Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 142 (2025)

Upper and Lower Bounds for the Linear Ordering Principle*

Edward A. Hirsch† Ilya Volkovich‡

October 21, 2026

Abstract

Korten and Pitassi (FOCS 2024) defined a new[1] complexity class $\mathbf{L}_2^{\mathbf{P}}$ as the polynomial-time Turing closure of the Linear Ordering Principle (a total function extending finding the minimum of an order [CK98] to the case where the order is not linear). They put it between **MA** (Merlin–Arthur protocols) and $\mathbf{S}_2^{\mathbf{P}}$ (the second symmetric level of the polynomial hierarchy).

In this paper we sandwich $\mathbf{L}_2^{\mathbf{P}}$ between $\mathbf{P}^{\mathbf{prMA}}$ and $\mathbf{P}^{\mathbf{prSBP}}$. (The oracles here are promise problems, and **SBP** is the only known class between **MA** and **AM**.) The containment in $\mathbf{P}^{\mathbf{prSBP}}$ is proved via an iterative process that uses a **prSBP** oracle to estimate the average order rank of a subset and find the minimum of a linear order.

Another containment result of this paper is $\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$ (where $\mathbf{O}_2^{\mathbf{P}}$ is the input-oblivious version of $\mathbf{S}_2^{\mathbf{P}}$). These containment results altogether have several byproducts:

- We give an affirmative answer to an open question posed by Chakaravarthy and Roy (Computational Complexity, 2011) whether $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{S}_2^{\mathbf{P}}$, thereby settling the relative standing of the existing (non-oblivious) Karp–Lipton–style collapse results of [CR11] and [Cai07],
- We give an affirmative answer to an open question of Korten and Pitassi whether a Karp–Lipton–style collapse can be proven for $\mathbf{L}_2^{\mathbf{P}}$,
- We show that the Karp–Lipton–style collapse to $\mathbf{P}^{\mathbf{prOMA}}$ is actually better than both known collapses to $\mathbf{P}^{\mathbf{prMA}}$ due to Chakaravarthy and Roy (Computational Complexity, 2011) and to $\mathbf{O}_2^{\mathbf{P}}$ also due to Chakaravarthy and Roy (STACS 2006). Thus we resolve the controversy between previously incomparable Karp–Lipton collapses stemming from these two lines of research.

AI Usage Statement

The original paper was written without any use of AI.

Claude Opus 5.5 contributed to this revision by correcting minor arithmetical errors in Prop. 5.5 and noticing a discrepancy in the previous version of the paper between the formal definition of the **OprMA** class and its use. The affected small subsection concerning the newly defined **OprMA** class is removed from this paper and will be discussed in the next revision of a new preprint ECCC TR26-250, which has enough tools for giving a better definition of this class and proving stronger results about it.

*An extended abstract of this paper appears in the Proceedings of the 43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

†Department of Computer Science, Ariel University, Israel. This research was conducted with the support of the State of Israel, the Ministry of Immigrant Absorption, and the Center for the Absorption of Scientists. Email: `edwardh@ariel.ac.il`

‡Computer Science Department, Boston College, Chestnut Hill, MA. Email: `ilya.volkovich@bc.edu`

[1]Note that this notation had been used in the past [Sch83] for a very different class, which has been apparently forgotten after that.

ISSN 1433-8092

Contents

1 Introduction **3**
- 1.1 Background . . . 3
  - 1.1.1 Classes Based on Symmetric Alternation . . . 3
  - 1.1.2 Classes Based on Merlin-Arthur Protocols . . . 4
- 1.2 Promise Problems as Oracles . . . 4
- 1.3 Our Contribution . . . 5
  - 1.3.1 A New Lower Bound for $\mathbf{L}_2^{\mathbf{P}}$ and the Strongest Non-Input-Oblivious Karp–Lipton Collapse . . . 5
  - 1.3.2 A New Upper Bound for $\mathbf{L}_2^{\mathbf{P}}$ . . . 5
  - 1.3.3 Aggregation of $\mathbf{prO}_2^{\mathbf{P}}$ queries and the Strongest Input-Oblivious Karp–Lipton Collapse . . . 6
- 1.4 Our Techniques . . . 7
  - 1.4.1 Approximate Counting, Set Size Estimation and **SBP** . . . 7
  - 1.4.2 Approximate Counting and the Order Rank Approximation . . . 8
  - 1.4.3 Derandomization in $\mathbf{L}_2^{\mathbf{P}}$ . . . 9
  - 1.4.4 Input-Oblivious Symmetric Alternation . . . 9
- 1.5 Organization of the Paper . . . 10

2 Definitions **10**
- 2.1 Classes of Promise Problems as Oracles . . . 10
- 2.2 Problems Avoid and LOP . . . 12
- 2.3 Complexity Classes . . . 12

3 $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prSBP}}$ **14**
- 3.1 Approximate Counting . . . 14
- 3.2 Estimating the Average Order Rank w.r.t. a Linear Order . . . 15
- 3.3 Finding the Minimum Using a **prSBP** Oracle . . . 17

4 Which Karp–Lipton–style Collapse is Better? **18**
- 4.1 A Karp–Lipton–style Collapse to $\mathbf{P}^{\mathbf{prOMA}}$ . . . 19
- 4.2 Promise Oblivious Merlin–Arthur Protocols are in Promise $\mathbf{O}_2^{\mathbf{P}}$ . . . 20
- 4.3 Merging input-oblivious promise queries . . . 20

5 $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}}$ **22**
- 5.1 The non-input-oblivious setting . . . 22
- 5.2 The Input-Oblivious Setting . . . 23

6 Discussion and Further Research **25**

1 Introduction

The seminal theorem of Richard M. Karp and Richard J. Lipton [KL80] connected non-uniform and uniform complexity by demonstrating a collapse of the Polynomial Hierarchy assuming **NP** has polynomial-size Boolean circuits. This collapse has since been very instrumental in transferring lower bounds against Boolean circuits of *fixed-polynomial*[2] size to smaller classes of the Polynomial Hierarchy. Since then, these results were strengthened in many ways leading to "minimial" complexity classes that have such lower bounds and to which the Polynomial Hierarchy collapses.

1.1 Background

1.1.1 Classes Based on Symmetric Alternation

An important notion in this context is that of *symmetric alternation*. Namely, one of the best collapses was based on the following idea ([Cai07], attributed to Sengupta): if polynomial-size circuits for SAT exist, two provers (defending the answers 'yes' and 'no', respectively) send such circuits to a polynomial-time bounded verifier who can, in turn, use them to verify membership in any language in **PH**. The corresponding class $\mathbf{S_2^P}$ [Can96, RS98] was thus shown to have fixed-polynomial circuit lower bounds. *(In Section 2 we provide formal definitions for all less known classes we use.)*

Indeed, since $\mathbf{NP} \subseteq \mathbf{S_2^P}$, if SAT requires superpolynomial circuits, we are done. Otherwise, the Polynomial Hierarchy, which is known to contain "hard" languages (that is, for every $k \in \mathbb{N}$, $\mathbf{PH} \not\subseteq \mathsf{Size}[n^k]$) by Kannan's theorem [Kan82], collapses to $\mathbf{S_2^P}$ and so do these hard languages. This technique has been known as a *win-win argument* in the literature [Kan82, BCG+96, KW98, Vin05, Cai07, San09, CR11, Vol14, IKV23]. Chen et al. [CMMW19] prove that there is a bidirectional relationship between fixed-polynomial lower bounds and Karp–Lipton–style theorems. In the linear-exponential regime, while the win-win argument can be extended to obtain *superpolynomial* lower bounds for $\mathbf{S_2^E}$ (the linear-exponential version of $\mathbf{S_2^P}$), it falls short of achieving truly *exponential* lower bounds, as it encounters the so-called "half-exponential" barrier (see [MVW99]).

Upon further inspection, one can observe that the presumed polynomial-size circuits for SAT do not actually depend on the input *itself*, but rather on its *length*. Based on this observation, the collapse was deepened to the *input-oblivious* version of $\mathbf{S_2^P}$, called $\mathbf{O_2^P}$ [CR06]. Yet, since $\mathbf{O_2^P}$ is not known and, in fact, *not believed* to contain **NP**, the fixed-polynomial lower bounds do not (immediately) carry over to $\mathbf{O_2^P}$.

This state of affairs remained unchanged for about fifteen years until a significant progress was made when Kleinberg et al. [KKMP21] initiated the study of total functions beyond **TFNP**. While Karp–Lipton's theorem has not been improved, lower bounds against $\mathsf{Size}[n^k]$ were pushed down to $\mathbf{O_2^P}$ [GLV24] and $\mathbf{L_2^P}$ [KP24], a new[1] important class which we describe in more detail below. At the same time, truly exponential lower bounds were established for $\mathbf{S_2^E}$ [Li24] (as it turns out, $\mathbf{S_2^E} = \mathbf{O_2^E}$ [GLV24]) and $\mathbf{L_2^E}$ [KP24].

An important feature of these new results was that they were based on reducing finding a hard function to a total search problem. Namely, the works of Korten [Kor22] and Li [Li24] reduced the question to the so-called *Range Avoidance* problem: given a function $f\colon \{0,1\}^n \to \{0,1\}^m$ with $m > n$, represented by a Boolean circuit, find a point outside its image. This problem is known in the bounded arithmetic community as the *dual Weak Pigeonhole Principle* (dWPHP) and has notable implications in proof complexity (see e.g. [PWW88, Jeř04] and the survey [Kra25]). In

[2] That is, for any $k \in \mathbb{N}$, the class contains a language that cannot be computed by Boolean circuits of size n^k, i.e. a language outside of $\mathsf{Size}[n^k]$.

[CHR24, Li24], Range Avoidance has been reduced to symmetric alternation. Subsequently, Korten and Pitassi [KP24] reduced Range Avoidance to the *Linear Ordering Principle*: given an implicitly described ordering relation, either find the smallest element or report a breach of the linear order axioms (for the case of a linear order this problem is known as MIN in the bounded arithmetic community [CK98]). A polynomial-time Turing closure of this principle gives rise to a new class $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{S}_2^{\mathbf{P}}$: a version of $\mathbf{S}_2^{\mathbf{P}}$ where the two provers provide points of a polynomial-time verifiable linear order on binary strings of a certain length (each point starting with the corresponding answer 0 or 1), and the prover that provides the smaller element wins.

1.1.2 Classes Based on Merlin-Arthur Protocols

In a parallel line of research, the same questions were considered for classes based on Merlin–Arthur proofs: Santhanam [San09] has shown fixed-polynomial lower bounds for *promise problems* possessing such proofs (i.e. the class **prMA**). In [CR11], Chakaravarthy and Roy have shown a Karp–Lipton–style collapse and thus fixed-polynomial size lower bounds for the class $\mathbf{P}^{\mathbf{prMA}}$. In particular, they presented a new upper bound for $\mathbf{S}_2^{\mathbf{P}}$ by showing that $\mathbf{S}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prAM}}$. Nonetheless, the relationship between $\mathbf{P}^{\mathbf{prMA}}$ and the classes of symmetric alternation (including $\mathbf{S}_2^{\mathbf{P}}$, $\mathbf{O}_2^{\mathbf{P}}$, and then-unknown $\mathbf{L}_2^{\mathbf{P}}$) remained open.

Combining their upper bound for $\mathbf{S}_2^{\mathbf{P}}$ with a result of [AKSS95], that $\mathbf{NP} \subseteq \mathbf{P/poly}$ implies an "internal collapse" $\mathbf{MA} = \mathbf{AM}$[3], [CR11] concluded that the Polynomial Hierarchy collapses all the way to $\mathbf{P}^{\mathbf{prMA}}$. Subsequently, by applying the win-win argument, they obtained fixed-polynomial bounds for $\mathbf{P}^{\mathbf{prMA}}$, which (unlike **prMA**) is a class of languages. It is to be noted though that since **prMA** is not a class of languages — while $\mathbf{P}^{\mathbf{prMA}}$ is, there is no *immediate* way to carry any lower bound against **prMA** over to $\mathbf{P}^{\mathbf{prMA}}$: it is not clear how to leverage (even) Turing reductions to construct a specific language consistent with a given promise problem.

Babai, Fortnow, and Lund [BFL91] prove that if $\mathbf{EXP} \subseteq \mathbf{P/poly}$, then $\mathbf{EXP} = \mathbf{MA}$. Although this is a much larger class, the proof has the advantage that it does not relativize. More collapses in the exponential regime have been proved since then [IKW02, BH92], and the win-win argument yields superpolynomial lower bounds for some of them: $\mathbf{MAEXP} \not\subseteq \mathbf{P/poly}$ [BFT98].

1.2 Promise Problems as Oracles

An important note is due on the use of a promise problem as an oracle, because the literature contains several different notions for this. The collapse result of Chakaravarthy and Roy that we use follows the *loose oracle access* mode adopted in [CR11]. Namely, oracle queries outside of the promise set are allowed and no particular behaviour of the computational model defining the promise class is expected on such queries. At the same time, the answer of the (base) machine using this oracle must be correct *irrespective* of the oracle's answers to such queries and no assumption is made on the internal consistency of the answers outside of the promise set.

For deterministic polynomial-time oracle machines this approach is equivalent to querying any language *consistent* with the promise problem, that is, a language that contains all the "yes" instances and does not contain the "no" instances of the promise problem. One can also extend it to complexity classes: a class $\mathcal{C}$ of languages is consistent with a class $\mathcal{D}$ of promise problems if for every problem $\Pi \in \mathcal{D}$ there is a language $L \in \mathcal{C}$ consistent with Π. The equivalence between the approaches follows from the works of [GS88, BF99], where the latter approach has been adopted. Nonetheless, we include a formal proof of this equivalence in Section 2.1 for the sake of completeness and for further reference.

[3] The internal collapse goes through for the promise versions of the classes as well.

1.3 Our Contribution

In this paper we prove the inclusions $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prSBP}}$ and $\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$, which not only give new upper and lower bounds for $\mathbf{L}_2^{\mathbf{P}}$, but also demonstrate that the Karp–Lipton–collapse to $\mathbf{P}^{\mathbf{prOMA}}$ is currently the best one both for symmetric–alternation–based and Merlin–Arthur–based classes of languages.

1.3.1 A New Lower Bound for $\mathbf{L}_2^{\mathbf{P}}$ and the Strongest Non-Input-Oblivious Karp–Lipton Collapse

Two open questions regarding symmetric alternation have been stated explicitly:

- whether $\mathbf{P}^{\mathbf{prMA}}$ is contained in $\mathbf{S}_2^{\mathbf{P}}$ [CR11] (note that for these two classes Karp–Lipton style theorems have been proved by [CR11] and by Samik Sengupta [unpublished], respectively),
- whether a Karp–Lipton–style theorem holds for $\mathbf{L}_2^{\mathbf{P}}$ [KP24].

In this paper we resolve both these questions affirmatively by showing the following containment. (Recall that $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{S}_2^{\mathbf{P}}$.)

Theorem 1. $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}}$.

Combining this theorem with a result of Chakaravarthy and Roy [CR11] that $\mathbf{NP} \subseteq \mathbf{P/poly}$ implies the collapse $\mathbf{PH} = \mathbf{P}^{\mathbf{prMA}}$, we obtain a Karp–Lipton–style collapse theorem for $\mathbf{L}_2^{\mathbf{P}}$, thus resolving an open question posed in [KP24].

Corollary 2. *If* $\mathbf{NP} \subseteq \mathbf{P/poly}$, *then* $\mathbf{PH} = \mathbf{L}_2^{\mathbf{P}} = \mathbf{P}^{\mathbf{prMA}}$.

Together with the result of [Cai07], it lines up all known non-input-oblivious classes for which a Karp–Lipton–style collapse has been shown:

$$\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{S}_2^{\mathbf{P}} \subseteq \mathbf{ZPP}^{\mathbf{NP}} \subseteq \mathbf{\Sigma}_2^{\mathbf{P}}.$$

1.3.2 A New Upper Bound for $\mathbf{L}_2^{\mathbf{P}}$

Another important result of this work is a new upper bound on $\mathbf{L}_2^{\mathbf{P}}$: we prove that $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prSBP}}$. The best known upper bound prior to our result followed from [KP24, CR11]: $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{S}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prAM}}$.

Theorem 3. $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prSBP}}$.

In summary, our two new inclusions (Theorems 1 and 3) yield the "normal" (non-input-oblivious) part of Figure 1.

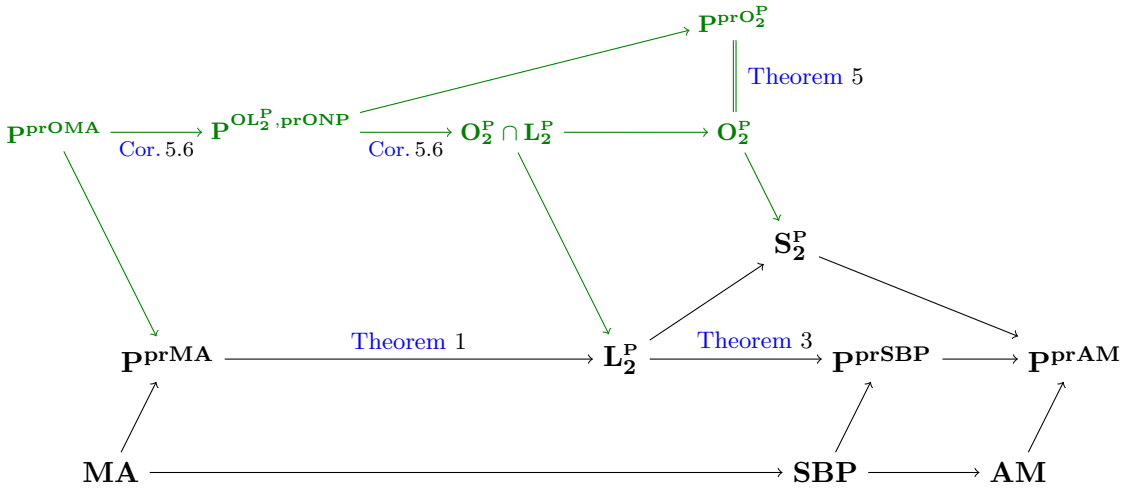

Figure 1: Containments of classes based on Merlin–Arthur protocols and on symmetric alternation.

Switching to the linear-exponential regime, in [KP24], Korten and Pitassi have shown that $\mathbf{L}_2^{\mathbf{E}}$ — the exponential version of $\mathbf{L}_2^{\mathbf{P}}$ — contains a language of circuit complexity $2^n/n$. By translation, our upper bound scales as $\mathbf{L}_2^{\mathbf{E}} \subseteq \mathbf{E}^{\mathbf{prSBP}}$ [4]. As a corollary we obtain a new circuit lower bound for the class $\mathbf{E}^{\mathbf{prSBP}}$. To the best of our knowledge, the strongest previously established bound for this class was "half-exponential" that followed from the bound on **MAEXP** [MVW99].

Corollary 4. $\mathbf{E}^{\mathbf{prSBP}}$ *contains a language of circuit complexity* $2^n/n$.

It is to be noted that Corollary 4 could be viewed as an *unconditional* version of a result of Aydinlioğlu et al. [AGHK11] as it recovers and strengthens their conclusion. In particular, [AGHK11] have shown the following[5]: if $\mathbf{P}^{\mathbf{NP}}$ is consistent with **prAM** or even **prSBP**, then $\mathbf{E}^{\mathbf{NP}}$ contains a language of circuit complexity $2^n/n$. Indeed, given the premises we obtain that $\mathbf{E}^{\mathbf{prSBP}} \subseteq \mathbf{E}^{\mathbf{P}^{\mathbf{NP}}} = \mathbf{E}^{\mathbf{NP}}$ from which the claim follows directly by Corollary 4.

1.3.3 Aggregation of $\mathbf{prO}_2^{\mathbf{P}}$ queries and the Strongest Input-Oblivious Karp–Lipton Collapse

Theorem 1 shows that $\mathbf{P}^{\mathbf{prMA}}$ is currently the smallest non-input-oblivious class for which a Karp–Lipton–style collapse is known. On the other hand, such a collapse was also shown for $\mathbf{O}_2^{\mathbf{P}}$ [CR06], which is input-oblivious. However, since the precise relationship between $\mathbf{O}_2^{\mathbf{P}}$ and $\mathbf{P}^{\mathbf{prMA}}$ remains unknown, one may ask: what is the strongest Karp–Lipton–style collapse? Our next result assists in navigating this question.

Theorem 5. $\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$.

We note that the "non-promise" version of this inclusion, i.e. $\mathbf{P}^{\mathbf{O}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$, was already established in [CR06]. However, this result does not carry over to the promise case. A similar phenomenon arises in the non-input-oblivious analogue of this question: while we know that $\mathbf{P}^{\mathbf{S}_2^{\mathbf{P}}} \subseteq \mathbf{S}_2^{\mathbf{P}}$ [CR06], it still remains open whether $\mathbf{P}^{\mathbf{prS}_2^{\mathbf{P}}} \subseteq \mathbf{S}_2^{\mathbf{P}}$.

[4] As it turns out, this class is also equal $\mathbf{E}^{\mathbf{prBPP}_{\mathbf{path}}}$. See Section 1.4.1 for more details.

[5] The corresponding claims in [AGHK11] were stated using slightly different terminology.

With this tool in hand, we can identify and show the strongest Karp–Lipton–style collapse that currently known. Namely, the collapse can be extended to $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{P}^{\mathbf{prMA}}$, where **prOMA** is the input-oblivious version of **prMA** and therefore is contained in **prMA**. At the same time, Theorem 5 allows us to show that $\mathbf{P}^{\mathbf{prOMA}}$ is also contained in $\mathbf{O}_2^{\mathbf{P}}$, thus making $\mathbf{P}^{\mathbf{prOMA}}$ smaller than both $\mathbf{P}^{\mathbf{prMA}}$ and $\mathbf{O}_2^{\mathbf{P}}$!

Theorem 5 also allows us to refine the chain of containments between $\mathbf{P}^{\mathbf{prOMA}}$ and $\mathbf{O}_2^{\mathbf{P}}$ by introducing a newly defined class, $\mathbf{OL}_2^{\mathbf{P}}$, an input-oblivious analogue of $\mathbf{L}_2^{\mathbf{P}}$ (see Figure 1). Note that this chain proceeds via $\mathbf{P}^{\mathbf{OL}_2^{\mathbf{P}},\mathbf{prONP}}$ rather than $\mathbf{OL}_2^{\mathbf{P}}$ itself, since $\mathbf{OL}_2^{\mathbf{P}}$ is not guaranteed either to share the convenient closure properties of $\mathbf{L}_2^{\mathbf{P}}$ or to contain **ONP**. Still we prove that

$$\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{P}^{\mathbf{OL}_2^{\mathbf{P}},\mathbf{prONP}} \subseteq \mathbf{O}_2^{\mathbf{P}} \cap \mathbf{L}_2^{\mathbf{P}}.$$

That is, $\mathbf{P}^{\mathbf{OL}_2^{\mathbf{P}},\mathbf{prONP}}$ can serve as an input-oblivious analogue of $\mathbf{L}_2^{\mathbf{P}}$ ($= \mathbf{P}^{\mathbf{L}_2^{\mathbf{P}},\mathbf{NP}}$ [KP24]).

1.4 Our Techniques

1.4.1 Approximate Counting, Set Size Estimation and SBP

Computing the number of accepting paths of a given non-deterministic Turing machine is a fundamental problem captured by the "counting" class **#P**. Yet, this class appears to be too powerful since, by Toda's Theorem [Tod91], even a single query to it suffices to decide any language in the polynomial hierarchy, $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}[1]}$! Given that, it is natural to explore approximations. To this end, one can consider the problem of *Approximate Counting* (ApproxCount for short) which refers to the task of *approximating* the number of accepting paths (within a constant factor). Equivalently, this problem can be framed as approximating the size of a set S represented as the set of satisfying assignments of a Boolean circuit C. Previously, it was shown that this task could be carried out by a randomized algorithm using an **NP** oracle ($\mathbf{FBPP}^{\mathbf{NP}}$) [Sto85, JVV86] and by a deterministic algorithm using a **prAM** oracle ($\mathbf{FP}^{\mathbf{prAM}}$) [Sip83, GS86]. Shaltiel and Umans [SU06] show how to accomplish this task in $\mathbf{FP}^{\mathbf{NP}}$, yet under a derandomization assumption. We note that all of these algorithms can be implemented using parallel (i.e. non-adaptive) oracle queries. That is, in $\mathbf{FBPP}_{\|}^{\mathbf{NP}}$, $\mathbf{FP}_{\|}^{\mathbf{prAM}}$ and $\mathbf{FP}_{\|}^{\mathbf{NP}}$, respectively. In [Jeř07], Jeřábek studied approximate counting in the context of bounded arithmetic and reduced to it many problems in various complexity classes. Approximate counting has also recently attracted considerable attention in the quantum literature [OS18, AKKT20, AR20, MAD25].

The decision version of the problem is to distinguish between two constant-factor *estimates* of the set size. For concreteness, consider the following problem called *set-size estimation* (or SSE for short): Given a set S (via a Boolean circuit C) and an integer m with the *promise* that either $|S| \geq m$ or $|S| \leq m/2$, our goal is to decide which case holds. Interestingly, this problem is complete for the class $\mathbf{SBP}$[6] introduced in [BGM06] by Böhler et al. as a relaxation of the class **BPP** to the case when the acceptance probability is not required to be bounded away from 0. This relaxation, as was shown in [BGM06], yields additional power: $\mathbf{MA} \subseteq \mathbf{SBP} \subseteq \mathbf{AM}$. The class **SBP**, which stands for *small bounded-error polynomial-time*, thus sits strictly between the two fundamental classes based on Arthur–Merlin protocols, yet its definition is not based on these protocols. Moreover, **SBP** remains the only known natural class that lies between **MA** and **AM**.

[6] Strictly speaking, the problem is complete for **prSBP**, the corresponding class of promise problems, and is therefore hard for **SBP**.

In terms of upper bounds, in a seminal paper [GS86], Goldwasser and Sipser have exhibited an Arthur–Merlin protocol not only for this problem, but also for the case when the set S is represented by a *non-deterministic* circuit[7]! This more general version of the problem (WSSE, where the set S is given via a non-deterministic circuit), is complete for the class **prAM**. (See Definition 3.1 for the formal definition of the problems; in fact, the factor-of-two gap in the estimates is arbitrary and can be replaced by any positive constant.) In fact, Goldwasser–Sipser's protocol proves the containment both for languages ($\mathbf{SBP} \subseteq \mathbf{AM}$) and for promise problems ($\mathbf{prSBP} \subseteq \mathbf{prAM}$). At the same time, it is important to highlight the distinction between the two versions of the problem — i.e. for the "standard" (SSE) vs non-deterministic (WSSE) Boolean circuits — which appears to be (at the very least) non-trivial. Notably, the work of [BGM06] established an oracle separation between **SBP** and **AM**.

On a similar note, by combining some of the previous techniques, we observe that ApproxCount can be carried out in $\mathbf{FP}^{\mathbf{prSBP}}$ rather than $\mathbf{FP}^{\mathbf{prAM}}$, and, in fact, even in $\mathbf{FP}_{\parallel}^{\mathbf{prSBP}}$. Given this observation, it is natural to study the computational power of $\mathbf{P}^{\textsc{ApproxCount}}$, that is, deterministic algorithms with oracle access to ApproxCount. Indeed, an immediate corollary of the above is that $\mathbf{P}^{\textsc{ApproxCount}} = \mathbf{P}^{\mathbf{prSBP}}$ and $\mathbf{P}_{\parallel}^{\textsc{ApproxCount}} = \mathbf{P}_{\parallel}^{\mathbf{prSBP}}$. At the same, O'Donnell and Say [OS18] previously showed that $\mathbf{P}_{\parallel}^{\textsc{ApproxCount}} = \mathbf{BPP}_{\mathbf{path}}$, a complexity class defined earlier by Han et al. [HHT97]. One can think of $\mathbf{BPP}_{\mathbf{path}}$ as a version of **BPP** in which different computational paths (of the same probability) may have different lengths. Incidentally, it was established in [BGM06] that $\mathbf{BPP}_{\mathbf{path}}$ can be obtained from **BPP** via the so-called "postselection" and that $\mathbf{SBP} \subseteq \mathbf{BPP}_{\mathbf{path}}$ (and resp. $\mathbf{prSBP} \subseteq \mathbf{prBPP}_{\mathbf{path}}$). Putting all together, one arrives at the following three clusters of complexity classes associated with approximate counting:

$$\mathbf{SBP} \subseteq \mathbf{BPP}_{\mathbf{path}} = \mathbf{P}_{\parallel}^{\textsc{ApproxCount}} = \mathbf{P}_{\parallel}^{\mathbf{prSBP}} \subseteq \mathbf{P}^{\textsc{ApproxCount}} = \mathbf{P}^{\mathbf{prSBP}} = \mathbf{P}^{\mathbf{prBPP}_{\mathbf{path}}},$$

where both inclusions are believed to be strict.

1.4.2 Approximate Counting and the Order Rank Approximation

The upper bound $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prSBP}}$ is obtained by developing a process that, given an arbitrary element in a linearly ordered set, rapidly converges to the set's minimum.

Approximate counting using a prSBP oracle. We show how to deterministically approximate the number of satisfying assignments of a Boolean circuit, given oracle access to **prSBP** (i.e. in $\mathbf{FP}^{\mathbf{prSBP}}$), using parallel queries. Our algorithm is based on **SBP** amplification that was used in [BGM06, Vol20]. A crucial observation is that, as we need a multiplicative approximation (up to the factor $1+\varepsilon$), it suffices to place the desired number between two consecutive powers of two; the correct place then could be found by either querying a **prSBP** oracle $O(n/\varepsilon)$ times in parallel or (using binary search) $O(\log_2(n/\varepsilon))$ times sequentially. This result could be on independent interest. See Lemma 3.4 for the formal statement.

Approximating the order rank w.r.t. a linear order. The *order rank* of an element α of a linearly ordered set U is the number of elements in this set that are strictly less than α (in particular, α is the minimum if and only if $\mathrm{rank}(\alpha) = 0$). We can extend this definition to non-empty subsets $S \subseteq U$, where $\mathrm{rank}(S)$ is the average order rank of elements in S.

[7] A non-deterministic circuit $C(x, w)$ accepts x if there exists a witness w for which $C(x, w) = 1$.

We reduce the problem of approximately comparing the average order ranks of two sets to approximate counting. To see how, consider a strict linear order $\lhd_E$ implicitly defined on $U = \{0,1\}^n$ using a Boolean circuit E, and observe that for a non-empty subset $S \subseteq U$, the average order rank of S is exactly the size of the set of pairs $\{(\upsilon, \alpha) \in U \times S \mid \upsilon \lhd_E \alpha\}$ divided by the size of S. Hence, this task can be carried out using a **prSBP** oracle.

An upper bound for the Linear Ordering Principle. As was mentioned, we develop a process that, given an arbitrary element in a linearly ordered set $U = \{0,1\}^n$, rapidly converges to the set's minimum.

Given an element $\alpha \in U$, we first define the set S as the set of all the elements less or equal to α. Formally, $S := \{x \mid x \leq \alpha\}$. Observe that $\mathrm{rank}(S) = \mathrm{rank}(\alpha)/2$. We then iteratively partition S into two disjoint sets, starting from $i = 1$:

$$S_0 = \{x \in S \mid x_i = 0\} \text{ and } S_1 = \{x \in S \mid x_i = 1\}.$$

By averaging argument, $\min\{\mathrm{rank}(S_0), \mathrm{rank}(S_1)\} \leq \mathrm{rank}(S)$. We then take S to be the subset (S_0 or S_1) with the smaller order rank and continue to the next value of i. That is, we fix the bits of the elements of S one coordinate at a time. Therefore, once $i = n$, our "final" set S contains *exactly* one element β and thus at that point $\mathrm{rank}(S) = \mathrm{rank}(\beta)$. On the other hand, as the order rank of the "initial" S was $\mathrm{rank}(\alpha)/2$ and the overall order rank could only decrease, we obtain that $\mathrm{rank}(\beta) \leq \mathrm{rank}(\alpha)/2$. We can then invoke the same procedure this time with β as its input. As the there are 2^n elements in U, this process will converge to the set's minimum after invoking the procedure at most n times, given *any* initial element.

The algorithm described above requires computing (or at least comparing) the average order ranks of two sets. Our analysis demonstrates that a procedure for approximate comparison, developed before, is sufficient for the implementation of this idea (though the factor at each step will be a little bit less than 2).

1.4.3 Derandomization in $\mathbf{L}_2^{\mathbf{P}}$

In [KP24], Korten and Pitassi show that $\mathbf{MA} \subseteq \mathbf{L}_2^{\mathbf{P}}$. The inclusion $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}}$ essentially follows their argument with the additional observation that since $\mathbf{L}_2^{\mathbf{P}}$ is a syntactic class, not only it *contains* **MA** (as was shown) but it is also *consistent* with **prMA**. Thus one can first construct a pseudorandom generator using an $\mathbf{L}_2^{\mathbf{P}}$ oracle [Kor22, KP24] and then leverage it to fully derandomize the **prMA** oracle not just in **prNP**, but actually in $\mathbf{NP} \subseteq \mathbf{L}_2^{\mathbf{P}}$! Therefore, $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{P}^{\mathbf{L}_2^{\mathbf{P}}} = \mathbf{L}_2^{\mathbf{P}}$.

We also observe that since the pseudorandom generator depends only on the input length (and not the input itself), derandomization also helps settling the relations between input-oblivious classes to a certain extent. The main difference is that unlike their non-oblivious counterparts, they do not posses all the desired properties w.r.t. Turing closure and natural containments. However, Theorem 5 (its technique is described below) eventually helps us building the chain from $\mathbf{P}^{\mathbf{prOMA}}$ to $\mathbf{O}_2^{\mathbf{P}}$ in two different ways: both directly and through derandomization and intermediate classes using $\mathbf{OL}_2^{\mathbf{P}}$.

1.4.4 Input-Oblivious Symmetric Alternation

A Karp–Lipton–style collapse to $\mathbf{P}^{\mathbf{prOMA}}$ follows from [CR11] by combining several previously known techniques. However, is this collapse stronger than the known collapse to $\mathbf{O}_2^{\mathbf{P}}$ [CR06]? The inclusion $\mathbf{prOMA} \subseteq \mathbf{prO}_2^{\mathbf{P}}$ can be transferred from a somewhat similar statement that was proven

in [CR06]; however, in order to prove $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{O_2^P}$ we need also the inclusion $\mathbf{P}^{\mathbf{prO_2^P}} \subseteq \mathbf{O_2^P}$, which seems novel. The main idea is that the two provers corresponding to the oracle give their input-oblivious certificates prior to the whole computation, and the verification algorithm performs a cross-check not only between the certificates of **different** provers but also between the certificates of the **same** prover, which allows us to simulate all oracle queries to $\mathbf{prO_2^P}$ in a single $\mathbf{O_2^P}$ algorithm. Indeed, our approach is made possible by the input-oblivious nature of the computational model: while the oracle queries may be adaptive and not known in advance (due to potential queries outside of the promise set), the certificates are universal for the whole computation and nothing else is required.

1.5 Organization of the Paper

The paper is organized as follows: In Section 2 we give the necessary definitions and also discuss oracle access modes in a more formal way. Section 3 contains the proof of Theorem 3 – a new upper bound on $\mathbf{L_2^P}$. Section 4 contains the proof of Theorem 5 which implies that collapse to $\mathbf{P}^{\mathbf{prOMA}}$ subsumes both collapses to $\mathbf{P}^{\mathbf{prMA}}$ and to $\mathbf{O_2^P}$. Section 5 contains the proof of Theorem 1 answering the open questions of Korten and Pitassi [KP24] and Chakaravarthy and Roy [CR11]. We also include a version of this theorem for input-oblivious classes. We discuss multiple research directions in Section 6.

2 Definitions

2.1 Classes of Promise Problems as Oracles

Before defining complexity classes, we first clarify what we mean by oracle access to classes of promise problems. A *promise problem* is a relaxation of (the decision problem for) a language.

Definition 2.1 (promise problem)**.** $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ *is a* promise problem *if* $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \emptyset$.

Similarly to [CR11], when an oracle is described as a promise problem, we use *loose access* to the oracle. The outer Turing machine is allowed to make queries outside of the promise set, and the oracle does not need to conform to the definition of the promise oracle class for such queries. However, the outer Turing machine must return the correct answer *irrespective* of oracle's behavior for queries outside of the promise set in particular, the oracle does not need to be consistent in its answers to the same query. Let us now formulate it in a more precisely to avoid misunderstanding:

Loose Oracle Access vs Access Through a Consistent Language One must be very careful when arguing about oracle access to promise classes. Several modes of access have been considered in the literature including guarded access [LR94], loose access (e.g., [CR11, HS15]), and access through a consistent language [GS88, BF99].

In this paper we use the same mode of access as in [CR11]. Fortunately, when the underlying computational model is a polynomial-time Turing machine [with a specific polynomial-time alarm clock], loose access is equivalent to access through a consistent language. In order to make this connection very clear, below we define both of them rigorously and show the equivalence. Note that for other computational devices, such as those corresponding to $\mathbf{O_2^{P\bullet}}$ or $\mathbf{S_2^{P\bullet}}$, the connection between the modes is much less clear.

Access through a consistent language. The following definition is from [GS88].

Definition 2.2 (consistency)**.** *Consider a promise problem* $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$*, where* $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \emptyset$*. We say that a language* O *is* consistent *with* Π*, if* $\Pi_{\text{YES}} \subseteq O$ *and* $\Pi_{\text{NO}} \subseteq \overline{O}$*. Let us denote it as* $O \subsetneq \Pi$*. Note that the containment of* O *outside of* $\Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ *can be arbitrary.*

In one line of the literature [GS88, BF99], when an oracle is described as a promise problem, the following mode of access is used (*access through a consistent language*): the outer Turing machine queries a language L consistent with this promise problem. In particular, it is allowed to make queries outside of the promise set. However, the outer Turing machine must return a correct answer for every possible choice of L.

Let us formalize it in the simplest case of $\mathbf{P}^{\Pi}$:

Definition 2.3. *A language* L *is in* $\mathbf{P}^{\Pi}$ *according to the consistent language mode, if there is a deterministic oracle Turing machine* $M^{\bullet}$ *with polynomial-time alarm clock stopping it in time* $p(n)$ *such that for every* $O \subsetneq \Pi$ *and for every input* x*,* $M^{O}(x) = L(x)$*.*

One can observe that for every such language O, $\mathbf{P}^{\Pi} \subseteq \mathbf{P}^{O}$, therefore the following holds as well, and in fact can be considered as an alternative definition.

Definition 2.4 (promise problems as oracles)**.** $\mathbf{P}^{\Pi} := \bigcap_{\substack{\Pi_{\text{YES}} \subseteq O \\ \Pi_{\text{NO}} \subseteq \overline{O}}} \mathbf{P}^{O}$.

For more details and discussion see e.g. [GS88, BF99].

Loose access. In the loose access mode, one considers oracle as a black box (physical device) that is only guaranteed to work correctly on the promise set. Outside this set it can produce any answer, in particular, it can give different answers to the same query, either during one deterministic computation, or for different witnesses if the computational model uses them, or for different inputs.

Let us define it rigorously in the simplest case of $\mathbf{P}^{\Pi}$:

Definition 2.5. *A language* L *is in* $\mathbf{P}^{\Pi}$ *according to the loose access mode, if there is a deterministic oracle Turing machine* $M^{\bullet}$ *with polynomial-time alarm clock stopping it in time* $p(n)$ *such that for every input* x $M^{\bullet}(x)$ *produces the answer* $L(x)$ *if the oracle answers "yes" on queries in* Π_Y *and answers "no" on queries in* Π_N*. (No hypothesis is made on its behaviour outside* $\Pi_Y \cup \Pi_N$*, in particular, its answers may be inconsistent for the same query.)*

Equivalence for $\mathbf{P}^{\Pi}$. Fortunately, it is very easy to verify the equivalence of these two definitions in the case of $\mathbf{P}^{\Pi}$.

Proposition 2.6. *Let* Π *be a promise problem. A language* L *belongs to* $\mathbf{P}^{\Pi}$ *according to the loose access mode if and only if it belongs to* $\mathbf{P}^{\Pi}$ *according to access through a consistent language.*

Proof. The "only if" direction is trivial since a consistent language is a particular case of a black box. For the "if" direction, consider $M^{\bullet}$ from the definition of access through a consistent language. To make it tolerant to the loose access mode, maintain a table of oracle answers, and if $M^{\bullet}$ attempts to repeat the same query, use the answer from the table instead. The definition of the machine will not change if its oracle is a language, thus it will still produce the same answer. On the other hand, its behaviour on input x when given a *blackbox* oracle is the same as its behaviour when given the following *language* L_x as an oracle: $L_x = \Pi_Y \cup Y_x$, where Y_x is the set of all queries for which the blackbox oracle gave the answer "yes" when asked for the first time. Since L_x is consistent with Π, $M^{L_x}(x) = L(x)$ according to the definition of access through a consistent language. □

2.2 Problems Avoid and LOP

Definition 2.7 (Avoid, Range Avoidance, [KKMP21, Kor22])**.**
Avoid *is the following total search problem.*
Input: *circuit C with n inputs and $m > n$ outputs.*
Output: $y \in \{0,1\}^m \setminus \operatorname{Im} C$.

Remark 2.8. *This problem, in a slightly different formulation, is known in the bounded arithmetic community as the* dual Weak Pigeonhole Principle *(d*WPHP*). See e.g. [Kra25] for more details.*

Korten [Kor22] have shown that for a stretch of $n+1$ this problem is equivalent to a stretch of $O(n)$ (and, of course, vice versa) under $\mathbf{P}^{\mathbf{NP}}$ reductions.

The following definition is due to Korten and Pitassi [KP24]. It extends the search problem MIN introduced in [CK98], to the setting where the input relation is not necessarily a linear order.

Definition 2.9 (LOP, Linear Ordering Principle, [KP24])**.**
LOP *is the following total search problem.*
Input: *ordering relation $<_E$ given as a Boolean circuit E with $2n$ inputs.*
Output: *either the minimum for $<_E$ (that is, x such that $\forall y \in \{0,1\}^n \setminus \{x\}$ $x <_E y$) or a counterexample, if $<_E$ is not a strict linear order. A counterexample is either a pair satisfying $x <_E y <_E x$ or a triple satisfying $x <_E y <_E z <_E x$.*

2.3 Complexity Classes

The following two definitions have been suggested by Korten and Pitassi who also proved their equivalence [KP24].

Definition 2.10 ($\mathbf{L}_2^{\mathbf{P}}$ via reductions)**.** *A language $L \in \mathbf{L}_2^{\mathbf{P}}$ if it can be reduced to* LOP *using a $\mathbf{P}^{\mathbf{NP}}$-Turing reduction. (Polynomial-time Turing reductions and polynomial-time many-one reductions have the same effect, as proved in [KP24].)*

The following is an alternative definition of $\mathbf{L}_2^{\mathbf{P}}$, which was shown in [KP24] to be equivalent.

Definition 2.11 ($\mathbf{L}_2^{\mathbf{P}}$ via symmetric alternation)**.** *A language $L \in \mathbf{L}_2^{\mathbf{P}}$ if there is a ternary relation R computable in time $s(n)$, where s is a polynomial,*

$$R \subseteq \{0,1\}^n \times \{0,1\}^{s(n)} \times \{0,1\}^{s(n)},$$

denoted $R_x(u,v)$ for $x \in \{0,1\}^n$, $u,v \in \{0,1\}^{s(n)}$, such that, for every fixed x, it defines a linear order on $s(|x|)$-size strings such that:

- *for every $x \in L$, the minimal element of this order starts with bit 1,*
- *for every $x \notin L$, the minimal element of this order starts with bit 0.*

It is immediate that the latter version of the definition is a particular case of the definition of $\mathbf{S}_2^{\mathbf{P}}$ [Can96, RS98]:

Definition 2.12. *A language $L \in \mathbf{S}_2^{\mathbf{P}}$ if there is a polynomial-time computable ternary relation $R \subseteq \{0,1\}^n \times \{0,1\}^{s(n)} \times \{0,1\}^{s(n)}$, denoted $R_x(u,v)$ for $x \in \{0,1\}^n$, $u,v \in \{0,1\}^{s(n)}$, such that:*

- *for every $x \in L$, there exists $w^{(1)}$ such that $\forall v$ $R_x(w^{(1)},v) = 1$,*

- *for every $x \notin L$, there exists $w^{(0)}$ such that $\forall u\ R_x(u, w^{(0)}) = 0$.*

We now formally define the class $\mathbf{O_2^P}$ [CR06], which is the input-oblivious version of $\mathbf{S_2^P}$. Since we will need also a promise version of it, we start with defining this generalization.

Definition 2.13. *A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ belongs to $\mathbf{prO_2^P}$ if there is a polynomial-time deterministic Turing machine A such that for every $n \in \mathbb{N}$, there exist $w_n^{(0)}$, $w_n^{(1)}$ (called* irrefutable certificates*) that satisfy for every $x \in \{0,1\}^n$:*

- *If $x \in \Pi_{\text{YES}}$, then for every v, $A(x, w_n^{(1)}, v) = 1$,*
- *If $x \in \Pi_{\text{NO}}$, then for every u, $A(x, u, w_n^{(0)}) = 0$.*

No assumption on the behavior of A is made outside the promise set except that it stops (accepts or rejects) in polynomial time.

$\mathbf{O_2^P}$ is the respective class of languages (that is, it corresponds to the case of $\Pi_{\text{YES}} = \overline{\Pi_{\text{NO}}}$).

We remind the definition of another oblivious promise class.

Definition 2.14. *A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ belongs to* **prOMA** *if there is a polynomial-time deterministic Turing machine A and, for every $n \in \mathbb{N}$, there exists w_n (a witness that serves for every positive instance of length n), that satisfy the following conditions for every $x \in \{0,1\}^n$:*

- *If $x \in \Pi_{\text{YES}}$, then $\forall r\ A(x, r, w_n) = 1$,*
- *If $x \in \Pi_{\text{NO}}$, then $\forall w\ \Pr_r[A(x, r, w) = 1] < 1/2$.*

Finally, we define also an input-oblivious version of $\mathbf{L_2^P}$.

Definition 2.15. *A language L belongs to $\mathbf{OL_2^P}$ if there is a polynomial p, polynomial-time deterministic Turing machine V computing a ternary predicate $\{0,1\}^n \times \{0,1\}^{p(n)} \times \{0,1\}^{p(n)}$ (we use the notation $V_x(u,v)$ to denote its result), and two sequences of length-$p(n)$ bit strings $(y_n)_{n\in\mathbb{N}}$, $(z_n)_{n\in\mathbb{N}}$ such that*

- *for every x, V_x is a strict linear order (define $u <_x v$ iff $V_x(u,v) = 1$),*
- $\forall x \in L \cap \{0,1\}^n \quad 1y_n = \min <_x$,
- $\forall x \in \overline{L} \cap \{0,1\}^n \quad 0z_n = \min <_x$.

Remark 2.16. *One may further require that not only the minimal element but also the entire ordering, for inputs of the same lengths, to coincide. Our results still hold true under this definition as well, although it is unclear whether the two definitions are equivalent or which will ultimately prove more useful.*

Remark 2.17. *Note that unlike $\mathbf{NP} \subseteq \mathbf{L_2^P}$, it is unclear whether $\mathbf{ONP} \subseteq \mathbf{OL_2^P}$. Therefore, when we need both these input-oblivious classes, we need to specify both oracles (as in Corollary 5.6(2)).*

We also define **SBP** for the sake of self-completeness.

Definition 2.18 ([BGM06])**.** *A language L is in* **SBP** *if there exist $\varepsilon > 0$, $k, \ell \in \mathbb{N}$ and a polynomial-time computable predicate $B(x,r)$ such that*

- *If $x \in L$ then $\implies \Pr_r[B(x,r) = 1] \geq (1+\varepsilon) \cdot \frac{1}{2^{n^k}}$,*
- *If $x \notin L$ then $\implies \Pr_r[B(x,r) = 1] \leq (1-\varepsilon) \cdot \frac{1}{2^{n^k}}$.*

where $n = |x|$ and r is uniformly distributed on $\{0,1\}^{n^\ell}$.

3 $\mathbf{L_2^P \subseteq P^{prSBP}}$

In this section we prove Theorem 3. Our proof strategy is as follows: Given a point in a linear order, we aim to move "down the order" (i.e. towards "smaller" points). At each stage we will skip over a constant fraction of the points remaining on our way to the minimum. In order to find the next point, we will employ a binary-search-like procedure to determine the bits of the desired point, one coordinate at a time. Here is where our **prSBP** oracle comes into play: At each step, we look at the remaining set of points partitioned into two subsets: the points where the appropriate bit is 0 and where that bit is 1, and select the subset with the (approximately) smaller average order rank.

Before we proceed with the main algorithm, we show how to approximate the size of a set using a **prSBP** oracle. This procedure could be of independent interest.

3.1 Approximate Counting

In this section we observe that one can approximate deterministically the number of satisfying assignments for a Boolean circuit using a **prSBP** oracle (i.e. $\mathbf{FP^{prSBP}}$). Previously, it was shown that this task could be carried out by a randomized algorithm using an **NP** oracle ($\mathbf{FBPP^{NP}}$) [Sto85, JVV86] and by a deterministic algorithm using a **prAM** oracle ($\mathbf{FP^{prAM}}$) [Sip83, GS86]. Note that queries to **prSBP**/**prAM** can be thought of as queries to specific promise problems.

Definition 3.1 (Set-Size Estimation, SSE and WSSE)**.** *Let C be a Boolean circuit and $m \geq 1$ be an integer given in binary representation. Then* $\mathrm{SSE} := (\mathrm{SSE}_{\mathrm{YES}}, \mathrm{SSE}_{\mathrm{NO}})$*, where*

$$\begin{aligned} \mathrm{SSE}_{\mathrm{YES}} &= \{(C, m) \mid \#_x C(x) \geq m\}, \\ \mathrm{SSE}_{\mathrm{NO}} &= \{(C, m) \mid \#_x C(x) \leq m/2\}. \end{aligned}$$

If C is a non-deterministic circuit, we denote the corresponding problem by WSSE*.*

These two promise problems are complete for promise classes **prSBP** and **prAM**, respectively. This is what is proved essentially in [BGM06] and [GS86] and formulated explicitly in [Vol20].

Lemma 3.2 (Implicit in [BGM06])**.** SSE *is* **prSBP***-complete.*

Lemma 3.3 (Implicit in [GS86])**.** WSSE *is* **prAM***-complete.*

In particular, these complete problems showcase that **prSBP** $\subseteq$ **prAM**. The following lemma implies that ApproxCount $\in \mathbf{FP}_{\|}^{\mathbf{prSBP}}$ and, in fact, provides a slightly stronger result in the form of one-sided approximation.

Lemma 3.4. *There exists a deterministic algorithm that given a Boolean circuit C on n variables and a rational number $\varepsilon > 0$ outputs an integer number t satisfying*

$$\#_x C \leq t \leq 4^{\varepsilon/3} \cdot \#_x C \leq (1+\varepsilon)\#_x C$$

in time polynomial in n, the size of C and $\frac{1}{\varepsilon}$, making non-adaptive oracle queries to SSE*.*

The result appears to follow from a combination of previous techniques (and may be considered "folklore"). For completeness, we now provide its proof. We begin with a definition and a useful inequality.

For a unary relation $R(x)$, we denote the number of elements in R as $\#_x R(x) := |\{x \mid x \in R\}|$. We will also abbreviate this notation to $\#_x R$. For $k \in \mathbb{N}$, we define $R^{\otimes k}$ – the k-th *tensor power of* R as

$$R^{\otimes k}((x_1, \ldots, x_k)) := R(x_1) \wedge R(x_2) \wedge \ldots \wedge R(x_k),$$

where $x_1, \ldots, x_k$ are k disjoint copies of the argument x of R, respectively.

Observation 3.5. *Then* $\#_x(R^{\otimes k}) = (\#_x R)^k$.

Observation 3.6 (Bernoulli's inequality)**.** *Let* $0 \leq \varepsilon \leq 1$. *Then* $4^{\varepsilon/3} \leq 1 + \varepsilon$.

Lemma 3.7 (Lemma 3.4 rephrased)**.** *There exists a deterministic algorithm that given a Boolean circuit* C *on* n *variables and a rational number* $\varepsilon > 0$ *outputs an integer number* t *satisfying*

$$\#_x C \leq t \leq 4^{\varepsilon/3} \cdot \#_x C \leq (1 + \varepsilon)\#_x C$$

in time polynomial in n*, the size of* C *and* $\frac{1}{\varepsilon}$*, making non-adaptive oracle queries to* SSE.

Proof. Let O be a language consistent with SSE. Set $k := \lceil \frac{3}{\varepsilon} \rceil$ and define $\hat{C} := C^{\otimes k}$. Consider the following algorithm:

1. If $(C, 1) \notin O$ **return** $t = 0$ // $\#_x C = 0$
2. Find i as the smallest j between 1 and $nk + 1$ such that $\left(\hat{C}, 2^j\right) \notin O$
3. **return** $t = \left\lfloor 2^{i/k} \right\rfloor$

If $\#_x C = 0$ then $(C, 1) \notin O$ and the algorithm outputs $t = 0$. Otherwise, $\#_x \hat{C} \geq 1$ and hence $\left(\hat{C}, 1\right) \in O$. On the other hand, by definition $\#_x \hat{C} \leq 2^{nk}$ and hence $\left(\hat{C}, 2^{nk+1}\right) \notin O$. Therefore, i is well-defined and we have that:

- $\left(\hat{C}, 2^i\right) \notin O \implies \#_x \hat{C} < 2^i \implies (\#_x C)^k < 2^i \implies \#_x C < 2^{i/k} \implies \#_x C \leq t$,
- $\left(\hat{C}, 2^{i-1}\right) \in O \Rightarrow \#_x \hat{C} > \frac{2^{i-1}}{2} \Rightarrow (\#_x C)^k > 2^{i-2} \Rightarrow 4^{1/k} \cdot \#_x C > 2^{i/k} \Rightarrow 4^{1/k} \cdot \#_x C > t$.

In conclusion,

$$\#_x C \leq t \leq 4^{1/k} \cdot \#_x C \leq 4^{\varepsilon/3} \cdot \#_x C \leq (1 + \varepsilon)\#_x C.$$

For the running time, the algorithm finds i by either querying the oracle O at most $nk+2$ times in parallel or (using binary search) $O(\log_2(nk))$ times sequentially, and thus the computation of t can be carried out in time $\mathrm{poly}(nk)$. □

3.2 Estimating the Average Order Rank w.r.t. a Linear Order

Let $U = \{0, 1\}^n$. A single-output Boolean circuit E with $2n$ inputs induces an ordering relation $\lessdot_E$ on U as

$$x \lessdot_E y \iff E(x, y) = 1.$$

If $\lessdot_E$ is a strict linear order, we call E a *linear order circuit*.

Observation 3.8. *There exists a deterministic Turing machine with* SAT *oracle that, given a circuit E on $2n$ variables, stops in time polynomial in n and the size of E and does the following: if E is a linear order circuit, it outputs "yes"; otherwise, it outputs a counterexample: a pair satisfying $x <_E y <_E x$ or a triple satisfying $x <_E y <_E z <_E x$.*

Fix any strict linear order $<$ on U.

Definition 3.9. *For an element $\alpha \in U$ we define its order rank as* $\operatorname{rank}(\alpha) := |\{x \in U \mid x < \alpha\}|$. *We can extend this definition to non-empty subsets $S \subseteq U$ of U by taking the average order rank: define* $\operatorname{rank}(S) := \frac{\sum_{x \in S} \operatorname{rank}(x)}{|S|}$. *If $S = \{x \in U \mid C(x) = 1\}$ is described by a circuit C, we use the same notation:* $\operatorname{rank}(C) = \operatorname{rank}(S)$.

Below are some useful observations that we will use later.

Observation 3.10.

- *For a non-empty subset $S \subseteq U$:* $|\{(\upsilon, \alpha) \in U \times S \mid \upsilon < \alpha\}| = |S| \cdot \operatorname{rank}(S)$.
- *For any $\alpha \in U$:* $\operatorname{rank}\{\upsilon \in U \mid \upsilon \leq \alpha\} = \operatorname{rank}(\alpha)/2$.
- *Let $S_0, S_1 \subseteq U$ be two non-empty* disjoint *subsets of U. Then*

$$\operatorname{rank}(S_0 \cup S_1) = \frac{|S_0| \cdot \operatorname{rank}(S_0) + |S_1| \cdot \operatorname{rank}(S_1)}{|S_0| + |S_1|}.$$

Proof. These observations follow from the definitions:

- $|\{(\upsilon, \alpha) \in U \times S \mid \upsilon < \alpha\}| = \sum_{\alpha \in S} |\{(\upsilon, \alpha) \mid U \ni \upsilon < \alpha\}| = \sum_{\alpha \in S} \operatorname{rank}(\alpha) = |S| \cdot \operatorname{rank}(S)$.
- $\operatorname{rank}\{\upsilon \in U \mid \upsilon \leq \alpha\} = \frac{1}{\operatorname{rank}(\alpha)+1} \cdot \sum_{\upsilon \leq \alpha} \operatorname{rank}(\upsilon) = \frac{1}{\operatorname{rank}(\alpha)+1} \cdot \frac{\operatorname{rank}(\alpha)(\operatorname{rank}(\alpha)+1)}{2} = \frac{\operatorname{rank}(\alpha)}{2}$.
- $\operatorname{rank}(S_0 \cup S_1) = \frac{1}{|S_0|+|S_1|} \cdot \left(\sum_{x \in S_0} \operatorname{rank}(x) + \sum_{y \in S_1} \operatorname{rank}(y) \right) = \frac{|S_0| \cdot \operatorname{rank}(S_0) + |S_1| \cdot \operatorname{rank}(S_1)}{|S_0|+|S_1|}$.

□

In the following lemma the rank is defined w.r.t. the order $<_E$ described by a linear order circuit E. This lemma allows us to estimate the order rank of a set using a **prSBP** oracle.

Lemma 3.11. *There exists a deterministic algorithm that given a Boolean circuit C on n variables, a linear order circuit E on $2n$ variables, and an $\varepsilon > 0$, outputs a rational number r satisfying:*

$$4^{-\varepsilon} \cdot \operatorname{rank}(C) \leq r \leq 4^{\varepsilon} \cdot \operatorname{rank}(C)$$

in time polynomial in n, the sizes of C and E, and in $\frac{1}{\varepsilon}$, given oracle access to SSE.

Proof. Consider a circuit $D(x, y) := C(y) \wedge E(x, y)$. That is, y is accepted by C and $x <_E y$. By Observation 3.10, $\#_{(x,y)} D = \#_x C \cdot \operatorname{rank}(C)$. By Lemma 3.4 we can compute integers t_C and t_D that approximate the numbers $\#_x C$ and $\#_{(x,y)} D$, respectively. Formally,

$$\#_x C \leq t_C \leq 4^{\varepsilon} \cdot \#_x C \text{ and } \#_{(x,y)} D \leq t_D \leq 4^{\varepsilon} \cdot \#_{(x,y)} D.$$

Therefore we obtain:

$$4^{-\varepsilon} \cdot \operatorname{rank}(C) \leq \frac{\#_{(x,y)} D}{4^{\varepsilon} \cdot \#_x C} \leq \frac{t_D}{t_C} \leq \frac{4^{\varepsilon} \cdot \#_{(x,y)} D}{\#_x C} = 4^{\varepsilon} \cdot \operatorname{rank}(C).$$ □

3.3 Finding the Minimum Using a prSBP Oracle

We use the approximation algorithms developed above in order to find an element that is much closer to the minimum than a given element. The following lemma describes the procedure Back that given an element α finds another element β whose order rank is smaller by a constant factor. We will use this procedure afterwards in order to find the minimum in a polynomial number of iterations.

The procedure proceeds by determining the bits of the new element, one coordinate at a time, using a **prSBP** oracle. The order rank is w.r.t. the order $\leq_E$ described by a linear order circuit E.

Lemma 3.12. *There exists a deterministic algorithm* Back *that given a linear order circuit E on $2n$ variables and an element $\alpha \in \{0,1\}^n$, outputs an element $\beta \in \{0,1\}^n$ such that* $\mathrm{rank}(\beta) \leq \frac{\mathrm{rank}(\alpha)}{\sqrt{2}}$*, in time polynomial in n and the size of E, given oracle access to* SSE.

Proof. Consider the following procedure:

Back(E, α) :

1. Define $C(x) := E(x, \alpha) \vee x = \alpha$. *// The set of all elements that are $\leq_E$ than or equal to α*
2. Set $\varepsilon = 1/(8n)$.
3. For $i = 1$ to n:
   (a) For $b \in \{0,1\}$: define $C_b := C|_{x_i=b}$
   (b) For $b \in \{0,1\}$: if $\#_x C_b = 0$ then set $C := C_{1-b}$, $\beta_i := 1 - b$; continue to the next i
   // If one of the sets is empty, we choose the other one
   (c) For $b \in \{0,1\}$: use Lemma 3.11 to approximate $\mathrm{rank}(C_b)$ with ε into r_b
   (d) If $r_1 \geq r_0$ then $C := C_0, \beta_i := 0$ else $C := C_1, \beta_i := 1$
   // Choose the set with smaller approximate order

After each iteration one more variable x_i gets its value β_i and is substituted into C, that is, in the current circuit C variables $x_1, \ldots, x_i$ are replaced by the corresponding constants $\beta_1, \ldots, \beta_i$. We claim that *after* each iteration the order rank of the resulting circuit is bounded from the above: $\mathrm{rank}(C) \leq 4^{2\varepsilon i} \cdot \frac{\mathrm{rank}(\alpha)}{2}$.

Indeed, by Observation 3.10, *before* the first iteration, we have that $\mathrm{rank}(C) = \frac{\mathrm{rank}(\alpha)}{2}$. Now consider any iteration. If C_1 or C_0 are empty, then $\mathrm{rank}(C)$ remains the same and $4^{2\varepsilon i} \leq 4^{2\varepsilon(i+1)}$. Otherwise, by Lemma 3.11, for $b \in \{0,1\}$:

$$4^{-\varepsilon} \cdot \mathrm{rank}(C_b) \leq r_b \leq 4^{\varepsilon} \cdot \mathrm{rank}(C_b).$$

If $r_1 \geq r_0$ then $\mathrm{rank}(C_1) \geq r_1 \cdot 4^{-\varepsilon} \geq r_0 \cdot 4^{-\varepsilon} \geq \mathrm{rank}(C_0) \cdot 4^{-2\varepsilon}$ and therefore by Observation 3.10:

$$\begin{aligned}\mathrm{rank}(C) &= \frac{\#_x C_0 \cdot \mathrm{rank}(C_0) + \#_x C_1 \cdot \mathrm{rank}(C_1)}{\#_x C_0 + \#_x C_1} \\ &\geq \frac{\#_x C_0 \cdot \mathrm{rank}(C_0) + \#_x C_1 \cdot \mathrm{rank}(C_0) \cdot 4^{-2\varepsilon}}{\#_x C_0 + \#_x C_1} \geq \mathrm{rank}(C_0) \cdot 4^{-2\varepsilon}.\end{aligned}$$

Equivalently, $\mathrm{rank}(C_0) \leq \mathrm{rank}(C) \cdot 4^{2\varepsilon}$. Similarly, if $r_1 < r_0$ then $\mathrm{rank}(C_1) \leq \mathrm{rank}(C) \cdot 4^{2\varepsilon}$. Therefore, at each step the order rank is multiplied at most by $4^{2\varepsilon}$.

Consequently, after the n-th iteration, C represents the set that contains only the element β and we have that

$$\mathrm{rank}(\beta) \leq 4^{2\varepsilon n} \cdot \mathrm{rank}(\alpha)/2 \leq \sqrt{2} \cdot \mathrm{rank}(\alpha)/2 = \mathrm{rank}(\alpha)/\sqrt{2}.$$

For the runtime, all the steps can be carried out in time polynomial in n and $\frac{1}{\varepsilon} = O(n)$. □

Note that the procedure Back has a unique fixed point, namely, the minimal element.

Observation 3.13. $\textsc{Back}(E, \alpha) = \alpha$ *if and only if* α *is the minimal element in* E.

Proof. $\mathrm{rank}(\alpha) = 0 \iff \mathrm{rank}(\alpha) \leq \mathrm{rank}(\alpha)/\sqrt{2}$. □

We are now ready to prove the main result of this section.

Theorem 3.14. $\mathbf{L}_2^{\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{prSBP}}$.

Proof. It suffices to construct a deterministic polynomial-time algorithm that solves LOP given oracle access to **prSBP**. Let E be a circuit on $2n$ inputs. We describe an algorithm for LOP.

1. Check that E is indeed a linear order using Observation 3.8.
2. Let $\alpha := 0^n$, $\beta := 1^n$.
3. While $\alpha \neq \beta$ repeat: $\alpha := \beta$, $\beta := \textsc{Back}(E, \alpha)$.
4. Output α.

Given Observation 3.8, we can assume w.l.o.g. that E is a linear order circuit. We claim that the algorithm will output the minimal element after at most $2n$ iterations. Indeed, by Lemma 3.12, the order rank of the element α after $2n$ iterations satisfies $\mathrm{rank}(\alpha) \leq \frac{\mathrm{rank}(1^n)}{\sqrt{2}^{2n}} \leq \frac{2^n - 1}{2^n} < 1$, thus the "While" cycle will terminate before that. □

4 Which Karp–Lipton–style Collapse is Better?

Chakaravarthy and Roy proved two Karp–Lipton–style collapses: down to $\mathbf{O}_2^{\mathbf{P}}$ [CR06] and down to $\mathbf{P}^{\mathbf{prMA}}$ [CR11]. These two classes seem to be incomparable thereby rising the question: which collapse result is stronger? We observe that the collapse to $\mathbf{P}^{\mathbf{prMA}}$ can actually be deepened to $\mathbf{P}^{\mathbf{prOMA}}$, where **prOMA** is the oblivious version of **prMA** — and subsequently show that latter class is contained in both previous classes. That is, $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{P}^{\mathbf{prMA}} \cap \mathbf{O}_2^{\mathbf{P}}$. Indeed, the "internal collapse" of **prMA** (and, in fact, even **prAM**) to **prOMA**, under the assumption that $\mathbf{NP} \subseteq \mathbf{P/poly}$, is implicit in [AKSS95]. Nonetheless, we include a formal proof in Section 4.1 below (Proposition 4.1) in order to present a self-contained argument:

$$\text{If } \mathbf{NP} \subseteq \mathbf{P/poly}, \text{ then } \mathbf{prAM} \subseteq \mathbf{prOMA} \text{ and } \mathbf{PH} = \mathbf{P}^{\mathbf{prOMA}}.$$

To show that this class is not only included in $\mathbf{P}^{\mathbf{prMA}}$ but also in $\mathbf{O}_2^{\mathbf{P}}$, we use two inclusions:

1. $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}}$.

2. $\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$.

For the first inclusion it suffices to show

$$\mathbf{prOMA} \subseteq \mathbf{prO}_2^{\mathbf{P}},$$

which is essentially proved in [CR06, Theorem 3]: $\mathbf{MA} \subseteq \mathbf{NO}_2^{\mathbf{P}}$, where $\mathbf{NO}_2^{\mathbf{P}}$ is a less known class that combines $\mathbf{S}_2^{\mathbf{P}}$ with $\mathbf{O}_2^{\mathbf{P}}$: one certificate is input-oblivious, while the other one is not. To "upgrade" this proof one needs to notice that the proof goes through for promise classes with all certificates being input-oblivious. As it is not difficult, we include a formal proof of this proposition in Section 4.2 below (Proposition 4.2). However, later (in Section 5.2) we give a tighter chain of containments that uses a different method for proving $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} (= \mathbf{O}_2^{\mathbf{P}})$, namely, the derandomization technique. It goes through newly introduced input-oblivious classes based on symmetric alternation.

The second inclusion

$$\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$$

seems novel, we prove it in Section 4.3 (Theorem 4.3). The containment result

$$\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{O}_2^{\mathbf{P}}$$

follows (Corollary 4.4).

4.1 A Karp–Lipton–style Collapse to $\mathbf{P}^{\mathbf{prOMA}}$

The following proposition shows that the collapse of [CR11] can be actually pushed down to $\mathbf{P}^{\mathbf{prOMA}}$. For this, we observe that by standard techniques (e.g. [AKSS95]) $\mathbf{prAM} \subseteq \mathbf{prOMA}$ under $\mathbf{NP} \subseteq \mathbf{P/poly}$.

Proposition 4.1. *If* $\mathbf{NP} \subseteq \mathbf{P/poly}$*, then* $\mathbf{prAM} \subseteq \mathbf{prOMA}$ *and* $\mathbf{PH} = \mathbf{P}^{\mathbf{prOMA}}$.

Proof. If $\mathbf{NP} \subseteq \mathbf{P/poly}$, then Merlin's proof in $\mathbf{prMA}$ can be made input-oblivious. Let us show it first for a larger class: $\mathbf{prAM}$. If $(\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \mathbf{prAM}$, then there is a polynomial-time Turing machine A such that

$$\begin{aligned} \forall x \in \Pi_{\text{YES}} \quad & \forall r\ \exists w\ A(x,r,w) = 1, \\ \forall x \in \Pi_{\text{NO}} \quad & \Pr_r[\exists w\ A(x,r,w) = 1] < 1/2. \end{aligned}$$

Let us ask Merlin to send a family S_n of CircuitSAT circuits of appropriate input sizes (slightly abusing the notation: S_n contains circuits for a polynomial range of input sizes, not just n); one can assume that they compute a correct satisfying assignment or say "no" (they can lie only if they say "no" for a satisfiable formula). Consider the Boolean circuit $A_{x,r}$ obtained by embedding x and r into A, its variables are the bits of the witness w. Now Arthur can use S_n to produce proofs by himself instead of Merlin's original proofs, because the following holds:

$$\begin{aligned} \forall n \in \mathbb{N}\ \exists S_n\ \forall x \in \Pi_{\text{YES}} \quad & \forall r\ A(x,r,S_n(A_{x,r})) = 1, \\ \forall x \in \Pi_{\text{NO}} \quad & \Pr_r[\exists S_n\ A(x,r,S_n(A_{x,r})) = 1] < 1/2. \end{aligned}$$

The first condition is true because Merlin can send correct CircuitSAT circuits. The second condition is true because if such a circuit S_n existed, then the original Merlin could have sent

$w = S_n(A_{x,r})$. Note that in this case although S_n may depend on x and r, A will able to catch a cheating Merlin.

Formally, a new Arthur A' expects S_n as a proof, applies S_n to $A_{x,r}$ itself and runs A on the resulting witness. Therefore, we get exactly the definition of **prOMA** (Def. 2.14):

$$\begin{aligned} \forall n \in \mathbb{N}\ \exists S_n\ \forall x \in \Pi_{\text{YES}} \qquad & \forall r\ A'(x, r, S_n) = 1, \\ \forall x \in \Pi_{\text{NO}} \qquad & \forall S\ \Pr_r[A'(x, r, S) = 1] < 1/2. \end{aligned}$$

Therefore, if $\mathbf{NP} \subseteq \mathbf{P/poly}$, then $\mathbf{prAM} \subseteq \mathbf{prOMA}$. In [CR11], it is shown that under the same premises $\mathbf{PH} = \mathbf{P}^{\mathbf{prAM}}$ and hence the claim follows by combining these results. □

4.2 Promise Oblivious Merlin–Arthur Protocols are in Promise $\mathbf{O_2^P}$

The following proof essentially repeats the proof of [CR06, Theorem 3], which says that $\mathbf{MA} \subseteq \mathbf{NO_2^P}$, we need to verify that it holds for promise problems as well, and Merlin's advice remains oblivious if it was oblivious before, that is, $\mathbf{prOMA} \subseteq \mathbf{prO_2^P}$.

Proposition 4.2. $\mathbf{prOMA} \subseteq \mathbf{prO_2^P}$.

Proof. Consider a promise problem $\Pi = \Pi_Y\ \dot\cup\ \Pi_N \in \mathbf{prOMA}$. There is a polynomial-time deterministic machine A such that

$$\forall n \in \mathbb{N}\ \exists w_n\ \forall x \in \Pi_{\text{YES}} \qquad \forall r\ A(x, r, w_n)) = 1, \tag{1}$$

$$\forall x \in \Pi_{\text{NO}} \qquad \forall w\ \Pr_r[A(x, r, w) = 1] < 1/2. \tag{2}$$

Condition (2) can be replaced by

$$\forall n \in \mathbb{N}\ \exists r'_n\ \forall x \in \Pi_N \cap \{0,1\}^n\ \forall w \in \{0,1\}^{p(n)}\ A(x, w, r'_n) = 0 \tag{2'}$$

because of the Adleman's trick (similarly to the treatment of $\mathbf{RP} \subseteq \mathbf{ONP}$ in [GM15] or $\mathbf{MA} \subseteq \mathbf{NO_2^P}$ in [CR06]): given x and w, the new verifier can apply the old one as $A(x, w, r_i)$ for $np(n)$ independent random strings $r_i \in \{0,1\}^{t(n)}$ in order to reduce the error from $\frac{1}{2}$ to $\frac{1}{2} \cdot \frac{1}{2^{np(n)}}$. Since for every pair (x, w) there are at most a $\frac{1}{2} \cdot \frac{1}{2^{np(n)}}$ fraction of strings $r'_n \in \{0,1\}^{tnp(n)}$ results in an error, there exists r' that satisfies (2'). It does not harm the condition (1) as well. Therefore, Merlin's proof w remains input-oblivious, whereas Arthur's universal random string r'_n is input-oblivious and w-oblivious. □

4.3 Merging input-oblivious promise queries

Theorem 4.3 (Theorem 5, restated)**.** $\mathbf{P}^{\mathbf{prO_2^P}} \subseteq \mathbf{O_2^P}$.

Remark: Formally, we show that $\mathbf{P}^{\Pi} \subseteq \mathbf{O_2^P}$ for every promise problem $\Pi \in \mathbf{prO_2^P}$.

Proof. Let $L \in \mathbf{P}^{\Pi}$ and let $M^\bullet$ be a deterministic oracle machine that decides L correctly given loose oracle access to Π (i.e. irrespective of the answers to its queries outside of the promise set), in time $p(n)$ (for a polynomial p). Consider the polynomial-time deterministic verifier $A(q, u, v)$ from the definition of $\Pi \in \mathbf{prO_2^P}$. For $n \in \mathbb{N}$, let $1, \ldots, p(n)$ be all possible lengths of oracle queries made by M given an input of length n. Define

$$W_n := (w_1^{(0)}, \ldots, w_{p(n)}^{(0)}, w_1^{(1)}, \ldots, w_{p(n)}^{(1)})$$

as a vector containing the irrefutable certificates (both "yes" and "no") of A for the appropriate input lengths. We now construct a new polynomial-time deterministic verifier $A'(x, U, V)$ that will demonstrate that $L \in \mathbf{O_2^P}$ and will show that, for any x, the string[8] $W_{|x|}$ constitutes an irrefutable certificate that can be used both as $U = (u_1^{(0)}, \ldots, u_{p(n)}^{(0)}, u_1^{(1)}, \ldots, u_{p(n)}^{(1)})$ and as $V = (v_1^{(0)}, \ldots, v_{p(n)}^{(0)}, v_1^{(1)}, \ldots, v_{p(n)}^{(1)})$.

Given (x, U, V) as an input, A' will simulate M. Whenever M makes an oracle query q to Π, A' will compute four bits:

$$a := A(q, u_{|q|}^{(1)}, v_{|q|}^{(0)}), \qquad b := A(q, v_{|q|}^{(1)}, u_{|q|}^{(0)}),$$
$$c := A(q, u_{|q|}^{(1)}, u_{|q|}^{(0)}), \qquad d := A(q, v_{|q|}^{(1)}, v_{|q|}^{(0)}),$$

and will proceed with the simulation of M as if the oracle answered $\ell := (a \wedge c) \vee (b \wedge d)$.

By definition, for any x, the machine M computes $L(x)$ correctly given the correct answers to the queries in the promise set (i.e. $q \in \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$) and irrespective of the oracle's answers outside of the promise set. Thus it suffices to prove that the oracle's answers to the queries in the promise set are computed correctly, which we show now by inspecting the four possible cases.

- Suppose $x \in L$ and $U = W_{|x|}$.
  - If $q \in \Pi_{\text{YES}}$ then $u_{|q|}^{(1)}$ is a 'yes'-irrefutable certificate and hence $a = c = 1 \implies \ell = 1$.
  - If $q \in \Pi_{\text{NO}}$ then $u_{|q|}^{(0)}$ is a 'no'-irrefutable certificate and hence $b = c = 0 \implies \ell = 0$.
- Suppose $x \notin L$ and $V = W_{|x|}$.
  - If $q \in \Pi_{\text{YES}}$ then $v_{|q|}^{(1)}$ is a 'yes'-irrefutable certificate and hence $b = d = 1 \implies \ell = 1$.
  - If $q \in \Pi_{\text{NO}}$ then $v_{|q|}^{(0)}$ is a 'no'-irrefutable certificate and hence $a = d = 0 \Longrightarrow \ell = 0$. □

Corollary 4.4. $\mathbf{P^{prOMA}} \subseteq \mathbf{O_2^P}$.

Proof. By Proposition 4.2 and Theorem 4.3. □

Remark 4.5. *When a semantic class without complete problems is used as an oracle, it may be ambiguous. However,* **prAM** *does have a complete problem* WSSE *(see Definition 3.1). By inspecting the proof of the collapse (Prop. 4.1) one can observe that* WSSE *actually belongs to* **prOMA** *under* $\mathbf{NP} \subseteq \mathbf{P/poly}$*, and the oracle Turing machine that demonstrates* $\mathbf{PH} = \mathbf{P^{prOMA}}$ *still queries a specific promise problem.*

For the inclusion of $\mathbf{P^{prOMA}}$ *in* $\mathbf{O_2^P}$*, the first part (Prop. 4.2) transforms one promise problem into another promise problem, thus in the inclusion* $\mathbf{P^{prOMA}} \subseteq \mathbf{P^{prO_2^P}}$ *it is also the case that a single oracle is replaced by (another) single oracle.*

Remark 4.6. *Unfortunately, the proof of Theorem 4.3 breaks down when one tries to extend it to* $\mathbf{P^{prS_2^P}}$ *since the queries to* $\mathbf{prS_2^P}$ *are computed adaptively and the computation maybe be altered given the (unpredictable) answers to the queries that lie outside of promise.*

However, almost the same proof shows that $\mathbf{P_{||}^{prS_2^P}} \subseteq \mathbf{S_2^P}$*. Here all the queries can be computed from the input itself, thus instead of the input-oblivious witness* W_n *one can give the non-input-oblivious witness specifically for these queries. The rest of the proof remains intact:* A' *will run* A *on the witnesses intended for the particular query and proceed as before.*

[8] There might be different versions of this string as the irrefutable certificates need not to be unique.

5 $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}}$

In this section we prove Theorem 1 by, essentially, expanding the proof of $\mathbf{MA} \subseteq \mathbf{L}_2^{\mathbf{P}}$ in [KP24]. We use the following statements from that paper and an earlier paper by Korten [Kor22]. We then proceed to the input-oblivious setting.

5.1 The non-input-oblivious setting

Definition 5.1 ([Kor22, Definitions 6, 7], [Kor21, Definitions 7, 8])**.** PRG *is the following search problem: given* 1^n*, output a pseudorandom generator*[9] $R = (x_1, \ldots, x_m)$*, that is, an array of strings* $x_i \in \{0,1\}^n$ *such that for every* n*-input circuit* C *of size* n*:*

$$\left| \Pr_{x \leftarrow U(R)} \{C(x) = 1\} - \Pr_{y \leftarrow U(\{0,1\}^n)} \{C(y) = 1\} \right| \leq \frac{1}{n}.$$

Korten demonstrates that such a generator containing $m = n^6$ strings can be constructed with a single oracle query to Avoid[10] (actually, the stretch is even larger than "plus 1 output wire").

Proposition 5.2 ([Kor22, Theorem 2], [Kor21, Theorem 3])**.** PRG *reduces in polynomial time to a single* Avoid *query.*

Korten and Pitassi demonstrate that Avoid (which they call Weak Avoid) can be solved with one oracle query to LOP.

Proposition 5.3 ([KP24, Theorem 1])**.** Avoid *is polynomial-time many-one reducible to* LOP*.*

The main result of this section is the following theorem.

Theorem 5.4 (Theorem 1, restated)**.** $\mathbf{P}^{\mathbf{prMA}} \subseteq \mathbf{L}_2^{\mathbf{P}}$.

Proof. We show how to replace the oracle **prMA** by $\mathbf{L}_2^{\mathbf{P}}$. Since $\mathbf{P}^{\mathbf{L}_2^{\mathbf{P}}} = \mathbf{L}_2^{\mathbf{P}}$ by Definition 2.10, the result follows.

One can assume that calls to the **prMA** oracle are made for input lengths such that Arthur can be replaced by a circuit $A(x, w, r)$ of size at most $s(n)$ for a specific polynomial s. One can assume perfect completeness for A, that is, for x in the promise set "YES", there is w such that $\forall r \; A(x, w, r) = 1$.

Before simulating the **prMA** oracle, our deterministic polynomial-time Turing machine will make oracle calls to $\mathbf{L}_2^{\mathbf{P}}$ in order to build a pseudorandom generator sufficient to derandomize circuits of size $s(n)$. By Proposition 5.2, such a pseudorandom generator G, which is a sequence $G(1^{s(n)})$ of pseudorandom strings $g_1, \ldots, g_m \in \{0,1\}^{s(n)}$ for m bounded by a polynomial in $s(n)$, can constructed (for $m = s(n)^6$ and error $\frac{1}{s(n)}$) using a reduction to Avoid. Subsequently, by Proposition 5.3, Avoid is reducible to LOP. As a result, $\{g_i\}_{i=1}^m$ can be computed in deterministic polynomial time by querying an $\mathbf{L}_2^{\mathbf{P}}$ oracle.

After G is computed, each call to the **prMA** oracle can be replaced by an $\mathbf{NP} \subseteq \mathbf{L}_2^{\mathbf{P}}$ query $\exists w \; C(w)$ for the circuit C that computes the conjunction of the circuits $A(x, w, g_i)$ with hardwired x and g_i, for every i. Note that such queries constructed for x outside of the promise set are still valid **NP** queries even if Arthur does not conform to the definition of **MA** in this case. These oracle answers are irrelevant, because the original $\mathbf{P}^{\mathbf{prMA}}$ machine must return the correct (in particular, the same) answer irrespectively of the oracle's answer. □

[9] Korten does not say that m has a polynomial dependence on n though we think it is assumed, and anyway his theorem provides a construction with $m = n^6$.

[10] In the paper the Avoid problem is referred to as "Empty".

5.2 The Input-Oblivious Setting

Korten proves (Prop. 5.2) that PRG (Def. 5.1) reduces to AVOID, and Korten and Pitassi [KP24] compute AVOID in $\mathbf{L}_2^P = \mathbf{P}^{\mathbf{L}_2^P}$. Since PRG has a unary input, one can observe that PRG can be computed using an input-oblivious oracle. This gives raise to tighter containments. Indeed, in the non-input oblivious setting this containments become equalities. However, input-oblivious classes lack some of the nice closure properties and therefore require a special treatment.

Proposition 5.5. PRG *can be computed in deterministic polynomial time with an* $\mathbf{OL}_2^P$ *oracle.*

Proof. It is shown in [Kor22] (see Prop. 5.2) that PRG consisting of $m = n^6$ strings can be computed in deterministic polynomial time with a single oracle query to AVOID; denote the algorithm generating this query by K, let us assume w.l.o.g. that it outputs a circuit C_n whose description has exactly n^d bits (padding if necessary), with $\ell(n)$ outputs, where $\ell(n)$ is the size of the solution for AVOID, and the constant d is large enough for $\ell(n) \leq n^{d-1}$. Since the input to this problem is unary, for each n, a single instance C_n of AVOID is to be solved, and its solution (i.e. any string outside of the image of C_n) solves PRG (in fact, without further processing). A specific solution to AVOID can be computed using a deterministic polynomial-time truth-table reduction to a language $L \in \mathbf{L}_2^P$ [KP24], that is, there is a polynomial p, a p-time DTM R computing the reduction, a p-time DTM T computing the truth table, a language $L \in \mathbf{L}_2^P$ with a polynomial-time verifier V computing an order $<_V$ such that $T[a_1, a_2, \ldots a_r]$ is a correct solution to AVOID, where r is the number of queries made by R (w.l.o.g. they are of the same size), and for each i, a_i is the first bit of the minimum certificate wrt $<_V$ applied to the i-th query.

Our oracle language $L' \in \mathbf{OL}_2^P$ is verified by the following verifier U (defining its order $<_U$), which merges queries using a construction similar to [KP24].

For two bit strings α, β we define the relation: $\alpha <_{lex} \beta$ to be 1 iff α is lexicographically smaller than β (and 0, otherwise).

Input: w
Certificates: y, z
Algorithm:

1. If $y = z$ as bit strings, then return 0. // $y \not<_U z$
2. Compute $t := |w|$.
3. Compute $n := \lfloor \sqrt[d]{t} \rfloor$.
4. Compute $i := t - n^d$.
5. Let a be the first bit of y, and b be the first bit of z.
   // *Informally, this bit is a claim for the value of the* i*-th bit of* PRG.
6. If $i < 1$ or $i > \ell(n)$ then // *Out of range:* t *is not of the form* $n^d + i$ *with* $1 \leq i \leq \ell(n)$.
   if $a \neq b$, then return the result of comparison $a < b$, otherwise return $y <_{lex} z$.
7. Run $K(1^n)$, denote the resulting circuit by C_n.
   // *Note that the length of its description is* n^d.
8. Run $R(C_n)$ to compute queries $q_1, q_2, \ldots, q_r$.

9. Parse y as $axy_1 \dots y_r$ and z as $bx'z_1 \dots z_r$, where $a, b \in \{0,1\}$, $x, x' \in \{0,1\}^{n^{\ell(n)}}$, $y_1, \dots, y_r, z_1, \dots, z_r \in \{0,1\}^s$, where s is the bit size of elements of $<_V$ for the queries computed by R.
// x, x' are candidates for the solution of AVOID *computed by R, T.*

10. Call y syntactically incorrect if either $x[i]$ (the i-th bit of x) differs from a or $T[y_1[1], \dots, y_r[1]] \neq x$.
// That is, y does not claim that the answer is $x[i]$, or its sub-certificates do not yield x as T's answer.

    Define syntactically incorrect z similarly.

11. If both y and z are syntactically incorrect, return $y <_{lex} z$.

12. If exactly only one of y, z is syntactically incorrect, state that it is greater than the other certificate, return 1 or 0 accordingly.

13. *Now y, z are syntactically correct.*
If $y_1 \dots y_r \neq z_1 \dots z_r$ then find the first j such that $y_j \neq z_j$ and return $V(q_j, y_j, z_j)$.

14. *We are done and never get to this point: the certificates are different, but the sub-certificates are equal, thus $x \neq x'$ so one of the certificates had to be recognized as syntactically incorrect by T lookup.*

Since $\ell(n) \leq n^{d-1} < dn^{d-1} \leq (n+1)^d - n^d$, the sets $\{n^d + 1, \dots, n^d + \ell(n)\}$ are pairwise disjoint for different n, so steps 3-4 correctly recover n and i from t. Hence the queries 1^{n^d+i}, $1 \leq i \leq \ell(n)$, reveal all $\ell(n)$ bits of the Avoid solution, i.e., of the PRG. Our verifier U is input-oblivious (it simply ignores the input and only uses its length), it computes a linear order, and its minimal element starts with the bit equal to the i-th bit of the pseudorandom generator computed by the composition of K, R, and T. Thus U defines a language in $\mathbf{OL_2^P}$, and consequent queries 1^{n^d+i} to it reveal the bits of the pseudorandom generator. □

Corollary 5.6.

1. $\mathbf{P^{prMA}} \subseteq \mathbf{P^{OL_2^P, NP}} \subseteq \mathbf{L_2^P}$.

2. $\mathbf{P^{prOMA}} \subseteq \mathbf{P^{OL_2^P, prONP}} \subseteq \mathbf{O_2^P} \cap \mathbf{L_2^P}$.

Proof. The first inclusion follows similarly to the proof of Theorem 5.4: the lower DTM can compute PRG using its $\mathbf{OL_2^P}$ oracle (Prop. 5.5) and then use it to derandomize **prMA** (resp., **prOMA**) using its **NP** (resp., **prONP**) oracle, which will get both a query to the old oracle and the PRG (note that since for **prOMA** we are talking about a **prONP** oracle, we do not need to discuss the (non-occurring) situation where the candidate PRG string it could be given is wrong: we put such strings out of the promise set).

The second inclusion in all the items follows from $\mathbf{NP} \subseteq \mathbf{L_2^P}$, $\mathbf{prONP} \subseteq \mathbf{prO_2^P}$, $\mathbf{OL_2^P} \subseteq \mathbf{L_2^P}$ (syntactically), $\mathbf{OL_2^P} \subseteq \mathbf{O_2^P}$ (similarly to $\mathbf{L_2^P} \subseteq \mathbf{S_2^P}$), $\mathbf{P^{L_2^P}} = \mathbf{L_2^P}$ [KP24], $\mathbf{P^{O_2^P}} = \mathbf{O_2^P}$ ($= \mathbf{P^{prO_2^P}}$ (Theorem 4.3)). □

6 Discussion and Further Research

Some Unresolved Containments

1. Can one strengthen our inclusion $\mathbf{L_2^P} \subseteq \mathbf{P^{prSBP}}$ to $\mathbf{S_2^P} \subseteq \mathbf{P^{prSBP}}$? One can try combining our techniques with the proof of $\mathbf{S_2^P} \subseteq \mathbf{P^{prAM}}$ by Chakaravarthy and Roy [CR11].

   Note that in the other direction it is open even whether $\mathbf{SBP} \subseteq \mathbf{S_2^P}$.

2. Chakaravarthy and Roy [CR11] asked whether $\mathbf{P^{prMA}}$ and $\mathbf{P^{prS_2^P}}$ are contained in $\mathbf{S_2^P}$. While we resolved the first question, the second one remains open. We note that, although in the input-oblivious world both inclusions hold ($\mathbf{P^{prOMA}} \subseteq \mathbf{P^{prO_2^P}} \subseteq \mathbf{O_2^P}$, Corollary 4.4), the proof of the latter inclusion (Theorem 5) is essentially input-oblivious (one needs to give all the certificates for the oracle non-adaptively, and queries cannot be predicted because oracle answers cannot be predicted for promise problems, thus in the non-input-oblivious setting it works for parallel queries only (Remark 4.6)).

3. As was mentioned, the $\mathbf{FP^{prSBP}}$ procedure for approximate counting can be implemented in $\mathbf{FP_{\|}^{prSBP}}$ — that is, using parallel (i.e. non-adaptive) oracle queries. On the other hand the containment $\mathbf{L_2^P} \subseteq \mathbf{P^{prSBP}}$, which uses approximate counting as a black-box subroutine, seems to require sequential, adaptive queries. Could one implement the latter containment using parallel queries (i.e. show that $\mathbf{L_2^P} \subseteq \mathbf{P_{\|}^{prSBP}}$)? In particular, as $\mathbf{PP}$ is consistent with $\mathbf{prSBP}$ [BGM06] and is closed under non-adaptive Turing reductions [FR96], this would imply that $\mathbf{L_2^P} \subseteq \mathbf{PP}$. Note that it is unknown even whether $\mathbf{P^{NP}} \subseteq \mathbf{PP}$, while $\mathbf{P^{NP}} \subseteq \mathbf{L_2^P}$. Moreover, there is an oracle separating the former two classes [Bei94].

4. A recent work of Gajulapalli et al. [GGLS25] places $\mathbf{L_2^P}$ in the class[11] $\mathbf{UEOPL^{NP}}$, which appears to be incomparable to our result (Theorem 3). Can one determine the relative status of $\mathbf{UEOPL^{NP}}$ and $\mathbf{P^{prSBP}}$?

5. Similarly to $\mathbf{L_2^P}$, one could define a class of languages reducible to Avoid. A similar class of search problems, $\mathbf{APEPP}$, has been defined by [KKMP21, Kor22] (and Korten [Kor22] proved that constructing a hard truth table is a problem that is complete for this class under $\mathbf{P^{NP}}$-reductions); however, we are asking about a class of languages. Korten and Pitassi have shown that $\mathbf{L_2^P}$ can be equivalently defined using many-one, Turing, or $\mathbf{P^{NP}}$-reductions, thus there are several options. One can observe that the containment $\mathbf{P^{prMA}} \subseteq \mathbf{L_2^P}$ (Theorem 5.4) is essentially proved via the intermediate class $\mathbf{P^{Avoid,NP}}$ that uses both an oracle for *Range Avoidance* (a single-valued or an essentially unique [KP24] version) and an oracle for SAT. Can one prove that one of the containments in $\mathbf{P^{prMA}} \subseteq \mathbf{P^{Avoid,NP}} \subseteq \mathbf{L_2^P}$ is in fact an equality?

6. We defined an input-oblivious version $\mathbf{OL_2^P}$ of $\mathbf{L_2^P}$, however, contrary to $\mathbf{NP} \subseteq \mathbf{L_2^P}$, it is not obvious whether $\mathbf{ONP} \subseteq \mathbf{OL_2^P}$ or even $\mathbf{ONP} \subseteq \mathbf{P^{prOL_2^P}}$.

Relations between collapse results and "hard" functions. Starting from Karp–Lipton's paper [KL80], Kannan's fixed-polynomial circuit complexity lower bounds [Kan82] were improving accordingly to new collapses: if a new collapse $\mathbf{NP} \subseteq \mathbf{P/poly} \implies \mathbf{PH} = \mathcal{C}$ is shown for a class $\mathcal{C}$

[11] $\mathbf{UEOPL}$ consists of problems that are many-one polynomial-time reducible to Unique-End-of-Potential-Line, see [FGMS20, GGLS25].

containing **NP**, it immediately implies lower bounds for this class, because if $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, we are already done.

However, a collapse to $\mathbf{O_2^P}$ [CR06] did not imply lower bounds for $\mathbf{O_2^P}$, because **NP** is unlikely to be contained in it (after all, $\mathbf{O_2^P} \subseteq \mathbf{P/poly}$). It was not until nearly two decades later that lower bounds for $\mathbf{O_2^P}$ have been shown by Gajulapalli, Li, and Volkovich [GLV24] building on recent progress for the range avoidance problem [Kor22, Li24], thus matching the progress on the two questions again.

Korten and Pitassi [KP24] have shown fixed-polynomial lower bounds for their new class $\mathbf{L_2^P}$ without showing a collapse result, thereby introducing a misalignment once again, yet this time in the opposite direction. Our paper's inclusion $\mathbf{P^{prMA}} \subseteq \mathbf{L_2^P}$ restores the balance.

However, we observed that in the input-oblivious world the currently best collapse is to $\mathbf{P^{prOMA}}$, which reopens this question. Does this class possess fixed-polynomial circuit lower bounds? One can observe that Santhanam's proof [San09] of $\mathsf{Size}[n^k]$ lower bounds for promise problems in **prMA** is input-oblivious. Indeed, the presented hard promise problems are actually in **prOMA**! However, these **promise problems** do not yield a **language** in $\mathbf{P^{prOMA}}$ that is hard for $\mathsf{Size}[n^k]$, and we leave this question for further research.

The best class for which fixed-polynomial circuit lower bounds can be proved (trivially) is $\mathbf{P}^{\varepsilon\text{-Hard-tt}}$ (for any particular fixed $\varepsilon > 0$), where ε-Hard-tt, asks[12] given 1^{2^n}, to output a truth table of a function $\{0,1\}^n \to \{0,1\}$ of circuit complexity at least $2^{\varepsilon n}$. Can one prove a collapse to this class (or at least to $\mathbf{P}^{\text{Avoid}}$)? Note that for the purpose of fixed-polynomial lower bounds even a limited version of ε-Hard-tt suffices where the truth table is non-empty for a number of entries greater than any polynomial and its complexity is only superpolynomial.

Acknowledgment

The authors are grateful to Yaroslav Alekseev for discussions, to Dmitry Itsykson for discussing and proofreading a preliminary version of this paper, and to Jan Krajíček for providing multiple references and for useful discussions.

References

[AGHK11] B. Aydinlioğlu, D. Gutfreund, J. M. Hitchcock, and A. Kawachi. Derandomizing Arthur–Merlin games and approximate counting implies exponential-size lower bounds. *Comput. Complex.*, 20(2):329–366, 2011.

[AKKT20] S. Aaronson, R. Kothari, W. Kretschmer, and J. Thaler. Quantum lower bounds for approximate counting via Laurent polynomials. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 7:1–7:47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[AKSS95] V. Arvind, J. Köbler, U. Schöning, and R. Schuler. If NP has polynomial-size circuits, then MA=AM. *Theor. Comput. Sci.*, 137(2):279–282, 1995.

[AR20] S. Aaronson and P. Rall. Quantum approximate counting, simplified. In *3rd Symposium on Simplicity in Algorithms, SOSA 2020*, pages 24–32. SIAM, 2020.

[12] Korten [Kor22] defines a smoother version of it where the input length is not necessarily a power of two.

[BCG+96] N. H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996.

[Bei94] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994.

[BF99] H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In *STACS*, pages 100–109, 1999.

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

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

[BGM06] E. Böhler, C. Glaßer, and D. Meister. Error-bounded probabilistic computations between MA and AM. *J. Comput. Syst. Sci.*, 72(6):1043–1076, 2006.

[BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, pages 116–127, 1992.

[Cai07] J.-Y. Cai. $S_2P \subseteq ZPP^{NP}$. *Journal of Computer and System Sciences*, 73(1):25–35, 2007.

[Can96] R. Canetti. More on BPP and the polynomial-time hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996.

[CHR24] L. Chen, S. Hirahara, and H. Ren. Symmetric exponential time requires near-maximum circuit size. In *Proceedings of the 56th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2024, to appear. Association for Computing Machinery, 2024.

[CK98] M. Chiari and J. Krajíček. Witnessing functions in bounded arithmetic and search problems. *The Journal of Symbolic Logic*, 63(3):1095–1115, 1998.

[CMMW19] L. Chen, D. M. McKay, C. D. Murray, and R. R. Williams. Relations and equivalences between circuit lower bounds and Karp-Lipton theorems. In *CCC-2019, LIPICS*, pages 30:1–21, 2019.

[CR06] V. T. Chakaravarthy and S. Roy. Oblivious symmetric alternation. In *STACS*, pages 230–241, 2006.

[CR11] V. T. Chakaravarthy and S. Roy. Arthur and Merlin as oracles. *Comput. Complex.*, 20(3):505–558, 2011.

[FGMS20] J. Fearnley, S. Gordon, R. Mehta, and R. Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020.

[FR96] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. *Inf. Comput.*, 124(1):1–6, 1996.

[GGLS25] K. Gajulapalli, S. Ghentiyala, Z. Li, and S. Saraogi. Downward self-reducibility in the total function polynomial hierarchy. *Electron. Colloquium Comput. Complex.*, TR25-121, 2025.

[GLV24] K. Gajulapalli, Z. Li, and I. Volkovich. Oblivious complexity classes revisited: Lower bounds and hierarchies. In *44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2024*, volume 323 of *LIPIcs*, pages 23:1–23:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[GM15] O. Goldreich and O. Meir. Input-oblivious proof systems and a uniform complexity perspective on P/poly. *ACM Transactions on Computation Theory (TOCT)*, 7(4):1–13, 2015.

[GS86] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[GS88] J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988.

[HHT97] Y. Han, L. A. Hemaspaandra, and T. Thierauf. Threshold computation and cryptographic security. *SIAM J. Comput.*, 26(1):59–78, 1997.

[HS15] E. A. Hirsch and D. Sokolov. On the probabilistic closure of the loose unambiguous hierarchy. *Inf. Process. Lett.*, 115(9):725–730, 2015.

[IKV23] R. Impagliazzo, V. Kabanets, and I. Volkovich. Synergy between circuit obfuscation and circuit minimization. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023*, volume 275 of *LIPIcs*, pages 31:1–31:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. of Computer and System Sciences*, 65(4):672–694, 2002.

[Jeř04] E. Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129:1–37, 2004.

[Jeř07] E. Jeřábek. Approximate counting in bounded arithmetic. *The Journal of Symbolic Logic*, 72(3):959–993, 2007.

[JVV86] M. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.

[KKMP21] R. Kleinberg, O. Korten, D. Mitropolsky, and C. Papadimitriou. Total functions in the polynomial hierarchy. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[KL80] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 302–309, 1980.

[Kor21] O. Korten. The hardest explicit construction. *CoRR*, abs/2106.00875, 2021.

[Kor22] O. Korten. The hardest explicit construction. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 433–444. IEEE, 2022.

[KP24] O. Korten and T. Pitassi. Strong vs. Weak Range Avoidance and the Linear Ordering Principle. *Electron. Colloquium Comput. Complex.*, TR24-076, 2024.

[Kra25] J. Krajíček. *Proof Complexity Generators*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2025.

[KW98] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998.

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

[LR94] K.-J. Lange and P. Rossmanith. Unambiguous polynomial hierarchies and exponential size. In *Structure in Complexity Theory Conference*, pages 106–115. IEEE Computer Society, 1994.

[MAD25] S. C. Marshall, S. Aaronson, and V. Dunjko. Improved separation between quantum and classical computers for sampling and functional tasks. In *40th Computational Complexity Conference, CCC 2025, August 5-8, 2025, Toronto, Canada*, volume 339 of *LIPIcs*, pages 5:1–5:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[MVW99] P. B. Miltersen, N. V. Vinodchandran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, pages 210–220, 1999.

[OS18] R. O'Donnell and A. C. C. Say. The weakness of CTC qubits and the power of approximate counting. *ACM Trans. Comput. Theory*, 10(2):5:1–5:22, 2018.

[PWW88] J. Paris, A. Wilkie, and A. R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53(4):1235–1244, 1988.

[RS98] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Comput. Complex.*, 7(2):152–162, 1998.

[San09] R. Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

[Sch83] U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983.

[Sto85] L. J. Stockmeyer. On approximation algorithms for #P. *SIAM J. Comput.*, 14(4):849–861, 1985.

[SU06] R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. *Comput. Complex.*, 15(4):298–341, 2006.

[Tod91] S. Toda. PP is as hard as the polynomial time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

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

[Vol14] I. Volkovich. On learning, lower bounds and (un)keeping promises. In *Proceedings of the 41st ICALP*, pages 1027–1038, 2014.

[Vol20] I. Volkovich. The untold story of SBP. In *The 15th International Computer Science Symposium in Russia, CSR*, volume 12159 of *Lecture Notes in Computer Science*, pages 393–405. Springer, 2020.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il