

Upper and Lower Bounds for the Linear Ordering Principle

Edward A. Hirsch  

Department of Computer Science, Ariel University, Israel

Ilya Volkovich  

Boston College, Chestnut Hill, MA, USA

Abstract

Korten and Pitassi (FOCS, 2024) defined a new¹ complexity class L_2^P as the polynomial-time Turing closure of the Linear Ordering Principle (a total function extending finding the minimum of an order [18] to the case where the order is not linear). They put it between MA (Merlin–Arthur protocols) and S_2^P (the second symmetric level of the polynomial hierarchy).

In this paper we sandwich L_2^P between P^{prMA} and P^{prSBP} . (The oracles here are promise problems, and SBP is the only known class between MA and AM .) The containment in P^{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 $P^{prO_2^P} \subseteq O_2^P$ (where O_2^P is the input-oblivious version of S_2^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 $P^{prMA} \subseteq S_2^P$, thereby settling the relative standing of the existing (non-oblivious) Karp–Lipton–style collapse results of [15] and [12],
- We give an affirmative answer to an open question of Korten and Pitassi whether a Karp–Lipton–style collapse can be proven for L_2^P ,
- We show that the Karp–Lipton–style collapse to P^{prOMA} is actually better than both known collapses to P^{prMA} due to Chakaravarthy and Roy (Computational Complexity, 2011) and to O_2^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.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness; Theory of computation → Circuit complexity

Keywords and phrases Complexity Classes, Structural Complexity Theory, Linear Ordering Principle, Symmetric Alternation, Merlin–Arthur Protocols, Karp–Lipton Collapse

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.52

Related Version *Full Version*: <https://arxiv.org/abs/2503.19188>

Acknowledgements The authors are grateful to Yaroslav Alekseev for discussing and to Dmitry Itsykson for discussing and proofreading a preliminary version of this paper. 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.

1 Introduction

The seminal theorem of Richard M. Karp and Richard J. Lipton [32] connected non-uniform and uniform complexity by demonstrating a collapse of the Polynomial Hierarchy (PH) assuming NP has polynomial-size Boolean circuits. This collapse has since been very

¹ Note that this notation had been used in the past [45] for a very different class, which has been apparently forgotten after that.

instrumental in transferring lower bounds against Boolean circuits of *fixed-polynomial*² size to smaller classes of **PH**. Since then, these results were strengthened in many ways leading to “minimal” complexity classes that have such lower bounds and to which **PH** 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 ([12], 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 [13, 43] 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 $k \in \mathbb{N}$, $\mathbf{PH} \not\subseteq \text{Size}[n^k]$) by Kannan’s theorem [31], 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 [31, 8, 34, 50, 12, 44, 15, 51, 26]. Chen et al. [17] 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 [40]).

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 [14]. 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. [33] initiated the study of total functions beyond **TFNP**. While Karp–Lipton’s theorem has not been improved, lower bounds against $\text{Size}[n^k]$ were pushed down to \mathbf{O}_2^P [22] and \mathbf{L}_2^P [36], a new¹ important class which we describe in more detail below. At the same time, truly exponential lower bounds were established for \mathbf{S}_2^E [38] (as it turns out, $\mathbf{S}_2^E = \mathbf{O}_2^E$ [22]) and \mathbf{L}_2^E [36].

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 [35] and Li [38] reduced the question to the so-called *Range Avoidance* problem: given a function $f: \{0, 1\}^n \rightarrow \{0, 1\}^m$ with $m > n$, represented by a Boolean circuit, find a point outside its image (this problem is known under the name dWPHP in the bounded arithmetic community and has implications in proof complexity, see [42, 28] and [37] for survey). In [16, 38], Range Avoidance has been reduced to symmetric alternation. Subsequently, Korten and Pitassi [36] 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 it is known as MIN in the bounded arithmetic community [18]). A polynomial-time Turing closure of this principle gave rise to a new class $\mathbf{L}_2^P \subseteq \mathbf{S}_2^P$: a version

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 $\text{Size}[n^k]$.

of S_2^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 [44] has shown fixed-polynomial lower bounds for *promise problems* possessing such proofs (i.e. the class \mathbf{prMA}). In [15], Chakaravarthy and Roy have shown a Karp-Lipton-style collapse and thus fixed-polynomial size lower bounds for the class \mathbf{PprMA} . In particular, they presented a new upper bound for S_2^P by showing that $S_2^P \subseteq \mathbf{PprAM}$. Nonetheless, the relationship between \mathbf{PprMA} and the classes of symmetric alternation (including S_2^P , O_2^P , and then-unknown L_2^P) remained open.

Combining their upper bound for S_2^P with a result of [3], that $\mathbf{NP} \subseteq \mathbf{P/poly}$ implies an “internal collapse” $\mathbf{MA} = \mathbf{AM}$ (which goes through for the promise versions of the classes as well), [15] concluded that the Polynomial Hierarchy collapses all the way to \mathbf{PprMA} . Subsequently, by applying the win-win argument, they obtained fixed-polynomial bounds for \mathbf{PprMA} , which (unlike \mathbf{prMA}) is a class of languages. It is to be noted though that since \mathbf{prMA} is not a class of languages – while \mathbf{PprMA} is, there is no *immediate* way to carry any lower bound against \mathbf{prMA} over to \mathbf{PprMA} : 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 [5] 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 [27, 11], and the win-win argument yields superpolynomial lower bounds for some of them: $\mathbf{MAEXP} \not\subseteq \mathbf{P/poly}$ [10].

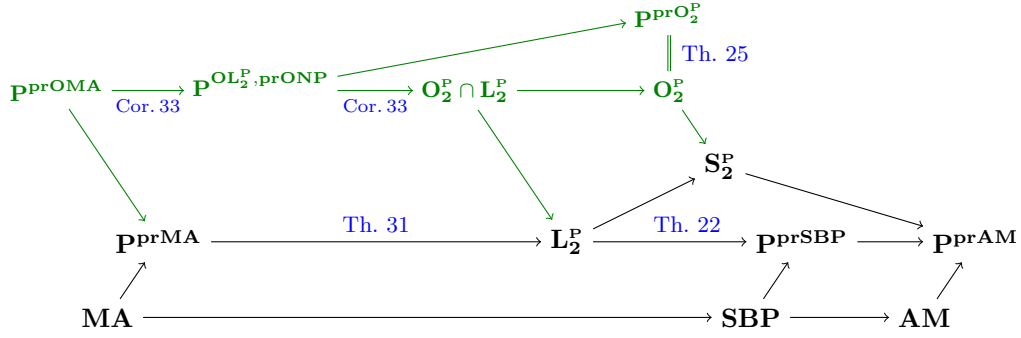
1.2 Promise Problems as Oracles

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 [15]. 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, 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 [24, 9], where the latter approach has been adopted. Nonetheless, we include a formal proof of this equivalence in the full version of the paper for the sake of completeness.

1.3 Our Contribution

In this paper we prove the inclusions $\mathbf{PprMA} \subseteq L_2^P \subseteq \mathbf{PprSBP}$ and $\mathbf{PprO}_2^P \subseteq O_2^P$, which not only give new upper and lower bounds for L_2^P , but also demonstrate that the Karp-Lipton-collapse to \mathbf{PprOMA} is currently the best one both for symmetric-alternation-based and Merlin-Arthur-based classes of languages.



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

1.3.1 A New Lower Bound for L_2^P and the Strongest Non-Input-Oblivious Karp–Lipton Collapse

Two open questions regarding symmetric alternation have been stated explicitly:

- whether \mathbf{PprMA} is contained in \mathbf{S}_2^P [15] (note that for these two classes Karp–Lipton style theorems have been proved by [15] and by Samik Sengupta [unpublished], respectively),
- whether a Karp–Lipton–style theorem holds for L_2^P [36].

In this paper we resolve both these questions affirmatively by showing the following containment. (Recall that $L_2^P \subseteq S_2^P$.)

► **Theorem 31.** $\mathbf{PprMA} \subseteq L_2^P$.

Combining this theorem with a result of Chakaravarthy and Roy [15] that $\mathbf{NP} \subseteq \mathbf{P/poly}$ implies the collapse $\mathbf{PH} = \mathbf{PprMA}$, we obtain a Karp–Lipton–style collapse theorem for L_2^P , thus resolving an open question posed in [36].

► **Corollary.** *If $\mathbf{NP} \subseteq \mathbf{P/poly}$, then $\mathbf{PH} = L_2^P = \mathbf{PprMA}$.*

Together with the result of [12], it lines up all known non-input-oblivious classes for which a Karp–Lipton–style collapse has been shown: $\mathbf{PprMA} \subseteq L_2^P \subseteq S_2^P \subseteq \mathbf{ZPP}^{\mathbf{NP}} \subseteq \Sigma_2^P$.

1.3.2 A New Upper Bound for L_2^P

Another important result of this work is a new upper bound on L_2^P : we prove that $L_2^P \subseteq \mathbf{PprSBP}$. The best known upper bound prior to our result followed from [36, 15]: $L_2^P \subseteq S_2^P \subseteq \mathbf{PprAM}$.

► **Theorem 22.** $L_2^P \subseteq \mathbf{PprSBP}$.

Our two new inclusions (Th. 31 and 22) yield the non-input-oblivious part of Fig. 1.

1.3.3 Aggregation of prO_2^P Queries and the Strongest Input-Oblivious Karp–Lipton Collapse

Th. 31 shows that \mathbf{PprMA} 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 O_2^P [14], which is input-oblivious. However, since the precise relationship between O_2^P and \mathbf{PprMA} remains unknown, one may ask: what is the strongest Karp–Lipton–style collapse? Our next result assists in navigating this question.

► **Theorem 25.** $\mathbf{P}^{\text{prO}_2^P} \subseteq \mathbf{O}_2^P$.

We note that the “non-promise” version of this inclusion, i.e. $\mathbf{P}^{\mathbf{O}_2^P} \subseteq \mathbf{O}_2^P$, was already established in [14]. 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 $\mathbf{P}^{\mathbf{S}_2^P} \subseteq \mathbf{S}_2^P$ [14], it still remains open whether $\mathbf{P}^{\text{prS}_2^P} \subseteq \mathbf{S}_2^P$.

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 $\mathbf{P}^{\text{prOMA}} \subseteq \mathbf{P}^{\text{prMA}}$, where **prOMA** is the input-oblivious version of **prMA** and therefore is contained in **prMA**. At the same time, Th. 25 allows us to show that $\mathbf{P}^{\text{prOMA}}$ is also contained in \mathbf{O}_2^P , thus making $\mathbf{P}^{\text{prOMA}}$ smaller than both \mathbf{P}^{prMA} and \mathbf{O}_2^P !

Th. 25 also helps us to tighten the chain of containments between $\mathbf{P}^{\text{prOMA}}$ and \mathbf{O}_2^P using a newly defined \mathbf{OL}_2^P class, an input-oblivious analogue of \mathbf{L}_2^P (see Fig. 1).

1.4 Our Techniques

1.4.1 SBP, $\mathbf{BPP}_{\text{path}}$, Approximate Counting, and Set Size Estimation

Computing the number of accepting paths of a given non-deterministic Turing machine is a fundamental problem captured by the “counting” class $\#\mathbf{P}$. Yet, this class appears to be too powerful since, by Toda’s Theorem [49], 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 \mathbf{NP} oracle ($\mathbf{FBPP}^{\mathbf{NP}}$) [48, 30] and by a deterministic algorithm using a **prAM** oracle ($\mathbf{FP}^{\text{prAM}}$) [47, 23]. Shaltiel and Umans [46] 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 $\mathbf{FBPP}_{\parallel}^{\mathbf{NP}}$, $\mathbf{FP}_{\parallel}^{\text{prAM}}$ and $\mathbf{FP}_{\parallel}^{\mathbf{NP}}$, respectively. Jeřábek studies approximate counting in the context of bounded arithmetic and reduces to it many problems associated with complexity classes [29]. Approximate counting has also recently attracted considerable attention in the quantum literature [41, 1, 2, 39].

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 **SBP** (strictly speaking for the corresponding class of promise problems **prSBP**) introduced in [7] 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 [7], 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 very different. Moreover, **SBP** remains the only known natural class that lies between **MA** and **AM**.

In terms of upper bounds, in a seminal paper [23], 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! (A non-deterministic circuit $C(x, w)$ accepts x if there exists a witness w for which $C(x, w) = 1$.) 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 12 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 [7] established an oracle separation between \mathbf{SBP} and \mathbf{AM} .

On a similar note, by combining some of the previous techniques, we observe that $\mathbf{APPROXCOUNT}$ can be carried out in $\mathbf{FP}^{\mathbf{prSBP}}$ rather than $\mathbf{FP}^{\mathbf{prAM}}$, and, in fact, even in $\mathbf{FP}_{\parallel}^{\mathbf{prSBP}}$. We describe the main idea in more details in Section 1.4.2. We defer the formal proof of this fact to the full version of the paper. Given this observation, it is natural to study the computational power of $\mathbf{P}^{\mathbf{APPROXCOUNT}}$, that is, deterministic algorithms with oracle access to $\mathbf{APPROXCOUNT}$. Indeed, an immediate corollary of the above is that $\mathbf{P}^{\mathbf{APPROXCOUNT}} = \mathbf{P}^{\mathbf{prSBP}}$ and $\mathbf{P}_{\parallel}^{\mathbf{APPROXCOUNT}} = \mathbf{P}_{\parallel}^{\mathbf{prSBP}}$. At the same, O’Donnell and Say [41] previously showed that $\mathbf{P}_{\parallel}^{\mathbf{APPROXCOUNT}} = \mathbf{BPP}_{\text{path}}$, a complexity class defined earlier by Han et al. [25]. One can think of $\mathbf{BPP}_{\text{path}}$ as a version of \mathbf{BPP} in which different computational paths (of the same probability) may have different lengths. Incidentally, it was established in [7] that $\mathbf{BPP}_{\text{path}}$ can be obtained from \mathbf{BPP} via the so-called “postselection” and that $\mathbf{SBP} \subseteq \mathbf{BPP}_{\text{path}}$ (and resp. $\mathbf{prSBP} \subseteq \mathbf{prBPP}_{\text{path}}$). Putting all together, one arrives at the following three clusters of complexity classes associated with approximate counting:

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

where both inclusions are believed to be strict.

1.4.2 Approximate Counting and the Order Rank Approximation

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

1.4.2.1 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 \mathbf{prSBP} (i.e. in $\mathbf{FP}^{\mathbf{prSBP}}$), using parallel queries. Our algorithm is based on \mathbf{SBP} amplification that was used in [7, 52]. 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 \mathbf{prSBP} oracle $O(n/\varepsilon)$ times in parallel or (using binary search) $O(\log_2(n/\varepsilon))$ times sequentially. This result (Lemma 15) could be of independent interest.

1.4.2.2 Approximating the rank w.r.t. a linear order

The *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 $\text{rank}(\alpha) = 0$). We can extend this definition to non-empty subsets $S \subseteq U$, where $\text{rank}(S)$ is the average rank of elements in S .

We reduce the problem of approximately comparing the average ranks of two sets to approximate counting. To see how, consider a strict linear order \preceq 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 rank of S is exactly the size of the set of pairs $\{(v, \alpha) \in U \times S \mid v \preceq \alpha\}$ divided by the size of S . Hence, this task can be carried out using a \mathbf{prSBP} oracle.

1.4.2.3 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 $\text{rank}(S) = \text{rank}(\alpha)/2$. We then iteratively partition S into two disjoint sets $S_0 = \{x \in S \mid x_i = 0\}$ and $S_1 = \{x \in S \mid x_i = 1\}$, starting from $i = 1$. By averaging argument, $\min\{\text{rank}(S_0), \text{rank}(S_1)\} \leq \text{rank}(S)$. We then take S to be the subset (S_0 or S_1) with the smaller 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 $\text{rank}(S) = \text{rank}(\beta)$. On the other hand, as the rank of the “initial” S was $\text{rank}(\alpha)/2$ and the overall rank could only decrease, we obtain that $\text{rank}(\beta) \leq \text{rank}(\alpha)/2$. We can then invoke the same procedure this time with β as its input. As 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 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^P

In [36], Korten and Pitassi show that $\mathbf{MA} \subseteq \mathbf{L}_2^P$. The inclusion $\mathbf{P}^{\text{prMA}} \subseteq \mathbf{L}_2^P$ essentially follows their argument with the additional observation that since \mathbf{L}_2^P is a syntactic class, not only it *contains* \mathbf{MA} but it is also *consistent* with prMA . Thus one can first construct a pseudorandom generator using an \mathbf{L}_2^P oracle [35, 36] and then leverage it to derandomize the prMA oracle not just in prNP , but actually in $\mathbf{NP} \subseteq \mathbf{L}_2^P$! Therefore, $\mathbf{P}^{\text{prMA}} \subseteq \mathbf{P}^{\mathbf{L}_2^P} = \mathbf{L}_2^P$.

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 possess all the desired properties w.r.t. Turing closure and natural containments. However, Th. 25 (its technique is described below) eventually helps us building the chain from $\mathbf{P}^{\text{prOMA}}$ to \mathbf{O}_2^P in two different ways: both directly and through derandomization and intermediate classes using \mathbf{OL}_2^P .

1.4.4 Input-Oblivious Symmetric Alternation

A Karp–Lipton–style collapse to $\mathbf{P}^{\text{prOMA}}$ follows from [15] by combining several previously known techniques. However, is this collapse stronger than the known collapse to \mathbf{O}_2^P [14]? The inclusion $\text{prOMA} \subseteq \text{prO}_2^P$ can be transferred from a somewhat similar statement that was proven in [14]; however, in order to prove $\mathbf{P}^{\text{prOMA}} \subseteq \mathbf{O}_2^P$ we need also the inclusion $\mathbf{P}^{\text{prO}_2^P} \subseteq \mathbf{O}_2^P$, 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 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 Discussion and Further Research

1.5.1 Connections to “hard” functions

Starting from Karp–Lipton’s paper [32], Kannan’s fixed-polynomial circuit complexity lower bounds [31] have been 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} containing \mathbf{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 [14] did not imply lower bounds for \mathbf{O}_2^P , because \mathbf{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 [22] building on recent progress for the range avoidance problem [35, 38], thus matching the progress on the two questions again.

Korten and Pitassi [36] 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 $\mathbf{P}^{\text{prMA}} \subseteq \mathbf{L}_2^P$ restores the balance.

However, the observation that in the input-oblivious world the currently best collapse $\mathbf{P}^{\text{prOMA}}$ reopens this question. Does this class possess fixed-polynomial circuit lower bounds? One can observe that Santhanam’s proof [44] of $\text{Size}[n^k]$ 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 $\mathbf{P}^{\text{prOMA}}$ that is hard for $\text{Size}[n^k]$, 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 $\varepsilon\text{-HARD-TT}$, asks given 1^{2^n} , to output a truth table of a function $\{0, 1\}^n \rightarrow \{0, 1\}$ of circuit complexity at least $2^{\varepsilon n}$ (Korten [35] defines a smoother version of it where the input length is not necessarily a power of two.) Can one prove a collapse to this class? Note that for the purpose of fixed-polynomial lower bounds even a limited version of $\varepsilon\text{-HARD-TT}$ suffices where the truth table is non-empty for a number of entries greater than any polynomial and its complexity is only superpolynomial.

Switching to the linear-exponential regime, in [36], Korten and Pitassi have shown that \mathbf{L}_2^E – the exponential version of \mathbf{L}_2^P – contains a language of circuit complexity $2^n/n$. By translation, our upper bound scales as $\mathbf{L}_2^E \subseteq \mathbf{E}^{\text{prSBP}} = \mathbf{E}^{\text{prBPP}_{\text{path}}}$. As a corollary we obtain a new circuit lower bound for $\mathbf{E}^{\text{prSBP}}$ (and hence for $\mathbf{E}^{\text{prBPP}_{\text{path}}}$). To the best of our knowledge, the strongest previously established bound for this class was “half-exponential” that followed from the bound on \mathbf{MAEXP} [40].

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

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

1.5.2 The smallest classes based on or containing Avoid

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 [33, 35] (and Korten [35] 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 L_2^P can be equivalently defined using many-one, Turing, or P^{NP} -reductions, thus there are several options. One can observe that the containment $P^{prMA} \subseteq L_2^P$ (Th. 31) is essentially proved via the intermediate class $P^{Avoid, NP}$ that uses both an oracle for *Range Avoidance* (a single-valued or an essentially unique [36] version) and an oracle for SAT. Can one prove that one of the containments in $P^{prMA} \subseteq P^{Avoid, NP} \subseteq L_2^P$ is in fact an equality?

One can define an input-oblivious version OL_2^P of L_2^P , however, contrary to $NP \subseteq L_2^P$, it is not obvious whether $ONP \subseteq OL_2^P$ or even $ONP \subseteq P^{prOL_2^P}$. Still $P^{OL_2^P} \subseteq O_2^P \cap L_2^P$. Nevertheless, one can extend the proof of $P^{prMA} \subseteq L_2^P$ to show $P^{prMA} \subseteq P^{OL_2^P, NP} \subseteq L_2^P$ and $P^{prOMA} \subseteq P^{OL_2^P, prONP} \subseteq O_2^P \cap L_2^P$. (These containments are included in Fig. 1, they give also an alternative proof of $P^{prOMA} \subseteq O_2^P$ that still uses Th. 25.)

1.5.3 Some Unresolved Containments

1. Can one strengthen our inclusion $L_2^P \subseteq P^{prSBP}$ to $S_2^P \subseteq P^{prSBP}$? One can try combining our techniques with the proof of $S_2^P \subseteq P^{prAM}$ by Chakaravarthy and Roy [15]. Note that in the other direction it is open even whether $SBP \subseteq S_2^P$.
2. Chakaravarthy and Roy [15] asked whether P^{prMA} and $P^{prS_2^P}$ are contained in 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 ($P^{prOMA} \subseteq P^{prO_2^P} \subseteq O_2^P$, Cor. 26), the proof of the latter inclusion (Th. 25) 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).
3. As was mentioned, the FP^{prSBP} procedure for approximate counting can be implemented in FP_{\parallel}^{prSBP} – that is, using parallel (i.e. non-adaptive) oracle queries. On the other hand the containment $L_2^P \subseteq P^{prSBP}$, 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 $L_2^P \subseteq P_{\parallel}^{prSBP}$)? In particular, as PP is consistent with $prSBP$ [7] and is closed under non-adaptive Turing reductions [20], this would imply that $L_2^P \subseteq PP$. Note that it is unknown even whether $P^{NP} \subseteq PP$, while $P^{NP} \subseteq L_2^P$. Moreover, there is an oracle separating the former two classes [6].
4. A recent work of Gajulapalli et al. [21] places L_2^P in the class $UEOPL^{NP}$ (where $UEOPL$ consists of problems that are many-one polynomial-time reducible to UNIQUE-END-OF-POTENTIAL-LINE, see [19, 21]), which appears to be incomparable to our result (Th. 22). Can one determine the relative status of $UEOPL^{NP}$ and P^{prSBP} ?

1.6 Organization of the Paper

The paper is organized as follows: In Section 2 we give the necessary definitions. Section 3 contains the proof of Th. 22 – a new upper bound on L_2^P . Section 4 contains the proof of Th. 25 which implies that collapse to P^{prOMA} subsumes both collapses to P^{prMA} and to O_2^P . In Section 5 we prove Th. 31 answering open questions of [36] and [15].

2 Definitions

2.1 Promise Classes as Oracles

A promise problem is a relaxation of (the decision problem for) a language.

► **Definition 1** (promise problem). $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is a promise problem if $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \emptyset$. We say that a language O is consistent with Π , if $\Pi_{\text{YES}} \subseteq O$ and $\Pi_{\text{NO}} \subseteq \bar{O}$.

52:10 Upper and Lower Bounds for LOP

Similarly to [15], 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.

2.2 Problems Avoid and LOP

► **Definition 2** (AVOID [33, 35]). AVOID is the following total search problem.

Input: circuit C with n inputs and $m > n$ outputs.

Output: $y \in \{0, 1\}^m \setminus \text{Im } C$.

This problem in a slightly different formulation is known for decades in the bounded arithmetic community under the name dWPHP (see [37] for survey).

Korten [35] proved that for a stretch of $n + 1$ this problem is equivalent to a stretch of $O(n)$ (and, of course, vice versa) using \mathbf{P}^{NP} reductions.

The following definition is due to Korten and Pitassi [36]. It extends the definition of the search problem MIN of [18] by the case where the input relation is not a linear order.

► **Definition 3** (LOP, Linear Ordering Principle). LOP is the following total search problem.

Input: ordering relation $<_{\varepsilon}$ given as a Boolean circuit E with $2n$ inputs.

Output: either the minimum for $<_{\varepsilon}$ (that is, x such that $\forall y \in \{0, 1\}^n \setminus \{x\} x <_{\varepsilon} y$) or a counterexample, if $<_{\varepsilon}$ is not a strict linear order. A counterexample is either a pair satisfying $x <_{\varepsilon} y <_{\varepsilon} x$ or a triple satisfying $x <_{\varepsilon} y <_{\varepsilon} z <_{\varepsilon} x$.

2.3 Complexity Classes

The following two definitions have been suggested by Korten and Pitassi who also proved their equivalence [36].

► **Definition 4** (\mathbf{L}_2^{P} via reductions). A language $L \in \mathbf{L}_2^{\text{P}}$ if it can be reduced to LOP using a \mathbf{P}^{NP} -Turing reduction. (Polynomial-time Turing reductions and polynomial-time many-one reductions have the same effect, as proved in [36].)

The following is an alternative definition of \mathbf{L}_2^{P} , which was shown in [36] to be equivalent.

► **Definition 5** (\mathbf{L}_2^{P} via symmetric alternation). A language $L \in \mathbf{L}_2^{\text{P}}$ if there is a ternary relation $R \subseteq \{0, 1\}^n \times \{0, 1\}^{s(n)} \times \{0, 1\}^{s(n)}$ computable in time $s(n)$, where s is a polynomial, 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 obvious that this version is a particular case of the definition of \mathbf{S}_2^{P} [13, 43]:

► **Definition 6.** A language $L \in \mathbf{S}_2^{\text{P}}$ if there is a polynomial-time computable 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} [14], 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 7.** A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ belongs to $\mathbf{prO}_2^{\mathbb{P}}$ 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:

- 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 behaviour of A is made outside the promise set except that it stops (accepts or rejects) in polynomial time.

$\mathbf{O}_2^{\mathbb{P}}$ 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 8.** A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ belongs to \mathbf{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:

- 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 $\mathbf{L}_2^{\mathbb{P}}$.

► **Definition 9.** A language L belongs to $\mathbf{OL}_2^{\mathbb{P}}$ 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 $V_x(u, v) = 1$),
- $\forall x \in L \cap \{0, 1\}^n \quad 1y_n = \min <_x$,
- $\forall x \in \bar{L} \cap \{0, 1\}^n \quad 0z_n = \min <_x$.

► **Remark 10.** Note that unlike $\mathbf{NP} \subseteq \mathbf{L}_2^{\mathbb{P}}$, it is unclear whether $\mathbf{ONP} \subseteq \mathbf{OL}_2^{\mathbb{P}}$. Therefore, when we need both these input-oblivious classes, we need to specify both oracles.

We also define \mathbf{SBP} for the sake of self-completeness.

► **Definition 11** ([7]). A language L is in \mathbf{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 $\mathbf{L}_2^{\mathbb{P}} \subseteq \mathbf{PprSBP}$

In this section we prove Th. 22. 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 \mathbf{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 rank.

Before we proceed with the main algorithm, we show how to approximate the size of a set using a \mathbf{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 with a **prSBP** oracle (in $\mathbf{FP}^{\mathbf{prSBP}}$). Previously, it was shown that this task could be carried out by a randomized algorithm with an **NP** oracle ($\mathbf{FBPP}^{\mathbf{NP}}$) [48, 30] and by a deterministic algorithm with a **prAM** oracle ($\mathbf{FP}^{\mathbf{prAM}}$) [47, 23]. Note that **prSBP/prAM** queries can be thought of as queries to specific promise problems.

► **Definition 12** (Set-Size Estimation, SSE and WSSE). *Let C be a Boolean circuit and $m \geq 1$ be an integer given in binary representation. Then $\text{SSE} := (\text{SSE}_{\text{YES}}, \text{SSE}_{\text{NO}})$, where $\text{SSE}_{\text{YES}} = \{(C, m) \mid \#_x C(x) \geq m\}$, $\text{SSE}_{\text{NO}} = \{(C, m) \mid \#_x C(x) \leq m/2\}$.*

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 proved essentially in [7] and [23] and formulated explicitly in [52].

► **Lemma 13** (Implicit in [7]). *SSE is **prSBP**-complete.*

► **Lemma 14** (Implicit in [23]). *WSSE is **prAM**-complete.*

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

► **Lemma 15.** *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 is considered “folklore”). We provide the proof in the full version of the paper.

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

Let $U = \{0, 1\}^n$. A single-output $2n$ -input Boolean circuit E induces an ordering relation \prec_E on U as $x \prec_E y \iff E(x, y) = 1$. If \prec_E is a strict linear order, we call E a *linear order circuit*.

► **Observation 16.** *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 $x \prec_E y \prec_E x$ or a triple satisfying $x \prec_E y \prec_E z \prec_E x$.*

Fix any strict linear order $<$ on U .

► **Definition 17.** *For an element $\alpha \in U$ we define its rank as $\text{rank}(\alpha) := |\{x \in U \mid x < \alpha\}|$. We can extend this definition to non-empty subsets $S \subseteq U$ of U by taking the average rank: define $\text{rank}(S) := \frac{\sum_{x \in S} \text{rank}(x)}{|S|}$. If $S = \{x \in U \mid C(x) = 1\}$ is described by a circuit C , we use the same notation: $\text{rank}(C) = \text{rank}(S)$.*

Below are some useful observations that we will use later.

► **Observation 18.**

- *For a non-empty subset $S \subseteq U$: $|\{(v, \alpha) \in U \times S \mid v < \alpha\}| = |S| \cdot \text{rank}(S)$.*
- *For any $\alpha \in U$: $\text{rank}\{v \in U \mid v \leq \alpha\} = \text{rank}(\alpha)/2$.*

- Let $S_0, S_1 \subseteq U$ be two non-empty disjoint subsets of U . Then

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

Proof.

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

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

► **Lemma 19.** *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 $4^{-\varepsilon} \cdot \text{rank}(C) \leq r \leq 4^\varepsilon \cdot \text{rank}(C)$ 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 $x \preceq y$. By Observation 18, $\#_{(x,y)} D = \#_x C \cdot \text{rank}(C)$. By Lemma 15 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: $4^{-\varepsilon} \cdot \text{rank}(C) \leq \frac{\#_{(x,y)} D}{4^\varepsilon \cdot \#_x C} \leq \frac{t_D}{t_C} \leq \frac{4^\varepsilon \cdot \#_{(x,y)} D}{\#_x C} = 4^\varepsilon \cdot \text{rank}(C)$. ◀

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 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 computing the bits of the new element, one at a time, using a **prSBP** oracle. The rank is w.r.t. the order \preceq described by a linear order circuit E .

► **Lemma 20.** *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 $\text{rank}(\beta) \leq \frac{\text{rank}(\alpha)}{\sqrt{2}}$, in time polynomial in n and the size of E , given oracle access to SSE.*

Proof. Consider the following procedure:

Back(E, α):

- Define $C(x) := E(x, \alpha) \vee x = \alpha$. // The set of all elements that are \preceq than or equal to α
- Set $\varepsilon = 1/(8n)$.
- For $i = 1$ to n :
 - For $b \in \{0, 1\}$: define $C_b := C|_{x_i=b}$
 - 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
 - For $b \in \{0, 1\}$: use Lemma 19 to approximate $\text{rank}(C_b)$ with ε into r_b
 - 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

52:14 Upper and Lower Bounds for LOP

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, \dots, x_i are replaced by the corresponding constants β_1, \dots, β_i . We claim that *after* each iteration the rank of the resulting circuit is bounded from the above: $\text{rank}(C) \leq 4^{2\epsilon i} \cdot \frac{\text{rank}(\alpha)}{2}$.

Indeed, by Observation 18, *before* the first iteration, we have that $\text{rank}(C) = \frac{\text{rank}(\alpha)}{2}$. Now consider any iteration. If C_1 or C_0 are empty, then $\text{rank}(C)$ remains the same and $4^{2\epsilon i} \leq 4^{2\epsilon(i+1)}$. Otherwise, by Lemma 19, for $b \in \{0, 1\}$, $4^{-\epsilon} \cdot \text{rank}(C_b) \leq r_b \leq 4^\epsilon \cdot \text{rank}(C_b)$. If $r_1 \geq r_0$ then $\text{rank}(C_1) \geq r_1 \cdot 4^{-\epsilon} \geq r_0 \cdot 4^{-\epsilon} \geq \text{rank}(C_0) \cdot 4^{-2\epsilon}$ and therefore by Observation 18: $\text{rank}(C) = \frac{\#_x C_0 \cdot \text{rank}(C_0) + \#_x C_1 \cdot \text{rank}(C_1)}{\#_x C_0 + \#_x C_1} \geq \frac{\#_x C_0 \cdot \text{rank}(C_0) + \#_x C_1 \cdot \text{rank}(C_0) \cdot 4^{-2\epsilon}}{\#_x C_0 + \#_x C_1} \geq \text{rank}(C_0) \cdot 4^{-2\epsilon}$.

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

Consequently, after the n -th iteration, C represents the set that contains only the element β and we have that $\text{rank}(\beta) \leq 4^{2\epsilon n} \cdot \text{rank}(\alpha)/2 \leq \sqrt{2} \cdot \text{rank}(\alpha)/2 = \text{rank}(\alpha)/\sqrt{2}$.

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

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

► **Observation 21.** $\text{BACK}(E, \alpha) = \alpha$ if and only if α is the minimal element in E .

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

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

► **Theorem 22.** $L_2^P \subseteq \text{PprSBP}$.

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

1. Check that E is indeed a linear order using Observation 16.
2. Let $\alpha := 0^n$, $\beta := 1^n$.
3. While $\alpha \neq \beta$ repeat: $\alpha := \beta$, $\beta := \text{BACK}(E, \alpha)$.
4. Output α .

Given Observation 16, 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 20, the rank of the element α after $2n$ iterations satisfies $\text{rank}(\alpha) \leq \frac{\text{rank}(1^n)}{\sqrt{2}^{2n}} \leq \frac{2^n - 1}{2^n} < 1$, thus the “While” cycle will terminate before that. ◀

4 Which Karp–Lipton–style Collapse is Better?

Chakaravorthy and Roy proved two Karp–Lipton–style collapses: down to \mathbf{O}_2^P [14] and down to \mathbf{PprMA} [15]. These two classes seem to be incomparable thereby rising the question: which collapse result is stronger? We observe that the collapse to \mathbf{PprMA} can actually be deepened to \mathbf{PprOMA} , where **prOMA** is the oblivious version of **prMA** – and subsequently show that latter class is contained in both previous classes. That is, $\mathbf{PprOMA} \subseteq \mathbf{PprMA} \cap \mathbf{O}_2^P$. 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 [3]. Nonetheless, we include a formal proof of this fact in the full version of the paper.

► **Proposition 23.** *If $\mathbf{NP} \subseteq \mathbf{P/poly}$, then $\mathbf{prAM} \subseteq \mathbf{prOMA}$ and $\mathbf{PH} = \mathbf{PprOMA}$.*

We prove that this class is not only included in $\mathbf{P}^{\mathbf{prMA}}$ but also in $\mathbf{O}_2^{\mathbf{P}}$ in two steps: (1) $\mathbf{prOMA} \subseteq \mathbf{prO}_2^{\mathbf{P}}$, (2) $\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$. The first inclusion is essentially proved in [14, Theorem 3], which says that $\mathbf{MA} \subseteq \mathbf{NO}_2^{\mathbf{P}}$. One needs to notice that the proof goes through for promise classes with all certificates being input-oblivious.

► **Proposition 24.** $\mathbf{prOMA} \subseteq \mathbf{prO}_2^{\mathbf{P}}$.

The second inclusion seems novel and we prove this result now:

► **Theorem 25.** $\mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}} \subseteq \mathbf{O}_2^{\mathbf{P}}$.

► **Remark.** Formally, we show that $\mathbf{P}^{\Pi} \subseteq \mathbf{O}_2^{\mathbf{P}}$ for every promise problem $\Pi \in \mathbf{prO}_2^{\mathbf{P}}$.

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 $\Pi \in \mathbf{prO}_2^{\mathbf{P}}$. For $n \in \mathbb{N}$, let $1, \dots, p(n)$ be all possible lengths of oracle queries made by M given an input of length n . Define $W_n := (w_1^{(0)}, \dots, w_{p(n)}^{(0)}, w_1^{(1)}, \dots, w_{p(n)}^{(1)})$ 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 $L \in \mathbf{O}_2^{\mathbf{P}}$ and will show that, for any x , the string $W_{|x|}$ constitutes an irrefutable certificate that can be used both as $U = (u_1^{(0)}, \dots, u_{p(n)}^{(0)}, u_1^{(1)}, \dots, u_{p(n)}^{(1)})$ and as $V = (v_1^{(0)}, \dots, v_{p(n)}^{(0)}, v_1^{(1)}, \dots, v_{p(n)}^{(1)})$.

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:

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

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 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 $b = d = 0 \implies \ell = 0$.
- 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 $a = c = 0 \implies \ell = 0$. ◀

► **Corollary 26.** $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{O}_2^{\mathbf{P}}$.

► **Remark 27.** When a semantic class without complete problems is used as an oracle, it may be ambiguous. However, \mathbf{prAM} does have a complete problem WSSE (see Definition 12). By inspecting the proof of the collapse (Prop. 23) one can observe that WSSE actually belongs to \mathbf{prOMA} under $\mathbf{NP} \subseteq \mathbf{P/poly}$, and the oracle Turing machine that demonstrates $\mathbf{PH} = \mathbf{P}^{\mathbf{prOMA}}$ still queries a specific promise problem.

For the inclusion of $\mathbf{P}^{\mathbf{prOMA}}$ in $\mathbf{O}_2^{\mathbf{P}}$, the first part (Prop. 24) transforms one promise problem into another promise problem, thus in the inclusion $\mathbf{P}^{\mathbf{prOMA}} \subseteq \mathbf{P}^{\mathbf{prO}_2^{\mathbf{P}}}$ it is also the case that a single oracle is replaced by (another) single oracle.

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

In this section we prove Th. 31 by, essentially, expanding the proof of $\mathbf{MA} \subseteq \mathbf{L}_2^{\text{P}}$ in [36]. We use the following statements from that paper and an earlier paper by Korten [35]. We then proceed to the input-oblivious setting.

5.1 The non-input-oblivious setting

► **Definition 28** ([35, Definitions 6, 7]). PRG is the following search problem: given 1^n , output a pseudorandom generator $R = (x_1, \dots, x_m)$, that is, an array of strings $x_i \in \{0, 1\}^n$ such that for every n -input circuit C of size n , $|\Pr_{x \leftarrow U(R)}\{C(x) = 1\} - \Pr_{y \leftarrow U(\{0,1\}^n)}\{C(y) = 1\}| \leq \frac{1}{n}$.

Korten proves that such a generator containing $m = n^6$ strings can be constructed with a single oracle query to AVOID (he refer to it as “EMPTY”), and Korten and Pitassi demonstrate that AVOID (which they call WEAK AVOID) can be solved with one oracle query to LOP.

► **Proposition 29** ([35, Th. 2]). PRG reduces in polynomial time to a single AVOID query.

► **Proposition 30** ([36, Th. 1]). AVOID is polynomial-time many-one reducible to LOP.

The main result of this section is the following theorem.

► **Theorem 31.** $\mathbf{P}^{\text{prMA}} \subseteq \mathbf{L}_2^{\text{P}}$.

Proof. We show how to replace prMA by \mathbf{L}_2^{P} . Since $\mathbf{P}^{\mathbf{L}_2^{\text{P}}} = \mathbf{L}_2^{\text{P}}$ by Def. 4, 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^{P} in order to build a pseudorandom generator sufficient to derandomize circuits of size $s(n)$. By Prop. 29, such a pseudorandom generator G , which is a sequence $G(1^{s(n)})$ of pseudorandom strings $g_1, \dots, g_m \in \{0, 1\}^{s(n)}$ for m bounded by a polynomial in $s(n)$, can be constructed (for $m = s(n)^6$ and error $\frac{1}{s(n)}$) using a reduction to AVOID. Subsequently, by Prop. 30, 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^{P} oracle.

After G is computed, each call to the prMA oracle can be replaced by an $\mathbf{NP} \subseteq \mathbf{L}_2^{\text{P}}$ 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 \mathbf{NP} queries even if Arthur does not conform to the definition of \mathbf{MA} in this case. These oracle answers are irrelevant, because the original \mathbf{P}^{prMA} machine must return the correct (in particular, the same) answer irrespectively of the oracle’s answer. ◀

5.2 The input-oblivious setting

Korten proves (Prop. 29) that PRG (Def. 28) reduces to AVOID, and Korten and Pitassi [36] compute AVOID in $\mathbf{L}_2^{\text{P}} = \mathbf{P}^{\mathbf{L}_2^{\text{P}}}$. 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.

► **Lemma 32.** PRG can be computed in deterministic polynomial time with an \mathbf{OL}_2^{P} oracle.

► **Corollary 33.**

1. $\mathbf{P}^{\text{prMA}} \subseteq \mathbf{P}^{\text{OL}_2^{\text{P}}, \text{NP}} \subseteq \mathbf{L}_2^{\text{P}}$.
2. $\mathbf{P}^{\text{prOMA}} \subseteq \mathbf{P}^{\text{OL}_2^{\text{P}}, \text{prONP}} \subseteq \mathbf{O}_2^{\text{P}} \cap \mathbf{L}_2^{\text{P}}$.

— **References** —

- 1 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. doi:10.4230/LIPICs.CCC.2020.7.
- 2 S. Aaronson and P. Rall. Quantum approximate counting, simplified. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA 2020*, pages 24–32. SIAM, 2020. doi:10.1137/1.9781611976014.5.
- 3 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. doi:10.1016/0304-3975(95)91133-B.
- 4 B. Aydinlioglu, 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. doi:10.1007/s00037-011-0010-8.
- 5 L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. doi:10.1007/BF01200056.
- 6 R. Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994. doi:10.1007/BF01263422.
- 7 E. Böhler, C. Glaßer, and D. Meister. Error-bounded probabilistic computations between MA and AM. *J. Comput. Syst. Sci.*, 72(6):1043–1076, 2006. doi:10.1016/J.JCSS.2006.05.001.
- 8 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. doi:10.1006/JCSS.1996.0032.
- 9 H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In *STACS*, pages 100–109, 1999.
- 10 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.
- 11 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. doi:10.1007/3-540-56287-7_99.
- 12 J.-Y. Cai. $S_2P \subseteq ZPP^{NP}$. *Journal of Computer and System Sciences*, 73(1):25–35, 2007.
- 13 R. Canetti. More on BPP and the polynomial-time hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996. doi:10.1016/0020-0190(96)00016-6.
- 14 V. T. Chakaravarty and S. Roy. Oblivious symmetric alternation. In *STACS*, pages 230–241, 2006.
- 15 V. T. Chakaravarty and S. Roy. Arthur and Merlin as oracles. *Comput. Complex.*, 20(3):505–558, 2011. doi:10.1007/s00037-011-0015-3.
- 16 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.
- 17 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.
- 18 M. Chiari and J. Krajíček. Witnessing functions in bounded arithmetic and search problems. *The Journal of Symbolic Logic*, 63(3):1095–1115, 1998. doi:10.2307/2586729.
- 19 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020. doi:10.1016/J.JCSS.2020.05.007.

- 20 L. Fortnow and N. Reingold. PP is closed under truth-table reductions. *Inf. Comput.*, 124(1):1–6, 1996. doi:10.1006/inco.1996.0001.
- 21 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. URL: <https://eccc.weizmann.ac.il/report/2025/121>.
- 22 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. doi:10.4230/LIPICs.FSTTCS.2024.23.
- 23 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.
- 24 J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988. doi:10.1137/0217018.
- 25 Y. Han, L. A. Hemaspaandra, and T. Thierauf. Threshold computation and cryptographic security. *SIAM J. Comput.*, 26(1):59–78, 1997. doi:10.1137/S0097539792240467.
- 26 R. Impagliazzo, V. Kabanets, and I. Volkovich. Synergy between circuit obfuscation and circuit minimization. In *APPROX/RANDOM 2023*, volume 275 of *LIPICs*, pages 31:1–31:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.APPROX/RANDOM.2023.31.
- 27 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. doi:10.1016/S0022-0000(02)00024-7.
- 28 E. Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129:1–37, 2004. doi:10.1016/J.APAL.2003.12.003.
- 29 E. Jeřábek. Approximate counting in bounded arithmetic. *Journal of Symbolic Logic*, 72(3):959–993, 2007. doi:10.2178/JSL/1191333850.
- 30 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. doi:10.1016/0304-3975(86)90174-X.
- 31 R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982. doi:10.1016/S0019-9958(82)90382-5.
- 32 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. doi:10.1145/800141.804678.
- 33 R. Kleinberg, O. Korten, D. Mitropolsky, and C. Papadimitriou. Total Functions in the Polynomial Hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 44:1–44:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.ITCS.2021.44.
- 34 J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998. doi:10.1137/S0097539795296206.
- 35 O. Korten. The hardest explicit construction. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 433–444. IEEE, 2022.
- 36 O. Korten and T. Pitassi. Strong vs. Weak Range Avoidance and the Linear Ordering Principle. *Electron. Colloquium Comput. Complex.*, TR24-076, 2024. URL: <https://eccc.weizmann.ac.il/report/2024/076>.
- 37 J. Krajíček. *Proof Complexity Generators*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2025.
- 38 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. doi:10.1145/3618260.3649615.

- 39 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. doi:10.4230/LIPICs.CCC.2025.5.
- 40 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.
- 41 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. doi:10.1145/3196832.
- 42 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. doi:10.1017/S0022481200028061.
- 43 A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Comput. Complex.*, 7(2):152–162, 1998. doi:10.1007/s000370050007.
- 44 R. Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009. doi:10.1137/070702680.
- 45 U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983. doi:10.1016/0022-0000(83)90027-2.
- 46 R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. *Comput. Complex.*, 15(4):298–341, 2006. doi:10.1007/s00037-007-0218-9.
- 47 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. doi:10.1145/800061.808762.
- 48 L. J. Stockmeyer. On approximation algorithms for #P. *SIAM J. Comput.*, 14(4):849–861, 1985. doi:10.1137/0214060.
- 49 S. Toda. PP is as hard as the polynomial time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. doi:10.1137/0220053.
- 50 N. V. Vinodchandran. A note on the circuit complexity of PP. *Theor. Comput. Sci.*, 347(1-2):415–418, 2005. doi:10.1016/j.tcs.2005.07.032.
- 51 I. Volkovich. On learning, lower bounds and (un)keeping promises. In *Proceedings of the 41st ICALP*, pages 1027–1038, 2014.
- 52 I. Volkovich. The untold story of SBP. In Henning Fernau, editor, *The 15th International Computer Science Symposium in Russia, CSR*, volume 12159 of *Lecture Notes in Computer Science*, pages 393–405. Springer, 2020. doi:10.1007/978-3-030-50026-9_29.