equal to

$$\Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \wedge A \in R_i^1 \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})].$$

Formally, we have that (for simplicity, let B denote the event $[Q_{i',j}$ solves $(X_{i'}, Y_{i'})$ on $A \wedge A \in R_i^1]$)

$$\begin{aligned}
&\mathbb{E}_{(X_i,Y_i)}[q_{i',j,(X_i,Y_i)} \cdot p_{(X_i,Y_i)}] \\
&= 2^{-2n_i m_i} \sum_{(X_i,Y_i)} \left(\Pr[B \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}}) \wedge A \text{ agrees with } (X_i, Y_i)] \right) \\
&= 2^{-2n_i m_i} \sum_{(X_i,Y_i)} \frac{\Pr[B \wedge A \text{ agrees with } (X_i, Y_i) \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})]}{\Pr[A \text{ agrees with } (X_i, Y_i) \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})]} \\
&= 2^{-2n_i m_i} \sum_{(X_i,Y_i)} \frac{\Pr[B \wedge A \text{ agrees with } (X_i, Y_i) \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})]}{2^{-2n_i m_i}} \\
&= \sum_{(X_i,Y_i)} \Pr[B \wedge A \text{ agrees with } (X_i, Y_i) \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})] \\
&= \Pr[B \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})].
\end{aligned}$$

Using (6) and (7), we can upper bound $\Pr[B \mid A$ agrees with $(X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})]$ by

$$\begin{aligned}
&\Pr[B \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})] \\
&= \Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \wedge A \in R_i^1 \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})] \\
&= \Pr[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \mid A \in R_i^1] \cdot \Pr[A \in R_i^1 \mid A \text{ agrees with } (X_{\le i-1}^{\text{fixed}}, Y_{\le i-1}^{\text{fixed}})] \\
&\le 2^{-c_2 m_i / n_i^{c_1} + O(n_{i-1}^{c_3})} \cdot 1. && \text{(by (7))}
\end{aligned}$$

Now assume that the claim does not hold. Then we have that

$$\begin{aligned}
&\mathbb{E}_{(X_i,Y_i)}[q_{i',j,(X_i,Y_i)} \cdot p_{(X_i,Y_i)}] \\
&\ge \mathbb{E}_{(X_i,Y_i)}[q_{i',j,(X_i,Y_i)} \cdot 2^{-n_i^{c_3}} \mid p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}] \cdot \Pr_{(X_i,Y_i)}[p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}] \\
&\ge \Pr_{(X_i,Y_i)}[q_{i',j,(X_i,Y_i)} > 2^{-c_2 m_{i'} / n_{i'}^{c_1} + n_i^{c_4}} \mid p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}] \cdot 2^{-n_i^{c_3}} \cdot 2^{-c_2 m_{i'} / n_{i'}^{c_1} + n_i^{c_4}} \cdot p_{\text{large}} \\
&\ge 2^{-c_2 m_{i'} / n_{i'}^{c_1} + n_i^{c_4} \cdot (1 - o(1))}.
\end{aligned}$$

Since c_4 is sufficiently large, this contradicts (by (7)). ◇

The union bound. By applying an union bound over all $Q_{i',j}$, Claim 5.8 show that, when we randomly select an instance (X_i, Y_i), conditioned on $p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}$, with high probability, $q_{i',j,(X_i,Y_i)}$ is small for every $i_0 \le i' \le i$ and $1 \le j \le i'$. Combining this with Claim 5.7, which shows that there exist instances that satisfy $p_{(X_i,Y_i)} \ge 2^{-n_i^{c_3}}$, we conclude the proof of Lemma 5.6. □

Analysis for step 2. Finally, we show that the rectangle obtained from Lemma 5.6 satisfies Property 6. Note that the property stated in Lemma 5.6 is only different from Property 6 on the requirement for protocols $Q_{i',j}$ with $h(i-1) < 2^{i'} \leq h(i)$. It then remains to show that:

Claim 5.9. *Let R_i be as constructed in Lemma 5.6. For $i_0 \leq i' \leq i$ and $1 \leq j \leq i'$ such that $h(i-1) < 2^{i'} \leq h(i)$, and for any oracle $A \in R_i$, $Q_{i',j}$ does not solve $(X_{i'}, Y_{i'})$ on A.*

Proof. Note that, since $Q_{i',j}$ is simulating an algorithm M that runs in time $2^{i'}$, it only uses the value of the oracle on the instances $(X_{\leq i}, Y_{\leq i})$. Since all oracles in R_i agree on $(X_{\leq i}, Y_{\leq i})$, if $Q_{i',j}$ solves $(X_{i'}, Y_{i'})$ on some oracle in R_i, then it does so on *every* oracle in R_i. Therefore, we only have to show that

$$\Pr_A[Q_{i',j} \text{ solves } (X_{i'}, Y_{i'}) \text{ on } A \mid A \in R_i] \leq 2^{-c_2 m_{i'}/n_{i'}^{c_1} + n_i^{c_4}} < 1.$$

This holds by Claim 5.5. □

References

[Aar05] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005. doi:10.1098/rspa.2005.1546. 3

[Aar06] Scott Aaronson. Oracles are subtle but not malicious. In *Conference on Computational Complexity* (CCC), pages 340–354. IEEE Computer Society, 2006. doi:10.1109/CCC.2006.32. 1, 2, 9

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. doi:10.1017/CBO9780511804090. 10

[AB18] Barış Aydınlıoğlu and Eric Bach. Affine relativization: Unifying the algebrization and relativization barriers. *ACM Trans. Comput. Theory*, 10(1):1:1–1:67, 2018. doi:10.1145/3170704. 4, 9, 19

[AIV92] Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. *Manuscript*, 1992. URL: https://people.eecs.berkeley.edu/~vazirani/pubs/relativizing.pdf. 9

[All90] Eric Allender. Oracles versus proof techniques that do not relativize. In *SIGAL International Symposium on Algorithms*, volume 450 of *Lecture Notes in Computer Science*, pages 39–52. Springer, 1990. doi:10.1007/3-540-52921-7_54. 9

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306. 40

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901. 40

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1(1):2:1–2:54, 2009. doi:10.1145/1490270.1490272. 1, 2, 3, 4, 8, 9, 10, 11, 19, 39, 40

[BCG+96] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996. doi:10.1006/jcss.1996.0032. 9

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.*, 1:3–40, 1991. doi:10.1007/BF01200056. 2, 9, 40

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486. 2, 9

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Conference on Computational Complexity* (CCC), pages 8–12, 1998. doi:10.1109/CCC.1998.694585. 1, 2, 3, 5, 6, 9, 39, 41

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006. doi:10.1137/S0097539705446810. 40, 41

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =?NP question. *SIAM J. Comput.*, 4(4):431–442, 1975. doi:10.1137/0204037. 1, 9

[Cai07] Jin-yi Cai. $\mathsf{S}_2^p \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007. doi:10.1016/j.jcss.2003.07.015. 9

[CCHO05] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp–Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005. doi:10.1016/j.ic.2005.01.002. 9

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *STOC*, pages 1990–1999. ACM, 2024. doi:10.1145/3618260.3649624. 1, 2, 5, 9, 20

[CLL25] Lijie Chen, Jiatu Li, and Jingxun Liang. Maximum circuit lower bounds for exponential-time arthur merlin. In *STOC*, pages 1348–1358. ACM, 2025. doi:10.1145/3717823.3718224. 2, 9

[CLO+23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *FOCS*, pages 1261–1270. IEEE, 2023. doi:10.1109/FOCS57990.2023.00074. 2

[CMMW19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and equivalences between circuit lower bounds and Karp–Lipton theorems. In *Computational Complexity Conference* (CCC), volume 137 of *LIPIcs*, pages 30:1–30:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.30. 9

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459. 40

[For94] Lance Fortnow. The role of relativization in complexity theory. *Bull. EATCS*, 52:229–243, 1994. 9

[GG11] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity* (ECCC), 18:136, 2011. URL: https://eccc.weizmann.ac.il/report/2011/136. 3

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008. doi:10.1017/CBO9780511804106. 10

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Comput. Complex.*, 27(2):245–304, June 2018. doi:10.1007/s00037-018-0166-6. 12

[HHT97] Yenjo Han, Lane A. Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM J. Comput.*, 26(1):59–78, 1997. doi:10.1137/S0097539792240467. 3

[HLR23] Shuichi Hirahara, Zhenjian Lu, and Hanlin Ren. Bounded relativization. In *CCC*, volume 264 of *LIPIcs*, pages 6:1–6:45. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CCC.2023.6. 9

[IKK09] Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Symposium on Theory of Computing* (STOC), pages 695–704. ACM, 2009. doi:10.1145/1536414.1536509. 9

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In *Computational Complexity Conference* (CCC), volume 102 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.7. 9

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Inf. Control.*, 55(1-3):40–56, 1982. doi:10.1016/S0019-9958(82)90382-5. 1, 9

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, volume 185 of *LIPIcs*, pages 44:1–44:18, 2021. doi:10.4230/LIPIcs.ITCS.2021.44. 1

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Symposium on Theory of Computing* (STOC), pages 302–309. ACM, 1980. doi:10.1145/800141.804678. 9

[Kla03] H. Klauck. Rectangle size bounds and threshold covers in communication complexity. In *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.*, pages 118–134, 2003. doi:10.1109/CCC.2003.1214415. 7, 14

[Kor21] Oliver Korten. The hardest explicit construction. In *Symposium on Foundations of Computer Science* (FOCS), pages 433–444. IEEE, 2021. doi:10.1109/FOCS52979.2021.00051. 1

[KW98] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998. doi:10.1137/S0097539795296206. 9

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605. 1, 9, 39

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *STOC*, pages 2000–2007. ACM, 2024. doi:10.1145/3618260.3649615. 1, 2, 9

[LORS24] Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. On the complexity of avoiding heavy elements. In *FOCS*, pages 2403–2412. IEEE, 2024. doi:10.1109/FOCS61266.2024.00140. 5, 20

[Mie09] Thilo Mie. Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries. *Ann. Math. Artif. Intell.*, 56(3-4):313–338, 2009. doi:10.1007/S10472-009-9169-Y. 41

[MVW99] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, volume 1627 of *Lecture Notes in Computer Science*, pages 210–220. Springer, 1999. doi:10.1007/3-540-48686-0_21. 2, 9

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/JCSS.1997.1494. 1

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *FOCS*, pages 640–650. IEEE, 2022. doi:10.1109/FOCS54457.2022.00067. 1

[San09] Rahul Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009. doi:10.1137/070702680. 2, 3, 5, 9, 10

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992. doi:10.1145/146585.146609. 1, 9, 39

[SM02] Larry J. Stockmeyer and Albert R. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *J. ACM*, 49(6):753–784, 2002. doi:10.1145/602220.602223. 9

[Vin05] N. V. Vinodchandran. A note on the circuit complexity of PP. *Theor. Comput. Sci.*, 347(1-2):415–418, 2005. doi:10.1016/j.tcs.2005.07.032. 2, 9

[VW23] Nikhil Vyas and R. Ryan Williams. On oracles and algorithmic methods for proving lower bounds. In *Innovations in Theoretical Computer Science Conference* (ITCS), volume 251 of *LIPIcs*, pages 99:1–99:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.99. 1

[Wil85] Christopher B. Wilson. Relativized circuit complexity. *J. Comput. Syst. Sci.*, 31(2):169–181, 1985. doi:10.1016/0022-0000(85)90040-6. 1

A On the Circuit Lower Bound of Buhrman–Fortnow–Thierauf

We verify that the proof of [BFT98] indeed gives a sub-half-exponential circuit lower bound for $\mathsf{E} \cup \mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}$ that algebrizes. We also interpret this proof as a "win-win" algorithm for Missing-String in the algebraic decision tree model of [AW09].

A.1 Some Preliminaries

In this section, when we relativize space-bounded computations, we take the query tape into account when measuring space complexity. That is, a $\mathsf{SPACE}^A[s(n)]$ machine can only make queries of length at most $s(n)$ to its oracle A. The class PSPACE^A in our context equals to the class $\mathsf{PSPACE}^{A[\mathrm{poly}]}$ in [AW09].

We say that $h(n)$ is a *sub-half-exponential* function, if $h(h(n)^c) \leq 2^n$ for every constant $c \geq 1$. We say that $h(n)$ is *nice*, if there is a deterministic algorithm that on input n, outputs the value of $h(n)$ in $\mathrm{poly}(n)$ time.

We use the notation $\langle P, V\rangle(x)$ to denote the outcome of the interactive protocol with prover P, verifier V, and common input x (which is a random variable). We use IP^A to denote the class of languages that can be decided by an interactive protocol where the prover is computationally unbounded and the verifier is a randomized polynomial-time machine with oracle access to A. Formally, $L \in \mathsf{IP}^A$ if there exists a randomized polynomial-time verifier V with oracle access to A such that for every $x \in \{0,1\}^*$:

- If $x \in L$, then there is an (unbounded) prover P such that $\Pr[\langle P, V\rangle(x) \text{ accepts}] = 1$.
- If $x \notin L$, then for every (unbounded) prover P, $\Pr[\langle P, V\rangle(x) \text{ accepts}] \leq 1/3$.

We need the algebrizing version of $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92].

Theorem A.1 ([AW09])**.** *For every oracle A and every multilinear extension $\widetilde{A}$ of A,*

$$\mathsf{PSPACE}^{A} \subseteq \mathsf{IP}^{\widetilde{A}}.$$

Moreover, the honest IP *prover can be implemented in* $\mathsf{PSPACE}^{\widetilde{A}}$*.*

We remark that Theorem A.2 can be proved by either $\mathsf{IP} = \mathsf{PSPACE}$, $\mathsf{MIP} = \mathsf{NEXP}$ [BFL91], or the PCP theorem [AS98, ALM+98, Din07]. In fact, in Section A.3, we will see another proof using PCP of *Proximity* [BGH+06].

A.2 Reviewing the BFT Proof

Theorem A.2. *Let $h(n)$ be a nice and sub-half-exponential function. For every oracle A and every multilinear extension $\widetilde{A}$ of A,*

$$(\mathsf{E}^{\widetilde{A}} \cup \mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}}) \not\subseteq \mathsf{SIZE}^{A}[h(n)].$$

Proof. Assuming $\mathsf{E}^{\widetilde{A}} \subseteq \mathsf{SIZE}^{A}[h(n)]$, we will prove that $\mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}} \not\subseteq \mathsf{SIZE}^{A}[h(n)]$.

First, let L^{A}_{hard} denote the language whose truth table on input length n is the lexicographically smallest length-2^n truth table that requires A-oracle circuit complexity more than $h(n)$. Clearly, $L_{\mathrm{hard}} \in \mathsf{SPACE}^{A}[\tilde{O}(h(n))] \setminus \mathsf{SIZE}^{A}[h(n)]$. (We stress two details here. First, there is a fixed $\mathsf{SPACE}[\tilde{O}(h(n))]$ oracle machine $M^{(-)}_{\mathrm{hard}}$ independent of the oracle A such that M^{A}_{hard} computes L^{A}_{hard}. Second, M^{A}_{hard} only makes queries of length at most $h(n)$ to the oracle A.) Let $(L'_{\mathrm{hard}})^{A} := \{(x,b) : L^{A}_{\mathrm{hard}}(x) = b\}$, then it follows from Theorem A.1 and a padding argument that $(L'_{\mathrm{hard}})^{A}$ admits an interactive proof $\langle P, V\rangle$ where V runs in randomized $\mathrm{poly}(h(n))$ time and the honest P runs in $2^{\mathrm{poly}(h(n))}$ time; both the verifier and the prover need access to $\widetilde{A}$.

Now we consider the following $\mathrm{Rob}_{h(n)}\text{-}\mathsf{MA}_{\mathsf{E}}^{\widetilde{A}}$ algorithm M. The algorithm receives an input $x \in \{0,1\}^n$ and a proof π from Merlin, where π consists of a bit b, as the purported value of $L_{\mathrm{hard}}(x)$, and an A-oracle circuit P' of 2^n size, treated as an IP prover for L_{hard}. (Note that the input length of P' is $\mathrm{poly}(h(n))$.) Then, M accepts if and only if $\langle P', V\rangle(x,b)$ accepts, i.e., P' convinces V that $L_{\mathrm{hard}}(x) = b$. Note that M corresponds to the pair of verifiers (V_0, V_1) where V_b accepts iff M accepts and the first bit of π is equal to b.

We can see that:

- First, M is an $(\mathsf{MA}_{\mathsf{E}} \cap \mathsf{coMA}_{\mathsf{E}})^{\widetilde{A}}$ algorithm that is robust w.r.t. $h(n)$ on every input length n. To see the robustness of M, fix an arbitrary oracle B and its multilinear extension $\widetilde{B}$, and consider a truth table w that has $h(n)$-size B-oracle circuits. Then, there exists an input $x \in \{0,1\}^n$ such that $L^{B}_{\mathrm{hard}}(x) \neq w_x$. By the soundness of V, it holds that for every circuit P' that Merlin could send,

  $$\Pr[\langle P', V\rangle(x, w_x) \text{ accepts}] \leq 1/3,$$

  Hence $V^{\widetilde{B}}_{w_x}$ rejects x.

- Second, if $\mathsf{E}^{\widetilde{A}} \subseteq \mathsf{SIZE}^{A}[h(n)]$, then $M^{\widetilde{A}}$ computes a function whose circuit complexity is larger than $h(n)$. This is because there is a prover P running in $2^{\mathrm{poly}(h(n))}$ time with oracle access to $\widetilde{A}$ such that for every input $x \in \{0,1\}^n$, $\Pr[\langle P(x,-), V\rangle(x,b) \text{ accepts}] \geq 2/3$. Therefore, under our assumption that $\mathsf{E}^{\widetilde{A}} \subseteq \mathsf{SIZE}^{A}[h(n)]$ and noting that the input length of P is $\mathrm{poly}(h(n))$, P can be implemented by an A-oracle circuit of size $h(\mathrm{poly}(h(n))) \leq 2^n$ and Merlin can send this circuit. It follows that every $x \in \{0,1\}^n$ is easy and $M^{\widetilde{A}}$ computes L^{A}_{hard}. □

A.3 BFT as Algorithms for Missing-String

In this sub-section, we interpret the lower bound in [BFT98] as algebraic query algorithms for Missing-String, more clearly illustrating its win-win analysis.

PCP of proximity. Our algebraic query algorithm makes use of *PCPPs* (probabilistically checkable proofs of proximity) [BGH+06]. Let $L \subseteq \{0,1\}^* \times \{0,1\}^*$ be a *pair language* where the first input is called the *explicit* input and the second input is called the *implicit* input. A PCPP verifier for L is a query algorithm V with the following specification:

- **Inputs:** V takes (x, r) as inputs where x is the explicit input for L and r is the internal randomness of V.
- **Oracles:** V has oracle access to the implicit input y and a PCPP proof π.
- **Completeness:** If $(x, y) \in L$, then there exists an honest PCPP proof π such that

$$\Pr_r[V^{y,\pi}(x, r) \text{ accepts}] = 1.$$

- **Soundness:** If y is δ-far from the set $\{y' : (x, y') \in L\}$, then for every PCPP proof π,

$$\Pr_r[V^{y,\pi}(x, r) \text{ accepts}] \leq 1/3.$$

  Here, $\delta > 0$ is called the ***proximity parameter*** for the PCPP.

We need the following construction of PCPP in [Mie09]:

Theorem A.3. *Let L be a pair language with explicit input length n and implicit input length K, such that $L \in \mathsf{NTIME}[T(n + K)]$ for some non-decreasing function $T(\cdot)$. For every constant $\delta > 0$, L admits a PCPP verifier with proximity parameter δ, randomness complexity $|r| = \log T(n) + O(\log \log T(n))$, query complexity $O(1)$, and verification time $\mathrm{poly}(n, \log K, \log T(n + K))$.*

Moreover, for every $(x, y) \in L$, given a valid witness w for (x, y), a valid PCPP proof π for (x, y) can be computed in deterministic polynomial time.

Algebraic query complexity. In the *algebraic query model*, the algorithm has query access to the multilinear extension of its input. Thanks to the PCPP, the only property of multilinear extensions we need in this sub-section is that it is a *systematic error-correcting code*.

An error-correcting code with ***distance $\delta > 0$*** is a function $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^\ell$ such that for every two different n-bit inputs x_1, x_2, the number of indices $i \in [\ell]$ such that $\mathsf{Enc}(x_1)_i \neq \mathsf{Enc}(x_2)_i$ is at least $\delta \cdot \ell$. The code is ***systematic*** if the first n bits of $\mathsf{Enc}(x)$ are always equal to x itself.

Fix any systematic error-correcting code Enc (such as the one that maps an input to its multilinear extension), an algebraic query algorithm for a problem L is an algorithm that receives $\mathsf{Enc}(x)$ as inputs and produces the output $L(x)$. Note that since Enc is systematic, the algorithm can access its input x in verbatim by looking at the first n bits of $\mathsf{Enc}(x)$.

Now we show that [BFT98] can be interpreted as win-win algorithms for Missing-String in the algebraic query model:

Theorem A.4. *Let $h(n)$ be a nice and sub-half-exponential function, $h'(n) = h(n)^C$ for some large enough universal constant C. Let $\mathcal{X}_1$ be an instance of* Missing-String$(n, h(n))$*, and $\mathcal{X}_2$ be an instance of* Missing-String$(h'(n), 2^n)$*. There are two (query) algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ which both receives $\mathsf{Enc}(\mathcal{X}_1)$ and $\mathsf{Enc}(\mathcal{X}_2)$ as inputs, such that:*

- *$\mathcal{A}_1$ is a robust $\mathsf{MA}^{\mathrm{dt}}$ algorithm with query complexity $\mathrm{poly}(n)$ that outputs a string of length n: No malicious prover (Merlin) can convince the algorithm to output a wrong answer for $\mathcal{X}_1$ except with probability $\leq 1/3$.*
- *$\mathcal{A}_2$ is a deterministic algorithm with query complexity $\mathrm{poly}(h(n))$ that outputs a string of length $h'(n)$.*
- *For every inputs $\mathcal{X}_1, \mathcal{X}_2$, either $\mathcal{A}_1(\mathcal{X}_1, \mathcal{X}_2)$ solves the instance $\mathcal{X}_1$, or $\mathcal{A}_2(\mathcal{X}_1, \mathcal{X}_2)$ solves the instance $\mathcal{X}_2$.*

Proof. Let L be the language consisting of $(a, \mathsf{Enc}(\mathcal{X}))$, where $a \in \{0,1\}^n$ is the explicit input, $\mathcal{X}$ is a Missing-String$(n, h(n))$ instance and $\mathsf{Enc}(\mathcal{X})$ is the implicit input, and $(a, \mathsf{Enc}(\mathcal{X})) \in L$ if and only if a is the lexicographically smallest missing string of $\mathcal{X}$. Let δ be the distance of Enc, and V be the PCPP verifier for L with robustness parameter δ as specified in Theorem A.3.

Let a^* be the lexicographically smallest missing string of $\mathcal{X}_1$, and π be the PCPP proof for $(a^*, \mathsf{Enc}(\mathcal{X}_1)) \in L$. If C is a large enough constant, then $|\pi| \leq h'(n)$. The algorithm $\mathcal{A}_2$ simply outputs π (padded with zeros if $|\pi| < h'(n)$), which can be computed in deterministic $\mathrm{poly}(h(n))$ time. Clearly, if π does not appear in $\mathcal{X}_2$, then $\mathcal{A}_2(\mathcal{X}_1, \mathcal{X}_2)$ solves the instance $\mathcal{X}_2$.

The $\mathsf{MA}^{\mathrm{dt}}$ algorithm $\mathcal{A}_1$ receives a proof from Merlin, which contains a string $a \in \{0,1\}^n$ and an index $j \in [2^n]$. The intended meaning is that a is the lexicographically smallest missing string of $\mathcal{X}_1$, and that $\mathcal{X}_2[j]$ (the j-th string in $\mathcal{X}_2$) is a valid PCPP proof that $(a, \mathsf{Enc}(\mathcal{X}_1)) \in L$. The $\mathsf{MA}^{\mathrm{dt}}$ algorithm runs the PCPP verifier $V^{\mathsf{Enc}(\mathcal{X}_1), \mathcal{X}_2[j]}(a)$ and accepts if and only if the PCPP verifier accepts. Clearly, if π appears in $\mathcal{X}_2$, then $\mathcal{A}_1(\mathcal{X}_1, \mathcal{X}_2)$ solves the instance $\mathcal{X}_1$. Moreover, the soundness of the PCPP verifier implies that no malicious Merlin can convince $\mathcal{A}_1$ to output a wrong answer except with a small probability: if a is not the lexicographically smallest missing string of $\mathcal{X}_1$, then $\mathsf{Enc}(\mathcal{X}_1)$ is δ-far from $\{\mathsf{Enc}(\mathcal{X}) : (a, \mathsf{Enc}(\mathcal{X})) \in L\}$, hence the verifier rejects with high probability. □

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il