Bounded Relativization

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan

Zhenjian Lu
University of Oxford, UK

Hanlin Ren
University of Oxford, UK

Abstract

Relativization is one of the most fundamental concepts in complexity theory, which explains the difficulty of resolving major open problems. In this paper, we propose a weaker notion of relativization called *bounded relativization*. For a complexity class $\mathfrak{C}$, we say that a statement is $\mathfrak{C}$-*relativizing* if the statement holds relative to every oracle $\mathcal{O} \in \mathfrak{C}$. It is easy to see that every result that relativizes also $\mathfrak{C}$-relativizes for every complexity class $\mathfrak{C}$. On the other hand, we observe that many non-relativizing results, such as $\mathsf{IP} = \mathsf{PSPACE}$, are in fact PSPACE-relativizing.

First, we use the idea of bounded relativization to obtain new lower bound results, including the following nearly maximum circuit lower bound: for every constant $\varepsilon > 0$,

$$\mathsf{BPE}^{\mathrm{MCSP}}/_{2^{\varepsilon n}} \not\subseteq \mathsf{SIZE}[2^n/n].$$

We prove this by PSPACE-relativizing the recent pseudodeterministic pseudorandom generator by Lu, Oliveira, and Santhanam (STOC 2021).

Next, we study the limitations of PSPACE-relativizing proof techniques, and show that a seemingly minor improvement over the known results using PSPACE-relativizing techniques would imply a breakthrough separation $\mathsf{NP} \neq \mathsf{L}$. For example:

- Impagliazzo and Wigderson (JCSS 2001) proved that if $\mathsf{EXP} \neq \mathsf{BPP}$, then BPP admits infinitely-often subexponential-time heuristic derandomization. We show that their result is PSPACE-relativizing, and that improving it to worst-case derandomization using PSPACE-relativizing techniques implies $\mathsf{NP} \neq \mathsf{L}$.
- Oliveira and Santhanam (STOC 2017) recently proved that every dense subset in P admits an infinitely-often subexponential-time pseudodeterministic construction, which we observe is PSPACE-relativizing. Improving this to almost-everywhere (pseudodeterministic) or (infinitely-often) deterministic constructions by PSPACE-relativizing techniques implies $\mathsf{NP} \neq \mathsf{L}$.
- Santhanam (SICOMP 2009) proved that pr-MA does not have fixed polynomial-size circuits. This lower bound can be shown PSPACE-relativizing, and we show that improving it to an almost-everywhere lower bound using PSPACE-relativizing techniques implies $\mathsf{NP} \neq \mathsf{L}$.

In fact, we show that if we can use PSPACE-relativizing techniques to obtain the above-mentioned improvements, then $\mathsf{PSPACE} \neq \mathsf{EXPH}$. We obtain our barrier results by constructing suitable oracles computable in EXPH relative to which these improvements are impossible.

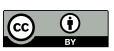
2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Oracles and decision trees; Theory of computation → Circuit complexity; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases relativization, circuit lower bound, derandomization, explicit construction, pseudodeterministic algorithms, interactive proofs

Digital Object Identifier 10.4230/LIPIcs.CCC.2023.6

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2023/070/` [37]

Funding *Shuichi Hirahara*: Supported by JST, PRESTO Grant Number JPMJPR2024, Japan.
Hanlin Ren: Received support from DIMACS through grant number CCF-1836666 from the National Science Foundation.



Acknowledgements We thank Rahul Santhanam for helpful discussions and for pointing out that the oracle in Theorem 52 can be constructed in EXPH, using [31] only as a black-box. We thank Lijie Chen for helpful discussions regarding [19], Ian Mertz for discussions about the statement (*), and Ryan Williams for useful discussions about time-space tradeoffs for SAT. We thank Lijie Chen (again) and an anonymous CCC reviewer for pointing out an error in a previous version of this paper. Part of this work was completed when the authors are visiting the Simons Institute for the Theory of Computing, participating in the *Meta-Complexity* program.

1 Introduction

The *relativization* barrier, introduced by Baker, Gill, and Solovay [11], is an influential meta-mathematical barrier in complexity theory. Techniques based on simulation and diagonalization tend to *relativize*, in the sense that they also work in an "oracle world" where every machine has access to an oracle $\mathcal{O}$. On the other hand, the P vs. NP question cannot be solved in a relativizing way: [11] showed that there exists an oracle $\mathcal{A}$ such that $\mathsf{P}^{\mathcal{A}} = \mathsf{NP}^{\mathcal{A}}$, as well as another oracle $\mathcal{B}$ such that $\mathsf{P}^{\mathcal{B}} \neq \mathsf{NP}^{\mathcal{B}}$. Relativization has been successful not only in explaining the difficulty of resolving the P vs. NP question, but also in pinning down the *exact* place where we are stuck – for example, there are oracles relative to which $\mathsf{BPP} = \mathsf{EXP}^{\mathsf{NP}}$ [32, 14], or $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{SIZE}[O(n)]$ [80].

In the early 1990s, the interactive proof results such as $\mathsf{IP} = \mathsf{PSPACE}$ [55, 66] generated much excitement among complexity theorists, as they are the first examples of "truly compelling" [4] non-relativizing results in complexity theory. Indeed, $\mathsf{coNP}^{\mathcal{O}} \not\subseteq \mathsf{IP}^{\mathcal{O}}$ relative to a random oracle $\mathcal{O}$ [28, 16]. The $\mathsf{IP} = \mathsf{PSPACE}$ result and its underlying technique, *arithmetization*, has significantly expanded our knowledge about circuit lower bounds [13, 40, 73, 1, 65, 27], derandomization [42, 70, 20, 21, 19], Karp–Lipton theorems [55, 10, 39, 18], meta-complexity [5, 61, 39, 60, 54], and other areas.

Still, it seems that arithmetization alone would not suffice to resolve the P vs. NP question. How far can we push these techniques? As relativization does not capture the "current techniques" anymore [16, 13, 1], what are the other barriers preventing us from making progress?

Aaronson and Wigderson [3] proposed the *algebrization* barrier to capture the limitations of arithmetization. Roughly speaking, an inclusion $\mathfrak{C} \subseteq \mathfrak{D}$ *algebrizes* if $\mathfrak{C}^{\mathcal{O}} \subseteq \mathfrak{D}^{\widetilde{\mathcal{O}}}$ for every oracle $\mathcal{O}$ and every low-degree extension $\widetilde{\mathcal{O}}$ of $\mathcal{O}$. This framework captures most results proved using arithmetization, such as $\mathsf{IP} = \mathsf{PSPACE}$ [66] and $\mathsf{MIP} = \mathsf{NEXP}$ [9].[1] On the other hand, [3] constructed oracles $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{O}_3$ such that $\mathsf{NP}^{\widetilde{\mathcal{O}_1}} \subseteq \mathsf{P}^{\mathcal{O}_1}$, $\mathsf{NEXP}^{\widetilde{\mathcal{O}_2}} \subseteq \mathsf{P}^{\mathcal{O}_2}/_{\mathrm{poly}}$, and $\mathsf{RP}^{\mathcal{O}_3} \not\subseteq \mathsf{P}^{\widetilde{\mathcal{O}_3}}$. Therefore, techniques based on arithmetization are not enough for proving long-standing conjectures in complexity theory such as $\mathsf{P} \neq \mathsf{NP}$, $\mathsf{NEXP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$, and $\mathsf{RP} = \mathsf{P}$.

It turns out that the algebrization barrier suffers from subtleties. For example, it was unclear how to formalize algebrization for complexity-theoretic statements that are neither separation nor inclusion; moreover, algebrization is not closed under *modus ponens*. Impagliazzo, Kabanets, and Kolokolova [38] and Aydınlıoğlu and Bach [8] revised the definition of algebrization to fix these issues. For more discussion of algebrization, see Section 1.4.

[1] A subtle issue on relativizing NEXP is that we need to restrict the NEXP machine to only query $\mathcal{O}$ on polynomially-long inputs. In other words, while we can show that $\mathsf{NEXP}^{\mathcal{O}[\mathrm{poly}]} \subseteq \mathsf{MIP}^{\widetilde{\mathcal{O}}}$ for every oracle $\mathcal{O}$ and its low-degree extension $\widetilde{\mathcal{O}}$, the statement $\mathsf{NEXP}^{\mathcal{O}} \subseteq \mathsf{MIP}^{\widetilde{\mathcal{O}}}$ is false in general. We refer the readers to Remark 2 for more details on relativizing space-bounded and exponential-time classes.

Unfortunately, algebrization did not become as popular as relativization in proving barrier results and predicting the limitations of "current techniques". Perhaps one reason is that algebrization barriers are harder to demonstrate, as one needs to construct an oracle $\mathcal{O}$ that diagonalizes against its low-degree extension $\widetilde{\mathcal{O}}$.[2] For many complexity theoretic statements that are slightly more complicated than inclusions ("$\mathfrak{C} \subseteq \mathfrak{D}$") or separations ("$\mathfrak{C} \not\subseteq \mathfrak{D}$"), it appears much harder to demonstrate algebrization barriers than relativization barriers.

1.1 Bounded Relativization

This paper takes one step back and considers relativization in the presence of non-relativizing techniques. Our main message is perhaps surprising: the shadow of the relativization barrier has *never* gone away, even though we already have powerful non-relativizing techniques at hand!

Specifically, we put forward a notion of *bounded relativization*. Roughly speaking, for a complexity class $\mathfrak{C}$, a complexity-theoretic statement is $\mathfrak{C}$-*relativizing* if it is true relative to every oracle $\mathcal{O} \in \mathfrak{C}$. It is easy to see that every statement that relativizes also $\mathfrak{C}$-relativizes, for every complexity class $\mathfrak{C}$. On the other hand, even though $\mathsf{IP} = \mathsf{PSPACE}$ is not relativizing, it is easily seen to be PSPACE-relativizing:

▶ **Proposition 1.** *For every oracle* $\mathcal{O} \in \mathsf{PSPACE}$, $\mathsf{IP}^{\mathcal{O}} = \mathsf{PSPACE}^{\mathcal{O}}$.

Proof. Since the proof of $\mathsf{IP} \subseteq \mathsf{PSPACE}$ is relativizing, we have $\mathsf{IP}^{\mathcal{O}} \subseteq \mathsf{PSPACE}^{\mathcal{O}}$.

On the other hand, since $\mathcal{O} \in \mathsf{PSPACE}$, we have $\mathsf{PSPACE}^{\mathcal{O}} = \mathsf{PSPACE} = \mathsf{IP} \subseteq \mathsf{IP}^{\mathcal{O}}$. ◀

▶ Remark 2. In this paper, when we relativize space-bounded machines (in particular PSPACE), we assume the query tape is counted into the space bound. That is, a $\mathsf{PSPACE}^{\mathcal{O}}$ machine can only query $\mathcal{O}$ on polynomially-long inputs. It is easy to see that under this definition, $\mathsf{PSPACE}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$.

On the other hand, we allow $\mathsf{EXP}^{\mathcal{O}}$ or $\mathsf{NEXP}^{\mathcal{O}}$ machines to issue exponentially-long queries to $\mathcal{O}$. When we restrict the query length to be a polynomial, we write $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]}$ and $\mathsf{NEXP}^{\mathcal{O}[\mathrm{poly}]}$ instead.

How is Proposition 1 helpful in studying the limitation of $\mathsf{IP} = \mathsf{PSPACE}$ as a technique? Say that we want to prove "a slight improvement of [42]" using "current techniques". (For now, the exact meaning of "a slight improvement of [42]" is not important; a concrete example will be given in Section 1.3.1. We also do not formally define "current techniques" here.) Suppose we construct an oracle $\mathcal{O}$, *in the usual, Baker–Gill–Solovay sense of relativization*, relative to which the "slight improvement of [42]" is impossible (see, e.g., Theorem 10).

- Suppose, in addition, that $\mathcal{O} \in \mathsf{PSPACE}$. It follows immediately that, if the term "current techniques" is interpreted as "PSPACE-relativizing techniques", then "current techniques" cannot prove the desired "slight improvement".
- In reality, it is often the case that we do not know how to put $\mathcal{O}$ in PSPACE; however, we can still show that $\mathcal{O} \in \mathsf{EXPH}$ in many cases. Then, the oracle $\mathcal{O}$ tells us that, if there is a PSPACE-relativizing proof of the "slight improvement of [42]", then this proof also implies $\mathsf{PSPACE} \neq \mathsf{EXPH}$. The latter would be a breakthrough in complexity theory, and in particular, it implies $\mathsf{L} \neq \mathsf{NP}$.

[2] Or, one needs to make sure the oracle $\mathcal{O}$ satisfies the so-called *Arithmetic Checkability Theorem* in the sense of [38].

To summarize, any EXPH-computable oracle presents the following barrier to "current techniques":

> If "current techniques" are PSPACE-relativizing and cannot separate L from NP, then "current techniques" also cannot prove the "slight improvement of [42]" (or any statement that is not EXPH-relativizing).

How "relativizing" are interactive proof results? The above discussion suggests that IP = PSPACE is actually "mildly relativizing" in the sense that it requires significant computational power (namely, super-polynomial space) to create an oracle world in which it is false. Using the same reasoning as Proposition 1, we can see that other interactive proof results, such as MIP = NEXP [9] and $\mathsf{MIP}^* = \mathsf{RE}$ [44], are also "mildly relativizing".

▶ **Proposition 3.** *The following are true:*

- **(Relativization of** MIP = NEXP**)** *For every* $\mathcal{O} \in \mathsf{NEXP} \cap \mathsf{coNEXP}$, $\mathsf{MIP}^{\mathcal{O}} = \mathsf{NEXP}^{\mathcal{O}[\mathrm{poly}]}$.
- **(Relativization of** $\mathsf{MIP}^* = \mathsf{RE}$**)** *For every* $\mathcal{O} \in \mathsf{R}$ *(i.e.,* $\mathcal{O}$ *is computable),* $(\mathsf{MIP}^*)^{\mathcal{O}} = \mathsf{RE}^{\mathcal{O}}$.

Proof. One direction relativizes: $\mathsf{MIP}^{\mathcal{O}} \subseteq \mathsf{NEXP}^{\mathcal{O}[\mathrm{poly}]}$, and $(\mathsf{MIP}^*)^{\mathcal{O}} \subseteq \mathsf{RE}^{\mathcal{O}}$. For the other direction, notice that $\mathsf{NEXP}^{\mathcal{O}[\mathrm{poly}]} \subseteq \mathsf{NEXP} = \mathsf{MIP} \subseteq \mathsf{MIP}^{\mathcal{O}}$ and $\mathsf{RE}^{\mathcal{O}} \subseteq \mathsf{RE} = \mathsf{MIP}^* \subseteq (\mathsf{MIP}^*)^{\mathcal{O}}$. ◀

Consequently, many results proved by combining interactive proof techniques and other relativizing techniques are also $\mathfrak{C}$-relativizing for a large complexity class $\mathfrak{C}$.[3] (It seems fair to say that most of them are PSPACE-relativizing.) This paper will explore the following two directions:

1. The positive direction: we relativize some previous results with an oracle $\mathcal{O} \in \mathsf{PSPACE}$ to obtain new results.
2. The negative direction: we construct oracle worlds in EXPH and demonstrate that certain seemingly minor improvements over known results might be hard to prove by PSPACE-relativizing techniques.

1.2 New Lower and Upper Bounds via Bounded Relativization

We first explain our positive results. Lu, Oliveira, and Santhanam [54] constructed a pseudodeterministic pseudorandom generator (PRG) with sub-polynomial seed length and one bit of advice that infinitely-often fools uniform algorithms. That is, for every $\varepsilon > 0$, they constructed a function $G : \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ computable in randomized polynomial time with one bit of advice, such that for every polynomial-time algorithm $\mathcal{A}$, G fools $\mathcal{A}$ on infinitely many input lengths. The term "pseudodeterministic" means that the randomized algorithm for G, on input $z \in \{0,1\}^{n^\varepsilon}$, will output the *fixed* string $G(z)$ with high probability.

Although the PRG construction in [54] is not relativizing, we observe that it is still PSPACE-relativizing. We exploit this fact to prove new circuit lower bounds and design new pseudodeterministic algorithms for the range avoidance problem.

[3] The fact that these results are $\mathfrak{C}$-relativizing might not be as obvious as Proposition 3, as one still needs to look into the proof of these results and replace the usage of interactive proof results by their bounded-relativizing counterparts. For example, using IP = PSPACE, Santhanam [65] showed that there is a problem in pr-MA without size-n^{100} circuits. To show this result is PSPACE-relativizing, one apparently needs to follow the proof of [65] and replace all occurrences of "IP = PSPACE" by Proposition 1.

A Nearly Maximum Circuit Lower Bound. Our first result shows that $\mathsf{BPE}^{\mathrm{MCSP}}/_{2^{\varepsilon n}}$ requires circuits of nearly maximum size. Here, MCSP is the *minimum circuit size problem* [46] that takes as input the length-N truth table of a function $f : \{0,1\}^{\log N} \to \{0,1\}$ and a size parameter s, and decides whether f can be computed by a size-s circuit.

▶ **Theorem 4.** *For every constant $\varepsilon > 0$, $\mathsf{BPE}^{\mathrm{MCSP}}/_{2^{\varepsilon n}}$ cannot be computed by circuits of size $2^n/n$.*

Note that every function can be computed by a circuit of size $(1 + o(1))2^n/n$ [56, 29]. Also, it is trivial that $\mathsf{E}/_{2^n}$ contains a language with maximum circuit complexity. For comparison, the advice length in our lower bound in Theorem 4 is only $2^{\varepsilon n}$, for an arbitrarily small constant $\varepsilon > 0$. Previously, the smallest complexity class known to require maximum circuit complexity is $\mathsf{E}^{\Sigma_2^p}$ [57]. It was proved in [39] that $\mathsf{ZPEXP}^{\mathrm{MCSP}}$ does not have polynomial-size circuits, but an exponential-size lower bound for this class is open.

Proof Idea of Theorem 4. Theorem 4 is proved by PSPACE-relativizing the PRG construction in [54]. Let $\mathcal{O} := \mathrm{MCSP} \in \mathsf{PSPACE}$. We can relativize the PRG in [54] with the MCSP oracle and obtain:

> For every $\varepsilon > 0$, there is a function $G : \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ computable in randomized polynomial time *with access to an* MCSP *oracle* and one bit of advice, such that for every polynomial-time algorithm $\mathcal{A}$ *with access to an* MCSP *oracle*, G fools $\mathcal{A}$ on infinitely many input lengths.

Let $\mathcal{A}$ be the following algorithm: given the truth table of a function $f : \{0,1\}^{\log N} \to \{0,1\}$, use the MCSP oracle to determine if the circuit complexity of f is at least $2^n/n$ (where $n := \log N$), and outputs 1 if and only if this is the case. Since most truth tables have circuit complexity at least $2^n/n$ [67, 56, 29], and $\mathcal{A}$ is fooled by the PRG G, it follows that there is a seed $z \in \{0,1\}^{N^{\varepsilon/2}}$ such that the function whose truth table is $G(z)$ also has circuit complexity at least $2^n/n$.

Consider the following language L. On input $x \in \{0,1\}^n$, it takes advice (z, α), where $z \in \{0,1\}^{2^{\varepsilon n/2}}$ is defined as above, and $\alpha \in \{0,1\}$ is the one-bit advice used by G. It computes the truth table $G(z)$ and accepts the input x if and only if the x-th bit of $G(z)$ is 1. Then L does not have circuits of size $2^n/n$. On the other hand, L can be computed by a $\mathsf{BPE}^{\mathrm{MCSP}}$ machine taking $2^{\varepsilon n}$ bits of advice. ◀

Besides the nearly-maximum circuit lower bound, we also prove new circuit lower bounds for meta-complexity problems and design new algorithms for range avoidance. We omit the proof ideas below since the main idea is essentially the same, namely relativizing the PRG in [54] to the MCSP oracle.

Circuit Lower Bounds for Meta-Complexity Problems. *Meta-computational* problems play a central role in a recently emerging area of research called *meta-complexity*, and have diverse applications in complexity theory, cryptography and learning. Roughly speaking, meta-computational problems are those that ask about the complexity (e.g., circuit complexity, time-bounded Kolmogorov complexity) of their inputs, and meta-complexity refers to the complexity of computing these problems themselves.

Before stating our result, we first recall some basic definitions. For a string x and a time bound function t, $\mathrm{K}^t(x)$, the *t-time-bounded Kolmogorov complexity* of x, is the minimum length of a string d such that $U(d)$ outputs x within $t(|x|)$ steps, where U is a time-optimal universal Turing machine fixed in advance. $\mathrm{rK}^t(x)$, the *randomized t-time-bounded*

Kolmogorov complexity of x, is defined in the same way as $\mathrm{K}^t(x)$ except that U in this case is a *randomized* universal Turing machine and we want x to be outputted with probability at least $2/3$. We can also define oracle versions of these complexity measures. For example, $\mathrm{rK}^{t,\mathcal{O}}$ can be defined in the same way as rK^t except that the universal Turing machine in this case has access to the oracle $\mathcal{O}$. (See Definition 21 for the formal definitions.)

Previously [35] showed that the problem of computing the $\mathrm{K}^{\mathrm{poly}}$ complexity of a given string is hard against fixed-polynomial-time deterministic algorithms. Later [54] showed a similar lower bound for computing $\mathrm{rK}^{\mathrm{poly}}$ but against fixed-polynomial-time randomized algorithms. Here, we show that computing $\mathrm{rK}^{\mathrm{poly},\mathrm{MCSP}}$ cannot be done using fixed-polynomial-size *circuits*.

▶ **Theorem 5** (Informal; see Theorem 32)**.** *For every $k \geq 1$, there is a polynomial t such that the problem of computing the $\mathrm{rK}^{t,\mathrm{MCSP}}$ complexity of a given string cannot be done using circuits of size n^k.*

In proving Theorem 5, we construct an efficient pseudodeterministic PRG, using MCSP oracle and a short advice string, that can fool polynomial-size *circuits*, which may be of independent interest. (See Theorem 33.)

Pseudodeterministic Construction for Range Avoidance. In the range avoidance problem, given a circuit $C\colon \{0,1\}^n \to \{0,1\}^{n+1}$, we are asked to find a string $x \in \{0,1\}^{n+1}$ that is not in the range of C. This problem is complete for the class called APEPP that corresponds to explicit constructions of objects whose existence can be shown using the probabilistic method [49, 51, 64]. It is open whether there is a deterministic polynomial-time algorithm with access to an NP oracle that solves the range avoidance problem; indeed, this open question is equivalent to the circuit lower bound $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}[2^{n/2}]$ [51].

Here, we present a new algorithm that *pseudodeterministically* solves the range avoidance problem for polynomial-size circuits. A pseudodeterministic algorithm is a probabilistic algorithm that "behaves like a deterministic algorithm" in the sense that it returns a *fixed* output with high probability over its internal randomness. Our algorithm runs in polynomial time, using an NP oracle and an advice of length n^ε (that is independent of the input circuit), and it works for infinitely many n, where n is the input length of the given circuit.

▶ **Theorem 6** (Informal; see Theorem 36)**.** *For every $\varepsilon > 0$ and $c \geq 1$, there exists a polynomial-time pseudodeterministic advice-taking oracle-algorithm A such that for infinitely many n, given a circuit $C\colon \{0,1\}^n \to \{0,1\}^{n+1}$ of size n^c, the algorithm A, with access to an NP oracle and n^ε bits of advice, outputs an $(n+1)$-bit string that is not in the range of C.*

1.3 Barriers for PSPACE-Relativizing Techniques

Next, we explain our negative results. We present EXPH-relativization barriers for "slightly improving" known results in *uniform derandomization*, *explicit construction*, and *circuit lower bounds*.

1.3.1 Uniform Derandomization

Standard "hardness vs. randomness" paradigm [59, 41] requires lower bounds against non-uniform circuits, such as $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{\varepsilon n}]$ for some constant $\varepsilon > 0$. In their seminal work, Impagliazzo and Wigderson [42] showed that hardness against uniform algorithms

also implies weak forms of derandomization: in particular, if $\mathsf{EXP} \neq \mathsf{BPP}$, then every algorithm in BPP can be derandomized in subexponential time infinitely often on average, i.e., $\mathsf{BPP} \subseteq \mathsf{i.o.heur\text{-}SUBEXP}$.[4]

We observe that their techniques are PSPACE-relativizing:

▶ **Proposition 7** (Uniform Derandomization in [42] is PSPACE-Relativizing). *Let* $\mathcal{O} \in \mathsf{PSPACE}$. *If* $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]} \neq \mathsf{BPP}^{\mathcal{O}}$, *then* $\mathsf{BPP}^{\mathcal{O}} \subseteq \mathsf{i.o.heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$.

More recently, Chen, Rothblum, and Tell [19] presented a uniform derandomization result on *almost all input lengths*: Given any language in PSPACE that is almost-everywhere hard against probabilistic algorithms, we can derandomize RP and BPP on average on almost every input length, where the derandomization of BPP requires some advice.[5] Their techniques are also PSPACE-relativizing:

▶ **Proposition 8** (Uniform Derandomization in [19] is PSPACE-Relativizing). *Let* $\mathcal{O} \in \mathsf{PSPACE}$ *and suppose that* $\mathsf{PSPACE}^{\mathcal{O}} \not\subseteq \mathsf{i.o.BPP}^{\mathcal{O}}$. *Then*

$$\mathsf{RP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]} \quad \textit{and} \quad \mathsf{BPP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]}/_{O(\log n)}.$$

The above results only obtain average-case derandomization instead of worst-case ones. Can we obtain worst-case derandomization based on uniform hardness assumptions, such as $\mathsf{EXP} \neq \mathsf{BPP}$? Consider the following "slight improvement of [42]":

▶ **Conjecture 9** (Worst-Case Uniform Derandomization). $\mathsf{EXP} \neq \mathsf{BPP} \implies \mathsf{BPP} \subseteq \mathsf{i.o.SUBEXP}$.

We present the following oracle suggesting that resolving Conjecture 9 would require new techniques.

▶ **Theorem 10.** *There is an oracle* $\mathcal{O} \in \mathsf{EXPH}$ *such that*

$$\mathsf{RP}^{\mathcal{O}} \not\subseteq \mathsf{i.o.DTIME}^{\mathcal{O}}[2^n] \quad \textit{and} \quad \mathsf{UP}^{\mathcal{O}} \not\subseteq \mathsf{BPTIME}^{\mathcal{O}}[2^n].$$

In this oracle world, we have $\mathsf{UP} \not\subseteq \mathsf{BPTIME}[2^n]$ (which is much stronger than $\mathsf{EXP} \neq \mathsf{BPP}$), while at the same time, worst-case derandomization of RP into deterministic fixed-exponential (2^n) time is impossible. This oracle is in EXPH, and therefore any PSPACE-relativizing proof of Conjecture 9 would also imply a breakthrough separation $\mathsf{PSPACE} \neq \mathsf{EXPH}$.

An open question left from [19] is whether the $O(\log n)$-bit advice can be removed in their derandomization of BPP. We present some evidence that this is difficult using current techniques:

▶ **Theorem 11.** *There is an oracle* $\mathcal{O}$ *such that*

$$\mathsf{BPP}^{\mathcal{O}} \not\subseteq \mathsf{heur\text{-}DTIME}^{\mathcal{O}}[2^n] \quad \textit{and} \quad \mathsf{UP}^{\mathcal{O}} \not\subseteq \mathsf{i.o.BPTIME}^{\mathcal{O}}[2^n].$$

Unfortunately, we do not know if the oracle $\mathcal{O}$ constructed in Theorem 11 is in EXPH. Nevertheless, under an assumption that is unlikely yet seems difficult to rule out, we show how to construct the oracle $\mathcal{O}$ in polynomial space. The assumption is that

$$\mathsf{SAT} \in \mathsf{DTIME}[n \cdot \mathrm{polylog}(n)] \cap \mathsf{NC}, \tag{*}$$

i.e., SAT admits both a near-linear-time sequential algorithm and a $\mathrm{polylog}(n)$-time parallel algorithm.

[4] Here, $\mathsf{heur\text{-}SUBEXP}$ is the class of problems solvable by a deterministic subexponential time heuristic over every samplable distribution; see Definition 38 for details.

[5] We only state the "low-end" version of [19] here; note that they also proved some "high-end" derandomization results.

▶ **Theorem 12.** *Suppose that* $\mathsf{SAT} \in \mathsf{DTIME}[n \cdot \mathrm{polylog}(n)] \cap \mathsf{NC}$. *Then there is an oracle* $\mathcal{O}$ *satisfying the conclusions of Theorem 11 that can be computed in polynomial space.*

It follows that if we use PSPACE-relativizing techniques to eliminate the $O(\log n)$-bit advice in the derandomization of [19], then we can refute (*), thus proving a non-trivial lower bound for SAT. Thus, we can still say that the oracle $\mathcal{O}$ in Theorem 11 presents some evidence that "current proof techniques" do not suffice to eliminate the $O(\log n)$-bit advice in Proposition 8.

1.3.2 Explicit Constructions

Next, we consider bounded relativization for explicit constructions. A property $Q \subseteq \{0,1\}^*$ is *dense* if for every input length $n \in \mathbb{N}$, we have that $|Q \cap \{0,1\}^n| \geq 2^n/\mathrm{poly}(n)$. Given any dense property Q decidable in polynomial time and an input length n, how hard is it to construct a length-n string that is in Q? This is a central open question in derandomization.

Recently, Oliveira and Santhanam [62] showed how to make progress if we allow the construction algorithm to be *pseudodeterministic*. Recall that an algorithm is *pseudodeterministic* if it outputs the same valid answer with high probability (despite being probabilistic). It was shown in [62] that for every dense property $Q \in \mathsf{P}$, there is a subexponential-time pseudodeterministic construction algorithm that succeeds infinitely often. We observe that this result is PSPACE-relativizing:

▶ **Proposition 13** ([62] is PSPACE-Relativizing; Informal)**.** *For every oracle* $\mathcal{O} \in \mathsf{PSPACE}$ *and every dense property* $Q \in \mathsf{P}^{\mathcal{O}}$*, there is a zero-error pseudodeterministic* $\mathcal{O}$*-oracle algorithm that on input* 1^n *outputs some element in* $Q \cap \{0,1\}^n$ *in subexponential time, for infinitely many* $n \in \mathbb{N}$.

Scott Aaronson's blog [2] contains a nice description of the result in [62], where he also mentioned that the result has "merely the following four caveats": (1) the algorithm runs in *subexponential* time instead of polynomial time; (2) the algorithm is not deterministic but *pseudodeterministic*; (3) instead of being almost-everywhere, the algorithm only succeeds on infinitely many input lengths; and (4) [62] only proved the *existence* of such an algorithm, but were unable to say what the algorithm is. In this paper, we show that caveats (2) and (3) cannot be improved by EXPH-relativizing techniques.

We start with (3), namely that EXPH-relativizing techniques cannot improve the pseudodeterministic construction algorithm in [62] to work on almost every input length.

▶ **Theorem 14.** *There is an oracle* $\mathcal{O} \in \mathsf{EXPH}$ *and a dense property* $Q \in \mathsf{P}^{\mathcal{O}}$ *such that every pseudodeterministic algorithm that runs in* $2^{o(n)}$ *time and attempts to find a string in* $Q \cap \{0,1\}^n$ *fails on infinitely many input lengths* n.

Goldwasser, Impagliazzo, Pitassi, and Santhanam [31] showed that the pseudodeterministic query complexity of a certain NP search problem is $\Omega(\sqrt{N})$, where N is the input length. We prove Theorem 14 by using the results in [31] *directly as a black box*. Each time we diagonalize against a certain pseudodeterministic machine by finding the lexicographically first counterexample (which is guaranteed to exist by [31]). This procedure, even when naïvely implemented, is computable in EXPH. We think this is the interesting aspect of Theorem 14, as it illustrates our main point that the ghost of (bounded) relativization has never gone away from complexity theory: for many natural complexity-theoretic statements that are non-relativizing, the straightforward counterexample oracle construction is already in EXPH.

Now we move to (2). We show that any deterministic construction algorithm for dense properties has to overcome an EXP-relativization barrier, even if the algorithm is allowed to only succeed on infinitely many input lengths:

▶ **Theorem 15.** *There is an oracle $\mathcal{O} \in \mathsf{EXP}$ and a dense property $Q \in \mathsf{P}^{\mathcal{O}}$ such that every deterministic algorithm that runs in $2^n/n^{\omega(1)}$ time fails to find a string in $Q \cap \{0,1\}^n$ on almost every input length n.*

In fact, the complexity of deterministic construction is known to be equivalent to the complexity of approximating Levin's Kt-complexity for a certain range of parameters [34]. Here, the Kt-complexity of a string x is defined to be the minimum of $|M| + \log t$ over all t-time Turing machines M of length $|M|$ such that M outputs x (see Definition 22 for a precise definition). Informally, MKtP is the problem of computing $\mathrm{Kt}(x)$ for a given string x. Although MKtP is EXP-complete under non-uniform polynomial-time reductions [5], it is a long-standing open question to show that $\mathsf{MKtP} \notin \mathsf{P}$. Using the connection between MKtP and deterministic construction [34], we show that Kt-complexity can be efficiently approximated under the oracle of Theorem 15.

▶ **Theorem 16** (Informal; see Theorem 55)**.** *There exists an oracle $\mathcal{O} \in \mathsf{EXP}$ under which $\mathrm{Kt}(x)$ can be approximated to within a factor of $(1+\varepsilon)$ for every constant $\varepsilon > 0$ in time $n^{O(\log n)}$ on input x of length n.*

Previously, Ren and Santhanam [63] constructed an oracle under which approximating $\mathrm{Kt}(x)$ to within a factor of $(2+\varepsilon)$ is in P. Theorem 16 improves this oracle construction, at the cost of increasing the running time from $n^{O(1)}$ to $n^{O(\log n)}$. We note that there are lower bounds for the randomized variant of MKtP [60, 36]. Their proof techniques are PSPACE-relativizing [60] and relativizing [36], respectively. Theorem 16 suggests that such proof techniques cannot be used to show a lower bound for approximating Kt-complexity without resolving $\mathsf{L} \neq \mathsf{P}$.

The property Q of Theorem 15 is a dense subset of $\left\{x : \mathrm{Kt}^{\mathcal{O}}(x) \geq n - c\log n\right\}$. To see why any $2^n/n^{\omega(1)}$-time algorithm $A^{\mathcal{O}}$ fails to find a string in $Q \cap \{0,1\}^n$, observe that the output of $A^{\mathcal{O}}$ on input 1^n has $\mathrm{Kt}^{\mathcal{O}}$-complexity at most $O(\log n) + \log(2^n/n^{\omega(1)}) \leq n - \omega(\log n)$, which is not in Q. To obtain Theorem 16, we regard Q as an errorless heuristic algorithm for $\mathrm{Kt}^{\mathcal{O}}$, and use the worst-case to average-case reduction for $\mathrm{Kt}^{\mathcal{O}}$ [33] to obtain a worst-case approximation algorithm for $\mathrm{Kt}^{\mathcal{O}}$-complexity.

1.3.3 Circuit Lower Bounds

Finally, we consider circuit lower bounds for Merlin–Arthur classes. Santhanam [65], improving [13], showed that for every integer k, there is a function computable in $\mathsf{MA}/_1$ (Merlin–Arthur protocols with one bit of advice) that does not have size-n^k circuits. We observe that this result is PSPACE-relativizing:

▶ **Proposition 17** ([65] is PSPACE-Relativizing)**.** *Let $\mathcal{O} \in \mathsf{PSPACE}$. Then for every constant $k \geq 1$, $\mathsf{MA}^{\mathcal{O}}/_1 \not\subseteq \mathsf{SIZE}^{\mathcal{O}}[n^k]$.*

By showing that (a variant of) the oracle constructed by [13] is in EXPH, we show that Santhanam's circuit lower bound cannot be improved to an almost-everywhere circuit lower bound by EXPH-relativizing proof techniques.

▶ **Theorem 18.** *There is an oracle $\mathcal{O} \in \mathsf{EXPH}$ such that $\text{pr-}\mathsf{MATIME}^{\mathcal{O}}[2^n] \subseteq \text{i.o.}\,\mathsf{SIZE}^{\mathcal{O}}[O(n)]$.*

In fact, it is unknown if the fixed polynomial-size lower bound for Σ_2^p due to Kannan [47] (i.e., $\Sigma_2^p \not\subseteq \mathsf{SIZE}[n^k]$ for every constant k) can be improved to an almost-everywhere circuit lower bound. We are not aware of any relativization barrier for improving Kannan's lower bound. To the best of our knowledge, Theorem 18 is the first evidence that "current proof techniques" cannot improve fixed polynomial-size circuit lower bounds for $\Sigma_2^p \supseteq$ pr-MA to almost everywhere. Note that the smallest complexity class known to be outside $\mathsf{i.o.SIZE}[n^k]$ for every k is $\Delta_3^p = \mathsf{P}^{\Sigma_2^p}$ [57].

▶ Remark 19 (The Infinite-Often Phenomenon). As noticed in [17], *almost-everywhere* separations in structural complexity theory are significantly harder to prove than infinitely-often separations. The PSPACE-relativization barrier provides an explanation of such difficulties – it is much easier to construct oracles (in EXPH) such that certain separation fails infinitely often. On the other hand, achieving almost-everywhere separations in oracle worlds might be much harder, and in some cases (such as in Theorem 18) the resulting oracle is not in EXPH anymore.

Besides circuit lower bounds for pr-MA, another notorious open problem mentioned in [17] is proving almost-everywhere NTIME hierarchies. Indeed, there is an oracle $\mathcal{O}$ relative to which $\mathsf{NTIME}[2^n] \subseteq \mathsf{i.o.RP}$ [12], and it is easy to see that this oracle can be implemented in EXPH.

1.4 Comparison with Algebrization

Our paper and the line of work on the algebrization barrier [23, 7, 3, 38, 8] share a common motivation, namely, to capture the limitations of "current techniques" after the interactive proof results such as $\mathsf{IP} = \mathsf{PSPACE}$ were proved via arithmetization. Therefore, we feel it necessary to compare the two barriers. However, before we make the comparison, let us briefly review the different variants of algebrization [23, 3, 38, 8].

1.4.1 Variants of Algebrization

Already in 1994, Fortnow [23] showed that every oracle $\mathcal{O}$ is many-one reducible to some oracle $\mathcal{A}$ such that $\mathsf{IP}^{\mathcal{A}} = \mathsf{PSPACE}^{\mathcal{A}}$; the oracle $\mathcal{A}$ is the "algebraic extension" of $\mathcal{O}$ as defined in [23, Section 5.3]. However, the definition of "algebraic extension" is rather involved and Fortnow did not show any unprovability results relative to "algebraic-extended" oracles.

Aaronson and Wigderson [3] defined an inclusion $\mathfrak{C} \subseteq \mathfrak{D}$ to *algebrize* if for every oracle $\mathcal{O}$ and its low-degree extension $\widetilde{\mathcal{O}}$, $\mathfrak{C}^{\mathcal{O}} \subseteq \mathfrak{D}^{\widetilde{\mathcal{O}}}$. This simplifies Fortnow's definition in the following sense. Roughly speaking, Fortnow's "algebraic extension" of $\mathcal{O}$ is another (Boolean) oracle that encodes all information of $\mathcal{O}$, $\widetilde{\mathcal{O}}$, $\widetilde{\widetilde{\mathcal{O}}}$, $\widetilde{\widetilde{\widetilde{\mathcal{O}}}}$, $\cdots$; the Aaronson–Wigderson definition is simpler in that it only considers $\mathcal{O}$ and $\widetilde{\mathcal{O}}$. However, this comes with a price of asymmetry that the RHS of an inclusion has access to $\widetilde{\mathcal{O}}$ but the LHS does not (e.g., $\mathsf{PSPACE}^{\mathcal{O}} \subseteq \mathsf{IP}^{\widetilde{\mathcal{O}}}$). Consequently, the Aaronson–Wigderson definition of algebrization is not closed under *modus ponens*: for example, although $\mathsf{NEXP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ does not algebrize [3, Theorem 5.6], it is possible that one could find an intermediate class $\mathfrak{C}$ such that both $\mathfrak{C} \subseteq \mathsf{NEXP}$ and $\mathfrak{C} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ algebrizes, thus using algebrizing techniques to separate NEXP from $\mathsf{P}/_{\mathrm{poly}}$.

Partially due to the above drawback, there are two subsequent works [38, 8] that aim at a more satisfactory definition of algebrization. Both definitions have the spirit that a statement is algebrizing if it holds relative to every oracle $\mathcal{O}$ *satisfying certain algebraic properties*. For example, a statement is IKK-algebrizing (IKK stands for Impagliazzo–Kabanets–Kolokolova) if it holds relative to every oracle $\mathcal{O}$ that satisfies the so-called *Arithmetic Checkability*

Theorem (ACT) [38]; a statement is *affine-relativizing* [8] if it holds relative to every affine extension – the result of a particular error correcting code applied to the characteristic string of a language. These variants of algebrization are indeed closed under *modus ponens*, and an algebrization barrier for a statement is simply an oracle that satisfies the algebraic property and under which the statement is false.

1.4.2 Comparison

Recall that a statement is $\mathfrak{C}$-relativizing if it holds relative to every oracle whose complexity is in $\mathfrak{C}$. In bounded relativization, we are interested in the complexity of the oracle instead of its algebraic properties. Therefore, the notions of (say) PSPACE-relativization and (say) IKK-algebrization appear incomparable: there are oracles outside PSPACE (in particular, undecidable ones) that satisfy the ACT; on the other hand, it is unknown if there are oracles in PSPACE that do not satisfy the ACT.

Moreover, as a framework for barrier results, we think that the main advantage of bounded relativization is its *simplicity*:

- To demonstrate a barrier result via bounded relativization, one only needs to construct a corresponding oracle *in the usual sense of relativization*, and show an upper bound on the complexity of the oracle (e.g., in EXPH). On the other hand, one needs to work against low-degree extensions of Boolean oracles or ensure the ACT in order to prove an algebrization barrier result.
- On the other hand, as demonstrated in Propositions 1 and 3, interactive proof characterizations of complexity classes are bounded-relativizing for trivial reasons. If $\mathsf{MIP}^* = \mathsf{RE}$ is true, then there is a one-liner proof of it being R-relativizing. In contrast, it appears that one needs to examine the 200-page proof of [44] thoroughly to tell whether it is algebrizing.

However, as we discussed before, the bounded relativization barrier has its own subtle disadvantages. An oracle construction in EXPH (such as Theorem 10) only tells us the following information: "if PSPACE-relativizing techniques can prove certain statements (such as Conjecture 9), then we can also separate EXPH from PSPACE". We interpret this as limitations of PSPACE-relativizing techniques, since we do not think such techniques are powerful enough to separate EXPH from PSPACE.[6] Hence, the bounded relativization barrier does not explain the difficulty of (e.g.) proving $\mathsf{PSPACE} \neq \mathsf{EXPH}$, since this difficulty itself is assumed to indicate barriers for proving other statements. We leave it as an interesting open problem to justify the statement that *although arithmetization-based techniques such as* $\mathsf{IP} = \mathsf{PSPACE}$ *are already available, fundamentally new techniques are needed to prove separation results such as* $\mathsf{PSPACE} \neq \mathsf{EXPH}$.[7]

[6] Although we should not be too pessimistic about the resolution of grand challenges in complexity theory such as PSPACE vs. EXPH, it seems extremely unlikely to us that they will be resolved by building an EXPH-computable oracle world inside which a certain statement is false, and proving the same statement using PSPACE-relativizing techniques. We suspect that techniques fundamentally "deeper" than those used to build oracles (e.g., diagonalization) are needed.

[7] Or, to refute this statement by proving $\mathsf{PSPACE} \neq \mathsf{EXPH}$!

2 Preliminaries

2.1 Definitions and Notations

For $n, i \in \mathbb{N}$ and $x \in \{0,1\}^n$, where $i \leq n$, we let $x_{[i]}$ denote the i-bit prefix of x. We use $\mathcal{U}_n$ to denote the uniform distribution over all length-n strings.

We assume the reader is familiar with basic complexity classes such as PSPACE, EXP, IP, BPP, MA; the definitions of these classes can be found in [30, 6]. One complexity class that occurs frequently in this paper and that might not be familiar to the broader audience, though, is EXPH, the exponential-time hierarchy.

▶ **Definition 20.** *A language L is in* EXPH *if there is a constant k, an exponential bound $e(n) = 2^{\mathrm{poly}(n)}$, and a $\mathrm{poly}(e(n))$-time algorithm M such that, for every input $x \in \{0,1\}^*$,*

$$x \in L \iff \exists y_1 \forall y_2 \exists y_3 \forall y_4 \ldots \exists y_k, M(x, y_1, \ldots, y_k) = 1,$$

where $y_i \in \{0,1\}^{e(n)}$ for each $i \in [k]$.

Equivalently, a language is in EXPH if it can be decided in exponential ($2^{\mathrm{poly}(n)}$) time with access to a PH oracle.

Pseudodeterministic Algorithms. A pseudodeterministic algorithm is a probabilistic algorithm that returns a *fixed* output with high probability on *any* given input. We will also consider pseudodeterministic *oracle* (or *advice-taking*) algorithms. Such an algorithm will maintain its pseudodeterministic behavior (i.e., outputting a fixed answer with high probability for any input) when a correct oracle (or advice) is given.

Time-Bounded Kolmogorov Complexity. We fix a time-optimal universal Turing machine U and a time-optimal randomized Turing machine V.

▶ **Definition 21.** *For a string $x \in \{0,1\}^*$, a time bound function $t\colon \mathbb{N} \to \mathbb{N}$, and an oracle $\mathcal{O}$, we define*

- ***(t-time-bounded Kolmogorov complexity)***

$$\mathrm{K}^{t,\mathcal{O}}(x) := \min_{d \in \{0,1\}^*} \Big\{ |d| : U^{\mathcal{O}}(d) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \Big\}.$$

- ***(Randomized t-time-bounded Kolmogorov complexity)***

$$\mathrm{rK}^{t,\mathcal{O}}(x) := \min_{d \in \{0,1\}^*} \Big\{ |d| : V^{\mathcal{O}}(d) \text{ runs in at most } t(|x|) \text{ steps and } \Pr[V^{\mathcal{O}}(d) = x] \geq 2/3 \Big\}.$$

We also consider Levin's notion of time-bounded Kolmogorov complexity [52].

▶ **Definition 22.** *For a string $x \in \{0,1\}^*$ and an oracle $\mathcal{O}$, we define*

$$\mathrm{Kt}^{\mathcal{O}}(x) := \min_{d \in \{0,1\}^*, t \in \mathbb{N}} \Big\{ |d| + \lceil \log t \rceil : U^{\mathcal{O}}(d) \text{ outputs } x \text{ in at most } t \text{ steps} \Big\}.$$

2.2 Technical Tools

▶ **Theorem 23** ([68])**.** *For every oracle $\mathcal{O}$, $\mathsf{BPP}^{\mathcal{O}} \subseteq (\Sigma_2^p)^{\mathcal{O}}$.*

We also need the following theorem for approximate counting that runs in linear time with a constant number of alternations. (In particular, Theorem 24 implies that approximate counting can be done in PH.)

▶ **Theorem 24** (Approximate Counting in Quasi-Linear-Time Hierarchy). *There is a* $\Sigma_5\mathsf{TIME}[n \cdot \mathrm{polylog}(n)]$ *machine* M *such that on input* (φ, K)*, accepts if* φ *has at least* K *satisfying assignments, and rejects if* φ *has at most* $K/2$ *satisfying assignments.*

Proof Sketch. Using universal hashing [69], we can show that there is a $\mathsf{BPTIME}[n \cdot \mathrm{polylog}(n)]^{\mathsf{SAT}}$ machine M' that approximates the number of satisfying assignments of φ'. Let $\mathcal{H}$ be a family of pairwise-independent hash functions whose output range is $\{0,1\}^{\lceil \log K \rceil}$, that is computable in near-linear time. (See, e.g., [72, Problem 3.3, (2)] for an example based on Toeplitz matrices.) The machine M' randomly samples $h \sim \mathcal{H}$ and $r \sim \{0,1\}^{\lceil \log K \rceil}$, and uses the SAT oracle to decide whether there is a satisfying assignment x of φ' such that $h(x) = r$.

Finally, the lemma follows from the fact that for every oracle $\mathcal{O}$, $\mathsf{BPTIME}^{\mathcal{O}}[t] \subseteq \Sigma_3\mathsf{TIME}[t \cdot \mathrm{polylog}(t)]^{\mathcal{O}}$ [74]. ◀

3 New Lower and Upper Bounds via Bounded Relativization

We start by showing a PSPACE-relativizing version of the pseudodeterministic PRG in [54].

3.1 A PSPACE-Relativizing, Pseudodeterministic, Efficient PRG

▶ **Theorem 25** (A PSPACE-Relativizing Efficient Pseudodeterministic PRG with One Bit of Advice). *For every* $\mathcal{O} \in \mathsf{PSPACE}$*,* $\varepsilon > 0$ *and* $c, d \geq 1$*, there exists a generator* $G = \{G_n\}_{n \in \mathbb{N}}$ *with* $G_n \colon \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ *for which the following holds:*

Efficiency: *There is a probabilistic polynomial-time algorithm* A *such that for every* $n \in \mathbb{N}$*, on input* 1^n *and* $z \in n^\varepsilon$*,* A*, with oracle access to* $\mathcal{O}$ *and one advice bit* $\alpha(n) \in \{0,1\}$ *that is independent of* z*, outputs* $G_n(z)$ *with probability* $\geq 2/3$*.*

Pseudorandomness: *For every language* $L \in \mathsf{DTIME}^{\mathcal{O}}[n^c]$*, there exist infinitely many input lengths* n *such that*

$$\left| \Pr_{x \sim \{0,1\}^n}[L(x) = 1] - \Pr_{z \sim \{0,1\}^{n^\varepsilon}}[L(G_n(z)) = 1] \right| \leq \frac{1}{n^d}.$$

To show Theorem 25, we consider two cases.

Case 1: $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$

▶ **Lemma 26.** *Let* $\mathcal{O} \in \mathsf{PSPACE}$*. If* $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$*, then there is a PRG with seed length* $O(\log n)$*, computable in pseudodeterministic polynomial time with oracle access to* $\mathcal{O}$*, that fools* $\mathsf{DTIME}^{\mathcal{O}}[n]$ *with error* $1/n$ *for all but finitely many* n*.*

We need the following "relativizing version" of a hardness-to-randomness construction.

▶ **Lemma 27** ([41, 50, 62])**.** *Let* $\mathcal{O}$ *be any language. If there is an* $\varepsilon > 0$ *and a Boolean function* $h \in \mathsf{BPE}^{\mathcal{O}}$ *that requires* $\mathcal{O}$*-oracle circuits of size* $2^{\varepsilon m}$ *on all but finitely many input length* m*, then there is a PRG with* $O(\log n)$ *seed length, computable in pseudodeterministic polynomial time with oracle access to* $\mathcal{O}$*, that fools* $\mathsf{DTIME}^{\mathcal{O}}[n]$ *with error* $1/n$ *for all but finitely many* n*.*

Proof of Lemma 26. Since $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$, by a simple padding argument, we have that $\mathsf{DSPACE}\left[2^{O(n)}\right] \subseteq \mathsf{BPE}^{\mathcal{O}}$.

By direct diagonalization, there is a language $L \in \mathsf{DSPACE}\left[2^{O(n)}\right]$ such that, for all but finitely many input lengths n, L does not have $\mathcal{O}$-oracle circuits of size $2^{0.9n}$. Indeed, using $2^{O(n)}$ space, we can find the lexicographically first truth table of length 2^n that cannot be

computed by $\mathcal{O}$-oracle circuits of size $2^{0.9n}$. This can be done because $\mathcal{O} \in \mathsf{PSPACE}$ and we can simulate any such circuit in $2^{O(n)}$ space. By the simulation in the previous paragraph, we have that $L \in \mathsf{BPE}^{\mathcal{O}}$. Now the desired conclusion follows from Lemma 27. ◀

Case 2: $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$

▶ **Lemma 28.** *Let $\mathcal{O}$ be any language and suppose $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$. Then for every $\varepsilon > 0$ and $c, d \geq 1$, there exists a* PRG *with seed length n^{ε}, computable in pseudodeterministic polynomial time with oracle access to $\mathcal{O}$ and one bit of advice, that fools $\mathsf{DTIME}^{\mathcal{O}}[n^c]$ with error $1/n^d$ for infinitely many n.*

Proof Sketch. The proof is essentially the same as that of [54]. The only difference is that instead of assuming $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}$ in [54], here we assume $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$. We provide a high-level description of the proof here. For details, we refer the reader to [54, Sections 1.2 and 3.1].

As in [54], Lemma 28 can be shown by combining hierarchy theorems for probabilistic classes, which can be viewed as hardness results against (uniform) probabilistic algorithms, with the hardness-to-randomness framework under uniform hardness assumptions [42, 70]. More specifically, for any oracle $\mathcal{O}$ and $k \geq 1$, we can construct a language $L_k \in \mathsf{BPP}^{\mathcal{O}}/_1$ that is hard against $\mathsf{BPTIME}^{\mathcal{O}}[n^k]/_1$. (See [54, Lemma 19].)

As in [26], assuming $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$, the language L_k is obtained by constructing a padded version of a certain PSPACE-completed language L_{hard} that has useful structural properties, such as *downward self-reducibility*, *random self-reducibility*, and *instance checkability* [70, 26], using the idea of an "optimal $\mathcal{O}$-*oracle* algorithm" for L_{hard}. (See [54, Lemma 17].)

The language L_k constructed in this way will have certain forms of downward self-reducibility and random self-reducibility. These properties enable us to plug L_k into the hardness-to-randomness construction of [70] (with additional "relativization properties" observed by [50]), and get a PRG that is infinitely-often secure against $\mathcal{O}$-*oracle* adversaries. Also, the fact that $L_k \in \mathsf{BPP}^{\mathcal{O}}/_1$ allows us to compute the PRG efficiently with oracle access to $\mathcal{O}$ and with one bit of advice, in a pseudodeterministic manner. ◀

Proof of Theorem 25. The theorem follows directly from Lemma 26 and Lemma 28, since in both the cases of $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$ and of $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$, we get a PRG that satisfies the conditions stated in the theorem. ◀

3.2 A Nearly Maximum Circuit Lower Bound for $\mathsf{BPE}^{\mathrm{MCSP}}/_{2^{\varepsilon n}}$

▶ **Theorem 29.** *For every $\varepsilon > 0$,*

$$\mathsf{BPE}^{\mathrm{MCSP}}/_{2^{\varepsilon n}} \not\subseteq \mathsf{SIZE}[2^n/n].$$

We first show the following two lemmas.

▶ **Lemma 30.** *For every $\varepsilon > 0$, there is a probabilistic polynomial-time algorithm A such that the following holds.*

- *For every $N \in \mathbb{N}$, on input 1^N, the algorithm A, with oracle access to* MCSP *and with an advice $\alpha(N) \in \{0,1\}^{N^{\varepsilon}}$, outputs with high probability a fixed truth table T_N of length $2^{n:=\lfloor \log N \rfloor}$ (which corresponds to some function $f\colon \{0,1\}^{n=\lfloor \log N \rfloor} \to \{0,1\}$), and*
- *for infinitely many $N \in \mathbb{N}$, T_N has circuit complexity greater than $2^n/n$.*

Proof. Let D be the following algorithm. On input $x \in \{0,1\}^N$, D lets $n := \lfloor \log N \rfloor$ and checks if $\mathrm{MCSP}\big(x_{[2^n]}, 2^n/n\big)$ is false. It is clear D can be determined in time $\mathsf{DTIME}^{\mathrm{MCSP}}[O(N)]$ and that D accepts x only if $x_{[2^n]}$ is a truth table with circuit complexity greater than $2^n/n$. Also, since most truth tables of length 2^n have circuit complexity greater than $2^n/n$ [67, 56], the acceptance probability of D is at least $1/2$.

Consider the generator $\{G_N\}_N$ from Theorem 25 that fools $\mathsf{DTIME}^{\mathrm{MCSP}}[O(N)]$ for infinitely many N. By the security of $\{G_N\}_N$, there exists some seed $z \in \{0,1\}^{N^\varepsilon}$ such that $D(G_N(z)) = 1$, and with oracle access to MCSP and with some bit $b \in \{0,1\}$, we can compute $G_N(z)$ pseudodeterministically in time $\mathrm{poly}(N)$. Therefore, given such z and b as advice (which is of $N^\varepsilon + 1$ bits but we can always scale the parameter ε), we can obtain with high probability some fixed truth table of length 2^n with circuit complexity greater than $2^n/n$. ◀

▶ **Lemma 31.** *For every $\varepsilon > 0$, there is a probabilistic algorithm B such that the following holds.*

- *For every $n \in \mathbb{N}$, on input 1^n, the algorithm B, with oracle access to* MCSP *and with an advice $\alpha(n) \in \{0,1\}^{2^{\varepsilon n}}$, runs in time $2^{O(n)}$ and outputs with high probability a fixed truth table T_n of length 2^n, and*
- *for infinitely many $n \in \mathbb{N}$, T_n has circuit complexity greater than $2^n/n$.*

Proof. Let $\varepsilon_0 := \varepsilon/2$. We first describe the advice used by the algorithm B. For input 1^n, the first part of the advice is a number N (if exists) such that

- $n = \lfloor \log N \rfloor$, and that
- the algorithm A from Lemma 30 (instantiated with parameter ε_0) on input 1^N, with oracle access to MCSP and with N^{ε_0} bits of advice, outputs with high probability a fixed truth table of length 2^n with circuit complexity greater than $2^n/n$.

Note that such a number N can be specified using $n+1$ bits. Also, we say that n is *good* if such an N exists. By Lemma 30, there are infinitely many good n.

The second part of the advice is the N^{ε_0} bits that are needed to compute $A(1^N)$. Note that the total number of bits for the advice is at most

$$n + 1 + N^{\varepsilon_0} < 2^{\varepsilon n}.$$

We let the output of $B(1^n)$ be the output of $A(1^N)$. Then whenever n is good, the canonical output of $A(1^N)$ (hence $B(1^n)$) will be a truth table of length 2^n with circuit complexity greater than $2^n/n$. ◀

Theorem 29 now follows easily from Lemma 31.

Proof of Theorem 29. Consider the language $L \in \mathsf{BPE}^{\mathrm{MCSP}}/_{2^{\varepsilon n}}$ that can be computed as follows. On input $x \in \{0,1\}^n$, we run $B(1^n)$, where B is the algorithm from Lemma 31, using MCSP as an oracle and using $2^{\varepsilon n}$ bits of advice. With high probability, we obtain the truth table of some fixed function $f_n \colon \{0,1\}^n \to \{0,1\}$. We then let $L(x) := f_n(x)$. By Lemma 31, for infinitely many n, f_n has circuit complexity greater than $2^n/n$, which implies that $L \notin \mathsf{SIZE}[2^n/n]$. ◀

3.3 Circuit Lower Bounds for Meta-Complexity Problems

In this subsection, we show that the problem of estimating $\mathrm{rK}^{\mathrm{poly},\mathrm{MCSP}}$ (the MCSP-oracle version of polynomial-time randomized Kolmogorov complexity) does not have fixed-polynomial-size circuits.

▶ **Theorem 32** (An Unconditional Circuit Lower Bound for Estimating $\mathrm{rK}^{\mathsf{poly},\mathrm{MCSP}}$). *For every $\varepsilon > 0$ and $c \geq 1$ there exists a constant $k \geq 1$ such that the following holds. Consider the following promise problem $\Pi^\varepsilon = (\mathcal{YES}_n, \mathcal{NO}_n)_{n\in\mathbb{N}}$, where*

$$\begin{aligned} \mathcal{YES}_n &:= \left\{ x \in \{0,1\}^n : \mathrm{rK}^{t,\mathrm{MCSP}}(x) \leq n^\varepsilon \right\}, \\ \mathcal{NO}_n &:= \left\{ x \in \{0,1\}^n : \mathrm{rK}^{t,\mathrm{MCSP}}(x) \geq n - 10 \right\}, \end{aligned}$$

and $t(n) = n^k$. Then $\Pi^\varepsilon \notin \mathsf{SIZE}[n^c]$.

To show Theorem 32, we first construct an efficient pseudodeterministic PRG, with oracle access to MCSP and an advice string of length $O(\log n)$, that can fool *circuits* (instead of uniform algorithms in [54]).

3.3.1 A Pseudodeterministic PRG Fooling Circuits

▶ **Theorem 33.** *For every $\varepsilon > 0$ and every $c, d \geq 1$, there is a generator $G = \{G_n\}_{n\in\mathbb{N}}$ with $G_n : \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ such that the following holds.*

Efficiency: *There is a probabilistic polynomial-time algorithm A such that for every $n \in \mathbb{N}$, on input 1^n and $z \in \{0,1\}^{n^\varepsilon}$, A, with oracle access to* MCSP *and an advice $\alpha(n) \in \{0,1\}^{O(\log n)}$ that is independent of z, outputs $G_n(z)$ with probability $\geq 2/3$.*

Pseudorandomness: *For infinitely many $n \in \mathbb{N}$, for every circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size at most n^c,*

$$\left| \Pr_{x\sim\{0,1\}^n}[C(x) = 1] - \Pr_{z\sim\{0,1\}^{n^\varepsilon}}[C(G_n(z)) = 1] \right| \leq \frac{1}{n^d}.$$

We need the following hardness-to-randomness construction.

▶ **Lemma 34** (Umans [71]). *There is a universal constant g and a function $G_{\mathrm{Umans}} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for all s and Y where the circuit complexity of Y (viewed as a truth table) is at least s^g, and for all circuits C of size s,*

$$\left| \Pr_{x\sim\{0,1\}^{g\log|Y|}}[C(G_{\mathrm{Umans}}(Y,x)) = 1] - \Pr_{x\sim\{0,1\}^s}[C(x) = 1] \right| \leq \frac{1}{s}.$$

Moreover, G_{Umans} is computable in time $\mathrm{poly}(|Y|)$.

We also need the following variant of Lemma 30, whose proof is essentially the same as that of Lemma 30. We omit the details here.

▶ **Lemma 35.** *For every $\varepsilon > 0$ and $d \geq 1$, there is a probabilistic polynomial-time algorithm A such that the following holds.*

- *For every $N \in \mathbb{N}$, on input 1^N and $z \in \{0,1\}^{N^\varepsilon}$, the algorithm A, with oracle access to* MCSP *and with an advice $\alpha_A(N) \in \{0,1\}$, outputs with high probability a fixed truth table $T_{N,z}$ of length $2^{m:=\lfloor \log N \rfloor}$ (which corresponds to some function $f\colon \{0,1\}^m \to \{0,1\}$), and*
- *for infinitely many $N \in \mathbb{N}$, with probability at least $1 - 1/N^d$ over z, $T_{N,z}$ has circuit complexity at least $2^{m/2}$.*

We now present the PRG in Theorem 33.

Proof of Theorem 33. The idea is to combine Lemma 35 and Lemma 34.

Let $c' := \max\{c, 2d\}$ and $\varepsilon_0 := \varepsilon/(7c'g)$, where g is the constant from Lemma 34.

For $n \in \mathbb{N}$, we first specify the advice for computing the generator G_n. The first part of the advice is a number N (if exists) such that

- $n^{3c'g} \leq N < (n+1)^{3c'g}$, and that

- the algorithm A from Lemma 35 (instantiated with parameters ε_0 and $d := 1$), on input 1^N and $z \in \{0,1\}^{N^{\varepsilon_0}}$, with oracle access to MCSP and with one advice bit $\alpha_A(N)$, outputs (with high probability) a fixed truth table of length $2^{\lfloor \log N \rfloor}$ that has circuit complexity greater than $2^{\lfloor \log N \rfloor/2}$, for at least $1 - 1/N$ fraction of the z's.

Note that N can be encoded using $O(\log n)$ bits. We say that n is *good* if such an N exists. By Lemma 35, there are infinitely many good n. The second part of the advice is then the bit $\alpha_A(N)$ needed to run the algorithm A.

We compute G_n as follow. Given $z := (z^{\mathsf{first}}, z^{\mathsf{second}})$, where $z^{\mathsf{first}} \in \{0,1\}^{n^{\varepsilon/2}}$ and $z^{\mathsf{second}} \in \{0,1\}^{O(\log n)}$, we first invoke the algorithm A from Lemma 35 on 1^N (where N is the first part of the advice as described in the previous paragraph) and z^{first} to obtain a truth table $T_{n,z^{\mathsf{first}}}$. We can compute A using MCSP as oracle and the bit $\alpha_A(N)$, which is the second part of the advice. We will then use the hardness-to-randomness construction in Lemma 34 on $T_{n,z^{\mathsf{first}}}$. Note that since $|T_{n,z^{\mathsf{first}}}| \leq n^{O(1)}$, the input length of $G_{\mathrm{Umans}}(T_{n,z^{\mathsf{first}}}, -)$ can be $O(\log n)$. Finally, we output

$$G_{\mathrm{Umans}}\left(T_{n,z^{\mathsf{first}}}, z^{\mathsf{second}}\right)_{[n]}.$$

It is easy to verify that G_n satisfies the efficiency condition stated in Theorem 33, since A is a pseudodeterministic polynomial-time algorithm and $G_{\mathrm{Umans}}(T_{n,z^{\mathsf{first}}}, -)$ runs in time $\mathrm{poly}(n)$ deterministically.

Next, we show the pseudorandomness condition of G_n. Note that for any good n, with probability at least $1 - 1/n^{c'}$ over z^{first}, the algorithm A will output (with high probability) some fixed truth table $T_{n,z^{\mathsf{first}}}$ with circuit complexity at least $n^{c'g}$. Whenever $T_{n,z^{\mathsf{first}}}$ has such circuit complexity, $G_{\mathrm{Umans}}\left(T_{n,z^{\mathsf{first}}}, -\right)$, by Lemma 34, will be a PRG that $(1/n^{c'})$-fools size-$(n^{c'})$ circuits. In order words, with probability at least $1 - 1/n^{c'}$ over z^{first}, for every circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size at most $n^{c'}$,

$$\left| \Pr_{x \sim \{0,1\}^n}[C(x) = 1] - \Pr_{z^{\mathsf{second}}}[C(G_{\mathrm{Umans}}(T_{n,z^{\mathsf{first}}}, z^{\mathsf{second}})) = 1] \right| \leq \frac{1}{n^{c'}}.$$

Then by a union bound, we get

$$\left| \Pr_{x \sim \{0,1\}^n}[C(x) = 1] - \Pr_{z}[C(G_s(z)) = 1] \right| \leq \frac{1}{n^{c'}} + \frac{1}{n^{c'}} \leq \frac{1}{n^d},$$

as desired. ◀

3.3.2 Circuit Lower Bound for $\mathrm{rK}^{\mathrm{poly},\mathrm{MCSP}}$

We complete the proof of Theorem 32 using our PRG in Theorem 33.

Proof of Theorem 32. Let t be a polynomial specified later. For the sake of contradiction, suppose there exists a circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size at most n^c such that for all $x \in \{0,1\}^n$,

- if $\mathrm{rK}^{t,\mathrm{MCSP}}(x) \leq n^{\varepsilon}$ then $C(x) = 1$, and
- if $\mathrm{rK}^{t,\mathrm{MCSP}}(x) \geq n - 10$ then $C(x) = 0$.

Note that by a simple counting argument, for most $x \in \{0,1\}^n$, we have $\mathrm{rK}^{t,\mathrm{MCSP}}(x) \geq n - 10$. This gives

$$\Pr_{x \sim \{0,1\}^n}[C(x) = 0] \geq \frac{1}{2}.$$

Consider the PRG in Theorem 32 instantiated with parameters $\varepsilon/2$, c and $d := 1$. We have that for infinitely many n, G_n $(1/n)$-fools circuits of size at most n^c. On the one hand, this implies

$$\Pr_{z\sim\{0,1\}^{n^{\varepsilon/2}}}[C(G_n(z)) = 0] \geq \Pr_{x\sim\{0,1\}^n}[C(x) = 0] - \frac{1}{n} \geq \frac{1}{3}. \tag{1}$$

On the other hand, since G_n can be computed, with oracle access to MCSP and with $O(\log n)$ bits of advice, pseudodeterministically in time $\mathrm{poly}(n)$, we have that there exists some polynomial t with $t(n) = \mathrm{poly}(n)$ such that for every $z \in \{0,1\}^{n^{\varepsilon/2}}$

$$\mathrm{rK}^{t,\mathrm{MCSP}}(G_s(z)) \leq n^{\varepsilon/2} + O(\log n) < n^{\varepsilon},$$

which by our assumption implies that

$$\Pr_{z\sim\{0,1\}^{\varepsilon/2}}[C(G_n(z)) = 1] = 1.$$

This contradicts Equation (1). ◀

3.4 Pseudodeterministic Construction for Range Avoidance

In this subsection, we show a pseudodeterministic algorithm, using an NP oracle and an advice string of sub-polynomial length, that solves the range avoidance problem infinitely often.

▶ **Theorem 36.** *For every $\varepsilon > 0$ and every $d \geq 1$, there is a probabilistic polynomial-time algorithm A such that the following holds.*

- *For every $m \in \mathbb{N}$, on any input circuit $C\colon \{0,1\}^m \to \{0,1\}^{m+1}$ of size m^d, the algorithm A, with access to an NP oracle and with an advice $\alpha(m) \in \{0,1\}^{m^{\varepsilon}}$,[8] outputs with high probability a string $x_C \in \{0,1\}^{m+1}$, and*
- *for infinitely many $m \in \mathbb{N}$, $x_C \notin \mathsf{Range}(C)$ for each input C.*

The following lemma is implicit in [51], attributed to [43].

▶ **Lemma 37** (Adaption of [51])**.** *There exists an algorithm R that runs in deterministic polynomial time with an NP oracle such that the following holds. For every $m \in \mathbb{N}$, given any circuit $C\colon \{0,1\}^m \to \{0,1\}^{m+1}$ and a truth table of length 2^n with circuit complexity at least $2^{n/2}$, where $n := 4\lceil \log(m \cdot |C|)\rceil + \lfloor \log m \rfloor$, the algorithm R outputs a string $x \in \{0,1\}^{m+1}$ such that $x \notin \mathsf{Range}(C)$.*

Proof Sketch. The proof is adapted from those of [51, Lemma 3 and Theorem 7].

First of all, it can be shown that for every $m \in \mathbb{N}$ and for every circuit $C\colon \{0,1\}^m \to \{0,1\}^{m+1}$, there exists a circuit $D\colon \{0,1\}^m \to \{0,1\}^{2m}$, which can be constructed efficiently from C, of size $m \cdot |C|$, such that given a string $y \in \{0,1\}^{2m}$ that is not in the range D and given access to an NP oracle, one can compute efficiently a string $x \in \{0,1\}^{m+1}$ that is not in the range of C. (See the proof of [51, Lemma 3].) Therefore, to show the lemma, it suffices to show how to find, for a given circuit D of size $m \cdot |C|$ mapping m bits to $2m$ bits, a string that is not in the range of D.

Given a circuit $D\colon \{0,1\}^m \to \{0,1\}^{2m}$, let $k := 4\lceil \log(|D|)\rceil$. It can be shown that one can efficiently construct a circuit $E\colon \{0,1\}^m \to \{0,1\}^{2^k m}$ with the following properties. Given a $(2^k m)$-bit string that is not in the range of E and given access to an NP oracle,

[8] We stress that the advice $\alpha(m)$ does *not* depend on the input circuit C.

we can compute in time $\mathrm{poly}(|D|)$ a string $x \in \{0,1\}^{2m}$ that is not in the range of D. Moreover, every string $y \in \{0,1\}^{2^k m}$ in the range of E has small circuit complexity, in the sense that there is a circuit $F_y \colon \{0,1\}^{k+\lceil \log m \rceil} \to \{0,1\}$ of size at most $O(|D| \log |D|)$ that computes a truth table (of length $2^{k+\lceil \log m \rceil}$) whose prefix is y. (See [51, Figure 1 and Proof of Theorem 7].)

Now if we have a truth table $T \in \{0,1\}^{2^n}$ with circuit complexity at least $2^{n/2}$, where $n = k + \lfloor \log m \rfloor$, we can construct a $(2^k m)$-bit string

$$y_T := T \circ 0^{2^k m - 2^n}.$$

For the sake of contradiction, suppose y_T is in the range of E. Then by the discussion in the previous paragraph, it is easy to construct a circuit from F_{y_T} of size at most $O(|D| \log |D|)$ that computes T, which has circuit complexity at least $2^{n/2} = 2^{(4\lceil \log(|D|) \rceil + \lfloor \log m \rfloor)/2} \geq |D|^2$. This gives a contradiction. ◀

We are now ready to show Theorem 36.

Proof of Theorem 36. The idea is to use Lemma 31 to obtain a hard truth table, and plug it in Lemma 37 to solve the range avoidance problem.

Let $\varepsilon_0 := \varepsilon/(10d)$, and let B be the pseudodeterministic algorithm from Lemma 31 instantiated with parameter ε_0.

We say that $m \in \mathbb{N}$ is *good* if for $n := 4\lceil (d+1) \log m \rceil + \lfloor \log m \rfloor$, the algorithm B on input 1^n, with oracle access to MCSP and with $2^{\varepsilon_0 n}$ bits of advice, outputs with high probability some fixed truth table of circuit complexity greater than $2^n/n$. Note that there are infinitely many good m, since there are infinitely many n such that $B(1^n)$ will successfully output a fixed hard truth table with high probability.

Consider the following algorithm A. Given a circuit $C \colon \{0,1\}^m \to \{0,1\}^{m+1}$ of size m^d, A first runs $B(1^n)$, where $n := 4\lceil (d+1) \log m \rceil + \lfloor \log m \rfloor$, using an NP-complete oracle (that can simulate MCSP) and using an advice string of length $2^{\varepsilon_0 n} \leq m^{\varepsilon}$, to obtain (with high probability) a fixed truth table of length 2^n. We then run the algorithm R from Lemma 37 using this truth table and using the NP-complete oracle again to get a string $x \in \{0,1\}^{m+1}$. Note that whenever m is good (and sufficiently large), the truth table output by $B(1^n)$ will have circuit complexity greater than $2^n/n \geq 2^{n/2}$, so using such a hard truth table the algorithm R will successfully output a string that is not in the range of C. ◀

4 Barriers for Derandomization under Uniform Assumptions

4.1 PSPACE-Relativizing Derandomization under Uniform Assumptions

We show that previous results on derandomization under uniform assumptions are PSPACE-relativizing. This includes the subexponential-time derandomization of Impagliazzo and Wigderson [42] (see also [70]) and the recent "unstructured hardness to average-case randomness" [19]. These results only achieve average-case derandomization, therefore it is helpful to introduce the following definition.[9]

[9] In the literature, it is more common to define heur-P as a family of *distributional* problems, which are pairs $(L, \mathcal{D})$ where L is a language and $\mathcal{D}$ is a distribution. In this paper, we only care about whether our derandomization succeeds on all polynomial-time samplable distributions. Thus, for simplicity, we define heur-P as simply a class of languages that admits polynomial-time heuristics over all polynomial-time samplable distributions.

▶ **Definition 38** (Average-Case Complexity Classes). *We say a language L is in* heur-P *if for every polynomial-time samplable distribution $\mathcal{D}$ and every polynomial p, there is a deterministic polynomial-time algorithm $\mathcal{A}$ such that*

$$\Pr_{x\sim\mathcal{D}}[\mathcal{A}(x) = L(x)] \geq 1 - 1/p(n).$$

Similarly, if the algorithm $\mathcal{A}$ runs in quasi-polynomial time, then we say $L \in$ heur-QuasiP; if for every $\varepsilon > 0$ there is such an algorithm $\mathcal{A}$ running in 2^{n^ε} time, then we say $L \in$ heur-SUBEXP.

We start by demonstrating that the techniques in [42] are PSPACE-relativizing:

▶ **Proposition 7** (Uniform Derandomization in [42] is PSPACE-Relativizing). *Let $\mathcal{O} \in$ PSPACE. If $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]} \neq \mathsf{BPP}^{\mathcal{O}}$, then $\mathsf{BPP}^{\mathcal{O}} \subseteq$ i. o. heur-$\mathsf{SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$.*

Proof Sketch. We first prove that if $\mathsf{PSPACE}^{\mathcal{O}} \neq \mathsf{BPP}^{\mathcal{O}}$, then $\mathsf{BPP}^{\mathcal{O}} \subseteq$ i. o. heur-$\mathsf{SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$. Observe that since $\mathcal{O} \in \mathsf{PSPACE}$, we have $\mathsf{PSPACE}^{\mathcal{O}} = \mathsf{PSPACE}$. Let L be a $\mathsf{PSPACE}^{\mathcal{O}}$-complete problem that is both downward self-reducible and random self-reducible (as constructed in [70]). One could define a family of pseudorandom generators $G_n : \{0,1\}^{n^{o(1)}} \to \{0,1\}^n$ based on the hardness of L such that the following holds: Let D be any oracle such that for every $n \in \mathbb{N}$,

$$|\Pr[D(\mathcal{U}_n) = 1] - \Pr[D(G_n(\mathcal{U}_{n^{o(1)}})) = 1]| > 1/n,$$

then L can be decided in randomized polynomial time with a D oracle. Moreover, G_n is computable in $2^{n^{o(1)}}$ time. If $\mathsf{BPP}^{\mathcal{O}} \not\subseteq$ i. o. heur-$\mathsf{SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$, then in particular G_n does not fool certain BPP algorithm (on average, on every input length). It follows from the reconstruction properties of $\{G_n\}$ that $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$.

We consider two cases. Suppose that $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]} \not\subseteq \mathsf{P}^{\mathcal{O}}/_{\mathrm{poly}}$, then it follows from standard relativizing hardness-randomness trade-offs [59, 41] that $\mathsf{BPP}^{\mathcal{O}} \subseteq$ i. o. $\mathsf{SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$. On the other hand, if $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]} \subseteq \mathsf{P}^{\mathcal{O}}/_{\mathrm{poly}}$, then by the relativizing Karp-Lipton theorem [48], $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]} \subseteq (\Sigma_2^p)^{\mathcal{O}} \subseteq \mathsf{PSPACE}^{\mathcal{O}}$. Since $\mathsf{PSPACE}^{\mathcal{O}} \neq \mathsf{BPP}^{\mathcal{O}}$, it follows (again) that $\mathsf{BPP}^{\mathcal{O}} \subseteq$ i. o. heur-$\mathsf{SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$. ◀

Before we consider [19], it is helpful to define the notion of *logspace-uniform* low-depth circuits:

▶ **Definition 39** (Logspace-Uniform Circuits, see [19, Definition 3.5]). *We say that a circuit family $\{C_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ of size $T(n)$ is* logspace-uniform *if there exists a machine M that on input 1^n runs in space $O(\log T(n))$ and prints C_n. For two functions $T(n)$ and $d(n)$, we denote the class of languages computable by logspace-uniform circuits of size $T(n)$ and depth $d(n)$ by* lu-$\mathsf{CKT}[T, d]$.

We show the following PSPACE-relativizing version of [19, Theorem 5.2]. Note that we can only achieve quasi-polynomial time derandomization if we want a PSPACE-relativizing result; this is because an oracle in PSPACE might require $2^{\mathrm{poly}(n)}$ time (instead of $2^{O(n)}$ time) to compute, and consequently, the HSG in [19] has seed length polylog(n) (instead of $O(\log n)$). It might be possible to obtain a $\mathsf{SPACE}[n]$-relativizing result that achieves polynomial-time derandomization, but we do not pursue this direction here.

▶ **Theorem 40** (High-End Uniform Derandomization in [19] is PSPACE-Relativizing)**.** *Let $\mathcal{O} \in \mathsf{PSPACE}$ and $\varepsilon > 0$ be a constant. Assume there is a function L computable by logspace-uniform circuits of size $2^{\mathrm{poly}(n)}$ and depth $2^{n^{o(1)}}$, making $\mathcal{O}$-oracle queries of $\mathrm{poly}(n)$ length, such that $L \notin \mathsf{i.o.BPTIME}\left[2^{n^{\varepsilon}}\right]^{\mathcal{O}}$. (That is, $\mathsf{lu\text{-}CKT}^{\mathcal{O}[\mathrm{poly}]}\left[2^{\mathrm{poly}(n)}, 2^{n^{o(1)}}\right] \not\subseteq \mathsf{i.o.BPTIME}^{\mathcal{O}}\left[2^{n^{\varepsilon}}\right]$.) Then*

$$\mathsf{RP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}QuasiP}^{\mathcal{O}} \quad \textit{and} \quad \mathsf{BPP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}QuasiP}^{\mathcal{O}}/_{\log^{o(1)} n}.$$

Proof Sketch. The key observation is that since $\mathcal{O} \in \mathsf{PSPACE}$, $\mathcal{O}$ itself admits a logspace-uniform circuit of size $2^{\mathrm{poly}(n)}$ and depth $\mathrm{poly}(n)$. It follows that the hard language L in our assumption can actually be computed by a logspace-uniform circuit of size $2^{\mathrm{poly}(n)}$ and depth $2^{n^{o(1)}}$ *without any oracles*. That is:

$$L \in \mathsf{lu\text{-}CKT}^{\mathcal{O}[\mathrm{poly}]}\left[2^{\mathrm{poly}(n)}, 2^{n^{o(1)}}\right] \subseteq \mathsf{lu\text{-}CKT}\left[2^{\mathrm{poly}(n)}, 2^{n^{o(1)}}\right].$$

The rest of the proof follows [19, Theorem 5.2] closely. In what follows, the input length of the hard problems with low-depth circuits will be denoted as ℓ instead of n. Let $L^{(1)} \in \mathsf{lu\text{-}CKT}\left[2^{O(\ell)}, 2^{\ell^{o(1)}}\right]$ be an instance-checkable problem from [19, Proposition 4.4] such that L polynomial-time reduces to $L^{(1)}$.[10] The instance checker for $L^{(1)}$ runs in $2^{\ell^{o(1)}}$ time. By [19, Lemma A.1], there exists a constant $\varepsilon' > 0$ and a language $L' \in \mathsf{lu\text{-}CKT}\left[2^{O(\ell)}, 2^{\ell^{o(1)}}\right]$ that is not in $\mathsf{i.o.BPTIME}^{\mathcal{O}}\left[2^{\ell^{\varepsilon'}}\right]$ and has an instance checker running in $2^{\ell^{o(1)}}$ time.

The next step is to create an HSG from the hard function L'. Let $\mathcal{A}$ be a probabilistic algorithm which we want to derandomize, n be its input length, and $M := n^k$ be the amount of randomness it uses. Let $\ell := \lceil \log^{2/\varepsilon'} n \rceil$, $f : \{0,1\}^n \to \{0,1\}^{2^\ell}$ be the multi-output function that discards the input and outputs the truth table of $L'_\ell := L' \cap \{0,1\}^\ell$. We use [19, Theorem 4.5] with parameter $\delta := \ell^{\varepsilon'/4-1}$ to obtain a generator G_f and a reconstruction algorithm R.

- The generator G_f runs in time $T^{O(1/\delta)} \leq 2^{\ell^2} \leq 2^{\mathrm{polylog}(n)}$ and outputs a list of M-bit strings.
- The reconstruction algorithm R gets oracle access to a function $D : \{0,1\}^M \to \{0,1\}$, and runs in time $2^{\ell^{o(1)}} \cdot T^\delta \leq 2^{\ell^{\varepsilon'/3}}$. Assume that D $(1/M)$-avoids the generator G_f, then w.p. $\geq 1 - \frac{O(\log^2 T)}{T}$, R^D prints an oracle circuit C_f such that the truth table of C_f^D is exactly L'_ℓ.

Given an $\mathsf{RP}^{\mathcal{O}}$ algorithm $\mathcal{A}$, we use the HSG to fool $\mathcal{A}$, making it run in deterministic $T^{O(1/\delta)} \leq 2^{\mathrm{polylog}(n)}$ time. Now suppose there is a distribution $\mathcal{D}$ samplable in polynomial time with $\mathcal{O}$ oracles, as well as an infinite set of input lengths $\mathcal{I} \subseteq \mathbb{N}$ such that our derandomization fails on $\mathcal{I}$ w.p. $1/\mathrm{poly}(n)$. This means that w.p. at least $1/\mathrm{poly}(n)$ over $x \sim \mathcal{D}$, we have $\Pr_r[A(x,r) = 1] \geq 1/2$ but $A(x,w) = 0$ for every w in the HSG.

We will use a reconstruction argument to compute L'_ℓ in probabilistic $2^{\ell^{\varepsilon'}}$ time with an $\mathcal{O}$ oracle, reaching a contradiction. We say an input length ℓ is nice if there is some $n \in \mathcal{I}$ such that $\ell = \lceil \log^{2/\varepsilon'} n \rceil$; there are infinitely many nice input lengths. On input $x \in \{0,1\}^\ell$,

[10] Proposition 4.4 of [19] only claimed to hold for $L \in \mathsf{lu\text{-}CKT}\left[2^{O(\ell)}, 2^{\ell^{o(1)}}\right]$ instead of $\mathsf{lu\text{-}CKT}\left[2^{\mathrm{poly}(\ell)}, 2^{\ell^{o(1)}}\right]$. We can use a padding argument to reduce L to some problem in $\mathsf{lu\text{-}CKT}\left[2^{O(\ell)}, 2^{\ell^{o(1)}}\right]$ and then invoke this proposition; the only difference is that the reduction runs in polynomial time instead of linear time.

where ℓ is a nice input length, we sample uniformly at random an input length n such that $\ell = \lceil \log^{2/\varepsilon'} n \rceil$, and sample $x \sim \mathcal{D}_n$. With probability $1/\mathrm{poly}(n)$, the following good event (which we denote $\mathcal{G}$) happens: n is good and $A(x,-)$ is a distinguisher for our hitting set. Then we invoke the instance checker for L'_ℓ on input x. Whenever the instance checker asks a query q, we use $R^{A(x,-)}$ to find the answer of $L'(q)$. If $\mathcal{G}$ happens then w.h.p. the instance checker outputs the correct answer $L'(x)$; even if $\mathcal{G}$ does not happen, w.h.p. the instance checker outputs either $L'(x)$ or $\perp$. By sampling n and x for $\mathrm{poly}(\ell)$ times, w.h.p. $\mathcal{G}$ will happen at least once, and we compute $L'_\ell(x)$ successfully. Since there are $\mathrm{poly}(n)$ samples, the instance checker runs in $2^{\ell^{o(1)}}$ time, the oracle $A(x,-)$ runs in $\mathrm{poly}(n)$ time, and the reconstruction algorithm runs in $2^{\ell^{\varepsilon'/3}}$ time, the total running time is

$$\mathrm{poly}(n) \cdot 2^{\ell^{o(1)}} \cdot \mathrm{poly}(n) \cdot 2^{\ell^{\varepsilon'/3}} \ll 2^{\ell^{\varepsilon'}}.$$

The proof for $\mathsf{BPP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}QuasiP}^{\mathcal{O}}/_{\log^{o(1)} n}$ is similar. The only difference is to notice that the targeted HSG constructed in [19, Theorem 4.5] is also a *targeted somewhere-PRG*, consisting of $2^{\ell^{o(1)}}$ sub-PRGs one of which is guaranteed to be secure. The advice for the derandomized algorithm is the index of the sub-PRG that is secure, thus has length $\log(2^{\ell^{o(1)}}) \le \log^{o(1)} n$. ◀

▶ **Proposition 8** (Uniform Derandomization in [19] is PSPACE-Relativizing)**.** *Let $\mathcal{O} \in \mathsf{PSPACE}$ and suppose that $\mathsf{PSPACE}^{\mathcal{O}} \not\subseteq \mathsf{i.o.}\,\mathsf{BPP}^{\mathcal{O}}$. Then*

$$\mathsf{RP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]} \quad \textit{and} \quad \mathsf{BPP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]}/_{O(\log n)}.$$

Proof Sketch. The proof outline is the same as Theorem 40, but the parameters are a bit different. In particular, the hard language L' is in $\mathsf{lu\text{-}CKT}\big[2^{O(\ell)}, \mathrm{poly}(\ell)\big] \setminus \mathsf{BPP}^{\mathcal{O}}$. Fix any constant $\varepsilon > 0$, the HSG has the following parameters:

$$\ell := n^{\varepsilon/3}, \quad M := n^k, \quad \text{and } \delta := (3/\varepsilon)\log \ell/\ell.$$

The generator runs in $T^{O(1/\delta)} \le 2^{O(\ell^2)} \le 2^{n^\varepsilon}$ time and outputs a list of M-bit strings. The reconstruction algorithm runs in $O(nT^\delta) \le \mathrm{poly}(\ell)$ time. If the generator fails to fool a certain $\mathsf{RP}^{\mathcal{O}}$ algorithm (on average and infinitely often), then we can use the reconstruction algorithm to compute L' in $\mathsf{BPP}^{\mathcal{O}}$, contradicting our hypothesis. Therefore the generator successfully fools every $\mathsf{RP}^{\mathcal{O}}$ algorithm (on average and infinitely often), and thus $\mathsf{RP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]}$.

Moreover, the generator is a somewhere PRG: It can be partitioned into $d' := \mathrm{poly}(\ell)$ "sub-PRGs" such that for every $\mathsf{BPP}^{\mathcal{O}}$ algorithm, one of the "sub-PRG" fools this algorithm successfully (on average and infinitely often). We embed the index of the correct "sub-PRG" as advice, taking $\log d' = O(\log n)$ bits. Therefore $\mathsf{BPP}^{\mathcal{O}} \subseteq \mathsf{heur\text{-}SUBEXP}^{\mathcal{O}[\mathrm{poly}]}/_{O(\log n)}$. ◀

4.2 Bounded-Relativization Barriers for Uniform Derandomization

4.2.1 Barrier for Worst-Case Derandomization under Uniform Assumptions

Recall that [42] showed that if $\mathsf{EXP} \neq \mathsf{BPP}$ then $\mathsf{BPP} \subseteq \mathsf{i.o.}\,\mathsf{heur\text{-}SUBEXP}$. We present some evidence that current techniques cannot improve this result to worst-case derandomization:

▶ **Theorem 10.** *There is an oracle $\mathcal{O} \in \mathsf{EXPH}$ such that*

$$\mathsf{RP}^{\mathcal{O}} \not\subseteq \mathsf{i.o.}\,\mathsf{DTIME}^{\mathcal{O}}[2^n] \quad \textit{and} \quad \mathsf{UP}^{\mathcal{O}} \not\subseteq \mathsf{BPTIME}^{\mathcal{O}}[2^n].$$

Note that $\mathsf{UP} \not\subseteq \mathsf{BPTIME}[2^n]$ is much stronger than $\mathsf{EXP} \neq \mathsf{BPP}$, yet it still does not imply fixed exponential time worst-case derandomization of RP by EXPH-relativizing techniques.

▶ **Corollary 41.** *If we can use* PSPACE*-relativizing techniques to show*

$$\mathsf{EXP} \neq \mathsf{BPP} \implies \mathsf{BPP} \subseteq \mathsf{i.o.SUBEXP},$$

then $\mathsf{L} \neq \mathsf{NP}$ *follows.*

Proof of Theorem 10. Our oracle world will consist of two oracles $\mathcal{P}$ and $\mathcal{Q}$ (it is easy to combine them into a single oracle). The oracle $\mathcal{P}$ creates a hard problem in RP against deterministic algorithms, and the oracle $\mathcal{Q}$ creates a hard problem in UP against probabilistic algorithms. In particular, every input to the oracle $\mathcal{P}$ will be of the form (x, r) where $|r| = 100|x|$, and let

$$L_{\mathcal{P}} := \left\{ x : \exists r \in \{0,1\}^{100|x|}, \mathcal{P}(x, r) = 1 \right\}.$$

We will guarantee that for every x, $\Pr_r[\mathcal{P}(x, r) = 1]$ is either 0 or greater than $1/2$, thus $L_{\mathcal{P}} \in \mathsf{RP}$. We also define

$$L_{\mathcal{Q}} := \left\{ 1^n : \exists r \in \{0,1\}^{100n}, \mathcal{Q}(r) = 1 \right\}.$$

We will guarantee that for every input length $100n$, $|\mathcal{Q} \cap \{0,1\}^{100n}| \leq 1$, thus $L_{\mathcal{Q}} \in \mathsf{UP}$.

Let $M_1, M_2, \ldots$ be an enumeration of Turing machines running in deterministic 2^n time, and $N_1, N_2, \ldots$ be an enumeration of probabilistic Turing machines running in 2^n time (that does not necessarily satisfy the BPP promise).[11] We want every M_i to fail to solve $L_{\mathcal{P}}$ on all but finitely many input lengths, and every N_i to fail to solve $L_{\mathcal{Q}}$ on infinitely many input lengths. In particular, let n_1 be a large enough constant, and $n_i = 2^{500 n_{i-1}}$ for each $i \geq 2$. Assuming that every machine appears in the sequence $\{N_i\}$ infinitely often, we want that for each $i \in \mathbb{N}$, N_i fails to solve $L_{\mathcal{Q}}$ on input 1^{n_i}.

We also need the following terminologies. The *n-th slice* of $\mathcal{P}$ is the set of inputs of the form (x, r) where $|x| = n$ and $|r| = 100n$. The *ℓ-th slice* of $\mathcal{Q}$ is simply the set of inputs with length ℓ. During our construction, there will be some entries $\mathcal{P}(x, r)$ and $\mathcal{Q}(r)$ that are *fixed*, meaning that their values will never change in the subsequent construction. There will also be some inputs x that are *frozen*, meaning that for every $r \in \{0,1\}^{100|x|}$, the values of $\mathcal{P}(x, r)$ are fixed and will never change in the subsequent construction. If an entry $\mathcal{P}(x, r)$ or $\mathcal{Q}(r)$ is not fixed and, in the case of $\mathcal{P}(x, r)$, x is not frozen, then we say this entry is *free*.

Our construction proceeds in stages. During stage n, we will fix the entire n-th slices of $\mathcal{P}$ and $\mathcal{Q}$; we will possibly also fix or freeze some other entries. It will be guaranteed that before the n-th stage:

1. the number of frozen strings beyond the n-th slice (including the n-th slice) is at most $2^{n/2}$;
2. the number of fixed strings (except the frozen ones) beyond the n-th slice (including the n-th slice) is at most 2^{5n}; and
3. each fixed or frozen entry beyond the n-th slice (including the n-th slice) is set to be 0.

[11] In this paper, whenever we enumerate a list of Turing machines, we assume every machine occurs in the list *infinitely many times*.

Diagonalization against M_i. Let $x_1, x_2, \ldots, x_n$ be n inputs of length n that are not frozen. Since there are at most $2^{n/2}$ frozen inputs of length n, it is always possible to select n unfrozen inputs. For each $i \in [n]$, we simulate the machine M_i on input x_i. Whenever M_i asks a query $\mathcal{P}(x, r)$ or $\mathcal{Q}(r)$, if this query is not free, then we return the corresponding value fixed before; if it is free, then we return 0 as the answer and also *fix* this query to be 0. Then M_i will output an answer $ans_i \in \{0, 1\}$. Note that there are at most $2^{5n} + n \cdot 2^n \ll 2^{100n}/3$ entries in $\mathcal{P}$ that are fixed, thus the vast majority of the entries $\mathcal{P}(x_i, r)$ are free. We fix all these entries $\mathcal{P}(x_i, r)$ to be $(1 - ans_i)$. It is easy to see that M_i fails to compute $L_{\mathcal{P}}$ on input x_i.

Diagonalization against N_i. If $n = 10n_i$ for some $i \in \mathbb{N}$, then we also need to fix the $100n_i$-th slice of $\mathcal{Q}$ so that N_i fails to compute $L_{\mathcal{Q}}$ on input 1^{n_i}. (In stage n, this happens after the above diagonalization against M_i.) We simulate N_i on input 1^{n_i}. Note that N_i is a probabilistic machine running in time 2^{n_i}, thus it has $B := 2^{2^{n_i}}$ computational branches, where each branch has $1/B$ probability mass. On each branch, whenever N_i asks a query $\mathcal{P}(x, r)$ or $\mathcal{Q}(r)$, if this query is not free, then we return the corresponding value fixed before; if it is free, then we return 0 as the answer to the query. Note that we cannot fix the query as there are potentially as many as B queries to fix. Instead, after simulating all the computational branches, we pick out the "heavy" queries that occur in a lot of branches and fix them:

- A string x is *heavy* if at least $\frac{1}{10^4|x|^3}$ fraction of branches queried some free entry of the form $\mathcal{P}(x, r)$.
- An entry $\mathcal{Q}(r)$ is *heavy* if at least $\frac{1}{10^4}$ fraction of branches queried $\mathcal{Q}(r)$.

We *freeze* every heavy x by setting every unfixed entry $\mathcal{P}(x, r)$ to be 0. We also *fix* every heavy $\mathcal{Q}(r)$ to be 0. Note that each computational branch of N_i only makes at most 2^{n_i} queries. Therefore, for each input length m where $n \leq m \leq 2^{n_i}$, the number of heavy strings $x \in \{0, 1\}^m$ is at most $10^4 2^{n_i} m^3$. The number of entries $\mathcal{Q}(r)$ fixed before this stage is at most 2^{5n}, and the number of heavy entries $\mathcal{Q}(r)$ is at most $10^4 2^{n_i}$, therefore the total number of fixed entries in $\mathcal{Q}$ after this stage is at most $2^{6n} \ll 2^{100n_i}$.

After simulating every computational branch of N_i, let p be the probability that N_i outputs 1 on input 1^n. If $p < 0.5$, then we pick a free $r \in \{0, 1\}^{100n_i}$ and set $\mathcal{Q}(r) = 1$; since the number of fixed entries $\mathcal{Q}(r)$ is $\ll 2^{100n_i}$, such r always exists. If $p \geq 0.5$, then we do nothing.

Now we have that N_i solves $L_{\mathcal{Q}}$ on input 1^{n_i} with probability less than 2/3 (actually, at most 1/2). Is this always the case in the future? Consider how the future changes to the oracles might affect the behavior of N_i. Let m be an input length where $n < m \leq 2^n$. During stage m (which has not happened yet), we will pick m inputs $x_1, x_2, \ldots, x_m \in \{0, 1\}^m$, then some entries of the form $\mathcal{P}(x_i, r)$ will be set to be 1. Also note that the probability mass of computational branches that each x_i affects is at most $\frac{1}{10^4 m^3}$. Therefore, the total probability mass of computational branches affected is at most

$$\sum_{m \in \mathbb{N}} \frac{m}{10^4 m^3} < \frac{1}{10^3}.$$

Note that we also set at most one entry $\mathcal{Q}(r)$ to be 1, where $|r| = 100n_i$. (The next time we diagonalize against N_{i+1} happens in stage $n_{i+1} \gg 2^n$.) Since this entry $\mathcal{Q}(r)$ is not heavy, the probability mass of computational branches that it affects is at most $\frac{1}{10^4}$.

It follows that only an $\frac{11}{10^4}$ fraction of computational branches of N_i will be affected in the future. That is, let p' be the probability that N_i outputs 1 on input 1^{n_i} at the end of our construction (i.e., after stage $\gg 2^n$), then $|p' - p| < \frac{11}{10^4}$. Therefore, it cannot be the case that

$$\Pr[N_i(1^{n_i}) = L_{\mathcal{Q}}(1^{n_i})] \geq 2/3.$$

Clean-up. At the end of stage n, we fix every unfixed input $\mathcal{P}(x, r)$ in the n-th slice to be 0. If $n = 10n_i$ then we also fix every unfixed input $\mathcal{Q}(r)$ in the $10n$-th slice to be 0. The number of frozen strings is at most $10^4 2^{n/10} n^3 \leq 2^{n/2}$ (note that at most one N_i contributes to the frozen strings since the input lengths $\{n_i\}$ are very far apart). The number of fixed entries above the $(n+1)$-st slice is at most

$$2^{5n} + n \cdot 2^n + \sum_{m=n}^{2^{n/10}} 10^4 2^{n/10} m^3 < 2^{5(n+1)}.$$

The complexity of $\mathcal{O}$. We show that the above oracles $\mathcal{P}$ and $\mathcal{Q}$ are computable in EXPH. It suffices to present a deterministic algorithm with access to a Σ_3^p oracle that on input 1^n, outputs the truth tables of the n-th slices of $\mathcal{P}$ and $\mathcal{Q}$ in $2^{O(n)}$ time.

We simulate each stage $n' \leq n$. Before stage n', we maintain the set of fixed entries $\mathcal{P}(x, r)$, $\mathcal{Q}(r)$, as well as the set of frozen strings x. There are at most $2^{O(n')}$ many such strings. In what follows, we use $\langle \mathcal{P}, \mathcal{Q} \rangle$ to denote (the length-$2^{O(n')}$ encoding of) the list of fixed entries and frozen strings.

It is easy to diagonalize against each M_i: we simply choose the inputs $x_1, x_2, \ldots, x_{n'}$, simulate each M_i on input x_i, and fix the oracles accordingly. This takes $2^{O(n')}$ time.

To simulate N_i, we need to define the following language capturing the weight of queries. Consider simulating the probabilistic machine N_i on the current oracle $\langle \mathcal{P}, \mathcal{Q} \rangle$. Whenever N_i makes a query, if this query is not free, then we return the answer of the query; otherwise we return 0. The following two claims are immediate corollaries of Theorem 23:

▷ Claim 42. There is a language Heavy-String$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, x)$ computable in Σ_2^p such that:

- if at least $\frac{1.1}{10^4 |x|^3}$ fraction of computational branches of N_i queried some free entry of the form $\mathcal{P}(x, -)$, then Heavy-String$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, x) = 1$;
- if at most $\frac{0.9}{10^4 |x|^3}$ fraction of computational branches of N_i queried some free entry of the form $\mathcal{P}(x, -)$, then Heavy-String$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, x) = 0$.

▷ Claim 43. There is a language Heavy-Entry$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, r)$ computable in Σ_2^p such that:

- if at least $\frac{1.1}{10^4}$ fraction of branches of N_i queried $\mathcal{Q}(r)$, then Heavy-Entry$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, r) = 1$;
- if at most $\frac{0.9}{10^4}$ fraction of branches of N_i queried $\mathcal{Q}(r)$, then Heavy-Entry$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, r) = 0$.

(Note that both Heavy-String and Heavy-Entry accepts inputs of length $2^{O(n')}$.)

With the aid of these two languages, it is possible to simulate stage n' now. We say a string x is *heavy* if Heavy-String$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, x) = 1$, and an entry $\mathcal{Q}(r)$ is *heavy* if Heavy-Entry$(\langle \mathcal{P}, \mathcal{Q} \rangle, N_i, r) = 1$. Since there are at most $2^{O(n')}$ heavy strings, we can find all of them in $2^{O(n')}$ time with access to an $\mathsf{NP}^{\text{Heavy-String}}$ oracle. Similarly, since there are $2^{O(n')}$ heavy entries, we can find all of them in $2^{O(n')}$ time with access to an $\mathsf{NP}^{\text{Heavy-Entry}}$ oracle. We freeze these heavy strings and fix these heavy entries accordingly.

Finally, we estimate the probability that N_i outputs 1 (given current versions of $\mathcal{P}$ and $\mathcal{Q}$) in Σ_2^p, within additive error 0.1. If this estimation is less than 1/2, then we pick a free $r \in \{0,1\}^{10n'}$ and set $\mathcal{Q}(r) = 1$; otherwise we do nothing.

At the end of stage n', we fix every unfixed entry in the n'-th slice of $\mathcal{P}(x,r)$ to be 0. If $n = 10n_i$ then we also fix every unfixed input $\mathcal{Q}(r)$ in the $10n$-th slice to be 0. This takes $2^{O(n')}$ time.

It is easy to see that simulating the n stages takes $2^{O(n)}$ time with access to a Σ_3^p oracle. ◀

▶ Remark 44. It is instructive to see why this construction only rules out worst-case derandomization instead of average-case derandomization. (If we could rule out average-case derandomization, then PSPACE $\neq$ EXPH follows from Proposition 7!)

The reason is that we do not want to affect the acceptance probability of $N_i(1^{n_i})$ by too much. Let $m \in [10n_i + 1, 2^{n_i}]$ be a "future" input length, we say $x \in \{0,1\}^m$ is "used for diagonalization" if there is some $\mathcal{P}(x,r)$ that is fixed to 1. Each string x used for diagonalization influences the accept probability of N_i by a small amount ($\frac{1}{10^4 m^3}$ in our proof), therefore we cannot afford to have too many strings used for diagonalization.

If we only want each M_i to fail in the worst case, we only need to use one input $x_i \in \{0,1\}^m$ to diagonalize against each M_i, hence we only need to use $\omega(1)$ strings x for diagonalization. On the other hand, if we want each M_i to fail on average (say, under the uniform distribution over $\{0,1\}^m$), then there has to be a lot of strings (e.g., $2^m/\mathrm{poly}(m)$) used for diagonalization; our previous guarantees on the acceptance probability of N_i will be completely destroyed.

4.2.2 Barrier for Almost-Everywhere Derandomization without Advice

It is natural to ask whether the results of [42] can be improved to *almost-everywhere* derandomization. Recently, substantial progress was made by Chen, Rothblum, and Tell [19], who showed average-case derandomization on *almost every input length* based on uniform assumptions. In particular, assuming $\mathsf{lu\text{-}CKT}[2^{O(n)}, 2^{o(n)}] \not\subseteq \mathsf{i.o.BPTIME}[2^{\Omega(n)}]$, they showed that $\mathsf{RP} \subseteq \mathsf{heur\text{-}P}$ and $\mathsf{BPP} \subseteq \mathsf{heur\text{-}P}/_{O(\log n)}$.

One open question is whether the $O(\log n)$-bit advice in the derandomization of BPP can be eliminated. In Theorem 11, we present an oracle world where this improvement is not possible. Unfortunately, we do not know how to implement this oracle in EXPH; nevertheless, we still show in Theorem 12 that eliminating the $O(\log n)$-bit advice by PSPACE-relativizing techniques would require proving new lower bounds for SAT.

▶ **Theorem 11.** *There is an oracle $\mathcal{O}$ such that*

$$\mathsf{BPP}^{\mathcal{O}} \not\subseteq \mathsf{heur\text{-}DTIME}^{\mathcal{O}}[2^n] \quad \textit{and} \quad \mathsf{UP}^{\mathcal{O}} \not\subseteq \mathsf{i.o.BPTIME}^{\mathcal{O}}[2^n].$$

Proof. We use a similar construction as in Theorem 10. The difference is that now we want a problem in BPP that is infinitely-often hard, and a problem in UP that is almost-everywhere hard, therefore we need to define the oracles differently. In particular, the oracle $\mathcal{P}$ only receives one input r, and the oracle $\mathcal{Q}$ receives two inputs (x,r), where $|r| = 10|x|$. The hard language in BPP is:

$$L_{\mathcal{P}} = \left\{ 1^n : \Pr_{r \sim \{0,1\}^{n^4}}[\mathcal{P}(r) = 1] \geq 1/2 \right\}.$$

It is guaranteed that for every n, the fraction of length-n^4 strings that are in $\mathcal{P}$ is either at most $1/3$ or at least $2/3$, therefore $L_{\mathcal{P}} \in \mathsf{BPP}$. On the other hand, the hard language in UP is:

$$L_{\mathcal{Q}} = \left\{ x : \exists r \in \{0,1\}^{10|x|}, \mathcal{Q}(x,r) = 1 \right\}.$$

It is guaranteed that for every input x, there is at most one string $r \in \{0,1\}^{10|x|}$ such that $\mathcal{Q}(x,r) = 1$, therefore $L_{\mathcal{Q}} \in \mathsf{UP}$. We will construct the oracles $\mathcal{P}$ and $\mathcal{Q}$ such that $L_{\mathcal{P}} \notin \mathsf{DTIME}[2^n]$ and $L_{\mathcal{Q}} \notin \mathsf{i.o.BPTIME}[2^n]$. Note that $L_{\mathcal{P}}$ is a unary language, therefore we also have $\mathsf{BPP} \not\subseteq \mathsf{heur\text{-}DTIME}[2^n]$.

We use the same terminologies as before: $M_1, M_2, \ldots$ is an enumeration of deterministic Turing machines running in 2^n time, and $N_1, N_2, \ldots$ is an enumeration of probabilistic Turing machines running in 2^n time. The ℓ-th *slice* of $\mathcal{P}$ is the set of inputs with length ℓ, and the n-th *slice* of $\mathcal{Q}$ is the set of inputs of the form (x,r) where $|x| = n$ and $|r| = 10n$. An entry is *fixed* if its value will never change in the subsequent construction. We choose n_1 to be a large enough constant and define a sequence $\{n_i\}$ where $n_i = 2^{10n_{i-1}}$ for each $i \geq 2$. We want every N_i to fail to solve $L_{\mathcal{Q}}$ on all but finitely many input lengths, and every M_i to fail to solve $L_{\mathcal{P}}$ on input 1^{n_i}.

Our construction proceeds in stages. We guarantee that:

- For every input length n that is not in the sequence $\{n_i\}$, the n-th slice of $\mathcal{P}$ is identically 0.
- For every input length n, there are at most $2n$ entries in the entire n-th slice of $\mathcal{Q}$ that returns 1.
- Consider only the oracle $\mathcal{Q}$. Before the n-th stage, the number of fixed entries beyond the n-th slice of $\mathcal{Q}$ (including the n-th one) is at most 2^{6n}, and all of them are fixed to be 0.

We start from the $n^\star$-th stage for some large enough constant $n^\star$. For each $n \geq n^\star$, in the n-th stage, we diagonalize against the machines N_i and M_i, and fix the n-th slice of $\mathcal{Q}$ as well as the n^4-th slice of $\mathcal{P}$ (when n equals some n_i). Details follow.

Diagonalization against N_i. Let n' be the smallest integer in the sequence $\{n_i\}$ such that $n' \geq n$, and i' be the index such that $n' = n_{i'}$. The $(n')^4$-th slice of $\mathcal{P}$ may contain either few or most strings, and we do not know which is the case yet. Therefore, we will diagonalize each N_i twice, first assuming the n'-th slice of $\mathcal{P}$ is nearly empty and then assuming the n'-th slice of $\mathcal{P}$ is nearly full. For this reason, we will need $2n$ distinct strings $x_{i,b} \in \{0,1\}^n$, one for each pair of $i \in [n]$ and $b \in \{0,1\}$. Note that at the beginning of the n-th stage, there are at most $2^{6n} \ll 2^{10n}$ strings fixed in the n-th slice of $\mathcal{Q}$, therefore we can always choose $2n$ strings $x_{i,b}$ where no entry of the form $\mathcal{Q}(x_{i,b}, r)$ is fixed.

For each $i \in [n]$ and $b \in \{0,1\}$, we simulate $N_i(x_{i,b})$, assuming that most entries in the n'-th slice of $\mathcal{P}$ returns b. Note that N_i is a probabilistic machine running in time 2^n, thus it has $B := 2^{2^n}$ computational branches, where each branch has $1/B$ probability mass. On each branch, whenever N_i asks a query $\mathcal{P}(r)$ or $\mathcal{Q}(x,r)$, if this query is already fixed, then we return the corresponding value fixed before; otherwise:

- Suppose the query is $\mathcal{P}(r)$. If $|r|$ is in the sequence $\{n_i^4\}$, then since $\mathcal{P}(r)$ is not fixed and $|r| \leq 2^n$, we know that $|r| = (n')^4$ and we return b as the answer. Otherwise ($|r| \neq (n')^4$) we return 0.
- Suppose the query is $\mathcal{Q}(x,r)$, then we return 0 as the answer.

Again, we can only afford to fix the *heavy* entries, defined as follows. Let K be the number of machine-input pairs that we still need to simulate in the future before fixing the $(n')^4$-th slice of $\mathcal{P}$; note that this includes both the current $N_i(x_{i,b})$ and the final $M_{i'}(1^{n'})$. That is:

$$K = 2(n-i) + (2-b) + \sum_{j=n+1}^{n'} (2j) + 1.$$

- An entry $\mathcal{P}(r)$ is *heavy* if $|r| = (n')^4$ and at least $2^{-(n'+4)K}$ fraction of branches queried $\mathcal{P}(r)$.
- An entry $\mathcal{Q}(x,r)$ is *heavy* if at least $\frac{1}{10^4|x|^3}$ fraction of branches queried $\mathcal{Q}(x,r)$.

We fix every heavy entry according to the way we answer this entry before. That is, every heavy entry $\mathcal{P}(r)$ is fixed to be b and every heavy entry $\mathcal{Q}(x,r)$ is fixed to be 0. The number of heavy entries in $\mathcal{P}$ is at most $2^{(n'+4)K} \cdot 2^n$; for each input length $m \geq n$, the number of heavy entries in the m-th slice of $\mathcal{Q}$ is at most $2^n \cdot 10^4 \cdot m^3$.

Let p be the probability (over the B computational branches) that $N_i(x_{i,b})$ outputs 1. If $p < 1/2$ then we find a string r such that $\mathcal{Q}(x_{i,b}, r)$ is unfixed yet and set $\mathcal{Q}(x_{i,b}, r) = 1$, making $x_{i,b} \in L_{\mathcal{Q}}$. Such a string r exists since we have fixed strictly less than 2^{10n} strings in the n-th slice of $\mathcal{Q}$. If $p \geq 1/2$ then we do nothing.

Assuming that most entries in the $(n')^4$-th slice of $\mathcal{P}$ indeed returns b, at this point the probability that N_i outputs the correct answer on input $x_{i,b}$ is at most $1/2$. However, the oracles $\mathcal{P}$ and $\mathcal{Q}$ might change in the future, thus the behavior of N_i might also change. What fraction of computational branches might be affected in the future?

- Consider the $K-1$ machine-input pairs that we will simulate in the future before the $(n')^4$-th slice of $\mathcal{P}$ is completely fixed. The number of heavy entries in the $(n')^4$-th slice of $\mathcal{P}$ that these machines fix is at most

  $$\sum_{k=1}^{K-1} 2^{(n'+4)k} \cdot 2^{n'} \leq 2 \cdot 2^{(n'+4)(K-1)} \cdot 2^{n'}.$$

  Therefore, the probability mass of computational branches of $N_i(x_{i,b})$ influenced by the future changes to the $(n')^4$-th slice of $\mathcal{P}$ is at most

  $$2 \cdot 2^{(n'+4)(K-1)} \cdot 2^{n'} \cdot 2^{-(n'+4)K} \leq 1/8.$$

- For each $m \leq 2^n$, there are at most $2m$ entries in the m-th slice of $\mathcal{Q}$ that we will fix to be 1. The probability mass of computational branches of $N_i(x_{i,b})$ influenced by these entries is at most

  $$\sum_{m=n}^{2^n} \frac{2m}{10^4 m^3} < \frac{1}{10^3}.$$

It follows that the future modifications to the oracles $\mathcal{P}$ and $\mathcal{Q}$ will only affect the accept probability of $N_i(x_{i,b})$ by $\frac{1}{8} + \frac{1}{10^3} < \frac{1}{6}$. Therefore, it cannot be the case that

$$\Pr[N_i(x_{i,b}) = L_{\mathcal{Q}}(x_{i,b})] \geq 2/3.$$

Diagonalization against M_i. If $n = n_i$ for some $i \in \mathbb{N}$, then we want that M_i does not compute $L_{\mathcal{P}}$ on input 1^n correctly. We simulate M_i on the input 1^n. When M_i asks a query $\mathcal{P}(r)$ or $\mathcal{Q}(x,r)$, if this query is already fixed, then we return the corresponding value fixed before; otherwise we return 0 and fix this queried entry.

After this simulation, we argue that only a small fraction of entries in the n^4-th slice of $\mathcal{P}$ are fixed. First, the simulation of $M_i(1^n)$ fixes at most 2^n entries. Second, there are at most $K = \sum_{j=1}^{n}(2j) = O(n^2)$ machine-input pairs $N_i(x_{i,b})$ simulated so far, and the k-th such machine fixes at most $2^{(n+4)k} \cdot 2^n$ entries. Therefore, the number of entries fixed by some previous $N_i(x_{i,b})$ is at most

$$\sum_{k=1}^{K} 2^{(n+4)k} \cdot 2^n \leq 2^{O(n^3)}.$$

It follows that all but $2^{O(n^3)}$ entries in the n^4-th slice of $\mathcal{P}$ are not fixed yet. Let $b \in \{0,1\}$ be the output bit of $M_i(1^n)$, then we set $L_{\mathcal{P}}(1^n) = 1 - b$ by fixing every unfixed entry in the n^4-th slice of $\mathcal{P}$ to be $1-b$. We have that on input 1^n, $L_{\mathcal{P}}$ satisfies the BPP promise, and M_i fails to solve $L_{\mathcal{P}}$.

Clear-up. At the end of stage n, we fix every unfixed input $\mathcal{Q}(x, r)$ on the n-th slice to be 0. It is easy to see that there are at most $2n$ entries in the n-th slice of $\mathcal{Q}(x, r)$ that returns 1.

During the n-th stage, the number of entries we fixed beyond the $(n+1)$-st slice of $\mathcal{Q}$ is at most

$$2^n + \sum_{m=n+1}^{2^n} 2^n \cdot 10^4 \cdot m^3 \leq 2^{5n}.$$

Thus the total number of entries we fixed beyond the $(n+1)$-st slice of $\mathcal{Q}$ is at most $2^{6n} + 2^{5n} < 2^{6(n+1)}$. ◀

Unfortunately, we do not know how to compute the above oracles in EXPH. The reason is that when we diagonalize N_i on input length n, the number of heavy entries $\mathcal{P}(r)$ we need to fix is $2^{(n'+4)K} \cdot 2^n = 2^{\mathrm{poly}(n')}$. Since n' might be exponentially large compared to n, this upper bound might be doubly exponential. Nevertheless, we show that under the assumption that

$$\mathsf{SAT} \in \mathsf{DTIME}[n \cdot \mathrm{polylog}(n)] \cap \mathsf{NC}, \tag{*}$$

the oracles $\mathcal{P}, \mathcal{Q}$ in Theorem 11 can be computed in PSPACE. It follows that

▶ **Corollary 45.** *Any* PSPACE*-relativizing proof of*

$$\mathsf{PSPACE} \not\subseteq \mathsf{i.o.BPP} \implies \mathsf{BPP} \subseteq \mathsf{heur\text{-}SUBEXP}$$

would also imply a breakthrough lower bound for SAT*, i.e., refuting Equation* (*).

▶ **Theorem 12.** *Suppose that* $\mathsf{SAT} \in \mathsf{DTIME}[n \cdot \mathrm{polylog}(n)] \cap \mathsf{NC}$*. Then there is an oracle* $\mathcal{O}$ *satisfying the conclusions of Theorem 11 that can be computed in polynomial space.*

We sketch some intuition before presenting the full proof of Theorem 12. The bottleneck of putting $\mathcal{O}$ into EXPH is that we do not have enough space to store all the heavy entries $\mathcal{P}(r)$ fixed by a probabilistic machine N_i, as there might be doubly-exponentially many of them. Therefore, whenever we simulate a machine N_i and it asks a query $\mathcal{P}(r)$, we have to compute from scratch whether $\mathcal{P}(r)$ was fixed by a previous machine. If we have an oracle $\textsc{Heavy}$ that given r, decides whether $\mathcal{P}(r)$ was already fixed, then we can simulate N_i in exponential time with a constant number of alternations (using Theorem 24). However, there are $\mathrm{poly}(n)$ machines N_j simulated before N_i, and each time we invoke $\textsc{Heavy}$, we need to simulate these machines to see if any of them has already fixed $\mathcal{P}(r)$. It follows that the simulation of N_i actually requires $\mathrm{poly}(n)$ alternations and $2^{\mathrm{poly}(n)}$ time. Still, under a strong enough assumption such as (*), we can simulate these $\mathrm{poly}(n)$ alternations in $2^{\mathrm{poly}(n)}$ time.

Proof of Theorem 12. Consider the construction in Theorem 11, with the only difference that we do not calculate the precise probabilities; instead, we compute their approximations in PH. Since $\mathsf{SAT} \in \mathsf{NC}$, we have $\mathsf{EXP} = \mathsf{PSPACE}$ by padding, thus it suffices to construct the oracles in $2^{\mathrm{poly}(n)}$ time.

The most involved part of this proof is to compute the following function $\textsc{Heavy}(n, \langle \mathcal{P}, \mathcal{Q} \rangle, i, b, x_{i,b}, r)$, which indicates whether $\mathcal{P}(r)$ is a heavy query of the probabilistic machine N_i on input $x_{i,b}$. Here, the input of $\textsc{Heavy}$ consists of the description of oracles $\mathcal{P}$ and $\mathcal{Q}$, an integer $i \leq n$, an input $x_{i,b} \in \{0,1\}^n$, and a query $r \in \{0,1\}^{(n_{i'})^4}$ (where $n_{i'}$ is the smallest element in the sequence $\{n_i\}$ such that $n_{i'} \geq n$). We may assume $|r| \leq 2^n$ as otherwise N_i could never query $\mathcal{P}(r)$. The oracle description $\langle \mathcal{P}, \mathcal{Q} \rangle$ will contain the following information:

- We record a table of all entries in $\mathcal{Q}$ that are fixed; there are at most $\sum_{m \le n} 2^m + 2^{6n} \le 2^{O(n)}$ such entries. For each entry, we also record a timestamp indicating when this entry is fixed.
- We record all entries of $\mathcal{P}$ up to the $n_{i'-1}$-th slice (but we do *not* include any information of the $n_{i'}$-th slice of $\mathcal{P}$); the description length of $\mathcal{P}$ is also at most $2^{O(n)}$.

Note that the input length of HEAVY is at most $2^{O(n)}$. Instead of requiring HEAVY to decide exactly whether $\mathcal{P}(r)$ is a heavy query, we only require a 2-approximation: if $\mathcal{P}(r)$ is queried by $N_i(x_{i,b})$ w.p. at least $2^{-(n'+4)K}$ then HEAVY returns 1, while if $\mathcal{P}(r)$ is queried w.p. at most $2^{-(n'+4)K}/2$ then HEAVY returns 0.

Let k be the number of machine-input pairs we have simulated before $N_i(x_{i,b})$ (inclusive). That is,

$$k := \sum_{m<n} (2m) + 2i + b + 1 \le O(n^2).$$

We will show that:

▷ Claim 46. If $\mathsf{SAT} \in \mathsf{DTIME}[n \cdot \mathrm{polylog}(n)]$, then we can compute HEAVY in deterministic $O(2^{k^2})$ time.

Proof. Consider simulating (a random computational branch of) $N_i(x_{i,b})$ while answering the oracle queries of N_i accordingly.[12] Note that we have not recorded the $n_{i'}$-th slice of $\mathcal{P}$ in our description $\langle \mathcal{P}, \mathcal{Q} \rangle$. As a consequence, whenever $N_i(x_{i,b})$ asks a query $\mathcal{P}(r_q)$ where $|r_q| = n_{i'}$, we need to recursively call HEAVY to decide whether this query was already fixed and which value it was fixed to. We enumerate every $n_{i'-1} < \tilde{n} \le n$, every machine N_j ($j \le \tilde{n}$), and every bit b'. Suppose that we feed the machine N_j with the input $x'_{j,b'} \in \{0,1\}^{n_{i'}}$.[13] Using the timestamps recorded in the table $\mathcal{Q}$, we can recover the state of the oracle $\mathcal{Q}$ before the simulation of $N_j(x'_{j,b'})$; we call this oracle $\mathcal{Q}'$. (The portion of oracle $\mathcal{P}$ up to the $n_{i'-1}$-th slice remains the same.) Then, we call $\textsc{Heavy}(\tilde{n}, \langle \mathcal{P}, \mathcal{Q}' \rangle, j, b', x'_{j,b'}, r_q)$ to see if this query is a heavy query fixed by N'_j. If it is, then we return $\mathcal{P}(r_q) := b'$; otherwise we search through the next machine. If the query $\mathcal{P}(r)$ was not fixed before and was asked in this simulation. then we return 1; otherwise we return 0.

The above argument implies that HEAVY can be computed recursively in the following sense. There is a machine $V_{\textsc{Heavy}}$ that gets $\mathsf{input} := (n, \langle \mathcal{P}, \mathcal{Q} \rangle, i, b, x_{i,b}, r)$ and some randomness $z \in \{0,1\}^{2^{O(n)}}$ (denoting a random computational branch of $N_i(x_{i,b})$), runs in $2^{O(n)}$ time with $\mathrm{poly}(n)$ invocations of the HEAVY oracle, and outputs 0 or 1. Let $\varepsilon := 2^{-(n'+4)/K}$, we want that $\textsc{Heavy}(\mathsf{input}) = 1$ if $\Pr_z[V_{\textsc{Heavy}}(\mathsf{input}; z) = 1] \ge \varepsilon$, and $\textsc{Heavy}(\mathsf{input}) = 0$ if $\Pr_z[V_{\textsc{Heavy}}(\mathsf{input}; z) = 1] \le \varepsilon/2$.

We can use induction to show that HEAVY can be computed in (deterministic) $O(2^{k^2})$ time. In particular, by the induction hypothesis, each call of the HEAVY oracle made by $V_{\textsc{Heavy}}$ on input can be computed in $O(2^{(k-1)^2})$ time. Therefore, $V_{\textsc{Heavy}}$ runs in at most $2^{(k-1)^2} \cdot \mathrm{poly}(k)$ time. By Theorem 24, there is a $\Sigma_5\mathsf{TIME}[2^{(k-1)^2} \cdot \mathrm{poly}(k)]$ time machine $\widetilde{\textsc{Heavy}}$ such that $\widetilde{\textsc{Heavy}}(\mathsf{input}) = 1$ if $\Pr_z[V_{\textsc{Heavy}}(\mathsf{input}; z) = 1] \ge \varepsilon$ and $\widetilde{\textsc{Heavy}}(\mathsf{input}) = 0$ if $\Pr_z[V_{\textsc{Heavy}}(\mathsf{input}; z) = 1] \le \varepsilon/2$. By our hypothesis that $\mathsf{SAT} \in \mathsf{DTIME}[n \cdot \mathrm{polylog}(n)]$, there is a deterministic machine HEAVY that runs in $2^{(k-1)^2} \cdot \mathrm{poly}(k) < 2^{k^2}$ time and decides the same language as $\widetilde{\textsc{Heavy}}$. ◁

[12] That is, whenever N_i asks a query $\mathcal{P}(r)$ or $\mathcal{Q}(x, r)$ that is fixed, we answer accordingly. Whenever N_i asks a query $\mathcal{P}(r)$ that is not fixed, if $|r| = (n')^4$ then we return b, otherwise we return 0. Whenever N_i asks a query $\mathcal{Q}(x, r)$ that is not fixed, we return 0.

[13] We assume that there is an easy and deterministic way of assigning the inputs $x_{j,b'}$ for each pair (j, b').

Now we show that given an integer N, it is possible to print the N-th slice of $\mathcal{P}$ and $\mathcal{Q}$ in deterministic $2^{\mathrm{poly}(N)}$ time. We simulate the stages n for $n = 1, 2, \cdots$, where during each stage we need to diagonalize against probabilistic machines $N_1, N_2, \cdots$. If $n = n_i$ for some $i \in \mathbb{N}$, then we also need to diagonalize against M_i.

- For each N_i, we can list the set of its heavy queries of the form $\mathcal{Q}(x, r)$ in deterministic $2^{\mathrm{poly}(n)}$ time with a $\mathsf{PH}^{\textsc{Heavy}}$ oracle. Note that Heavy can be decided in time quasi-polynomial in its input length; also recall we assumed that $\mathsf{SAT} \in \mathsf{P}$. Thus we can enumerate the heavy queries of the form $\mathcal{Q}(x, r)$ in deterministic $2^{\mathrm{poly}(n)}$ time (without additional oracles). We fix all these queries. (Note that we do not fix the heavy queries of the form $\mathcal{P}(r)$; instead, we use the oracle Heavy to decide whether a query $\mathcal{P}(r)$ is fixed.) Then we use the PH oracle to estimate the probability that $N_i(x_{i,b})$ outputs 1. If the probability is small then we pick some r such that $\mathcal{Q}(x_{i,b}, r)$ is unfixed, and fix this entry to 1; otherwise we do nothing.
- Before we diagonalize against M_i, we spend $2^{\mathrm{poly}(n)}$ time to retrieve the list of fixed entries in the n^4-th slice of $\mathcal{P}$ from the oracle Heavy. Then we simulate M_i, and for every unfixed query, we fix it to be 0. Let $b \in \{0, 1\}$ be the output of M_i, then we set every unfixed entry in the n^4-th slice of $\mathcal{P}$ to be $1 - b$.

It is easy to see that the above process runs in $2^{\mathrm{poly}(N)}$ time. ◀

▶ Remark 47. A beautiful line of work [24, 53, 25, 22, 76, 77, 15, 58] investigated time-space trade-off lower bounds for SAT. Lower bounds proved in these works come in two flavors: "SAT cannot be solved by a machine with certain time and space bounds *simultaneously*", or "SAT either cannot be solved in some time bound, or cannot be solved in some space bound (even by two different machines)". The state-of-the-art lower bounds of the first flavor is that SAT cannot be solved in n^c time and $n^{o(1)}$ space *simultaneously*, for every $c < 2\cos(\pi/7) \approx 1.801$ [77]; note that such lower bounds do not contradict (*). The state-of-the-art lower bounds of the second flavor is that SAT either requires more than $n \cdot \mathrm{polylog}(n)$ time or requires $\log^{2-o(1)} n$ depth [58]. To the best of our knowledge, it is still an open question to disprove (*).

5 Barriers for Explicit Constructions

5.1 PSPACE-Relativizing Pseudodeterministic Constructions

In this section, we show that the previous results on pseudodeterministic constructions are PSPACE-relativizing. A property Q is *dense* if for every $n \in \mathbb{N}$, $|Q \cap \{0,1\}^n| \geq 2^n/\mathrm{poly}(n)$. For every dense property Q computable in polynomial time, Oliveira and Santhanam [62] presented a pseudodeterministic algorithm that on input 1^n, outputs a canonical element in Q_n with high probability. Their pseudodeterministic algorithm runs in subexponential time, is zero-error (i.e., the algorithm never outputs any element besides $\perp$ ("failure") and the canonical one), and is correct on infinitely many input lengths n. We verify that their argument holds relative to every oracle $O \in \mathsf{PSPACE}$.

▶ **Proposition 48** (Formal Version of Proposition 13)**.** *Let $\mathcal{O} \in \mathsf{PSPACE}$. Then for every $\varepsilon > 0$ and every dense property $Q \in \mathsf{P}^{\mathcal{O}}$, there exist a zero-error pseudodeterministic $\mathcal{O}$-oracle algorithm A with running time 2^{n^ε} and an infinite sequence $\{x_{n_i}\}_{i \in \mathbb{N}}$ such that $x_{n_i} \in Q \cap \{0,1\}^{n_i}$ for each $i \in \mathbb{N}$ and that*

$$\Pr_A\left[A^{\mathcal{O}}(1^{n_i}) = x_{n_i}\right] \geq 3/4.$$

Proof Sketch. We follow the analysis from [62].

Let $c \geq 1$ and $Q \in \mathsf{DTIME}^{\mathcal{O}}[n^c]$ be a dense property. It suffices to show an HSG $\{H_n\}_n$ of seed length n^ε such that H_n can be computed pseudodeterministically with zero error in time 2^{n^ε} with oracle access to $\mathcal{O}$, and that for infinitely many n, H_n hits Q on input length n. This is because we can then enumerate all $z \in \{0,1\}^{n^\varepsilon}$ and find the first z such that $Q(H_n(z)) = 1$. Such a z exists since Q is a dense property and H_n hits Q.

First of all, using the "easy witness" method [45], one can show that there exists a family of sets $\{H_n^{\mathsf{easy}} \subseteq \{0,1\}^n\}_n$, such that each H_n^{easy} is computable deterministically in time 2^{n^ε} with oracle access to $\mathcal{O}$, and that if Q avoids $\{H_n^{\mathsf{easy}}\}_n$ on all but finitely many input length n, then $\mathsf{BPP}^{\mathcal{O}} = \mathsf{ZPP}^{\mathcal{O}}$. Roughly speaking, H_n^{easy} contains the truth tables of all $\mathcal{O}$-circuits $C\colon \{0,1\}^{\log n} \to \{0,1\}$ of size at most $n^{\varepsilon/10}$. If Q avoids H_n^{easy}, then we can obtain, with high probability and without error, an n-bit truth table that has $\mathcal{O}$-circuit complexity at least $n^{\varepsilon/10}$, by randomly picking an n-bit string that is accepted by Q. Such hard truth tables can then be used to derandomize $\mathsf{BPP}^{\mathcal{O}}$. (See [62, Proof of Lemma 2] for details.)

If $\mathsf{BPP}^{\mathcal{O}} \neq \mathsf{ZPP}^{\mathcal{O}}$, $\{H_n^{\mathsf{easy}}\}_n$ will give a valid HSG. Now assume $\mathsf{BPP}^{\mathcal{O}} = \mathsf{ZPP}^{\mathcal{O}}$. It suffices to construct an infinitely often HSG that can be computed pseudodeterministically with oracle access to $\mathcal{O}$, since it can be made zero-error using the assumption $\mathsf{BPP}^{\mathcal{O}} = \mathsf{ZPP}^{\mathcal{O}}$.

We consider two cases. Suppose $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$. Then the existence of a valid HSG follows easily from Lemma 26. Now suppose $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$. We can also use the following hardness-to-randomness construction to obtain a valid HSG.

▶ **Theorem 49** ([42, 70, 50]). *Let $\mathcal{O}$ be any oracle. If $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$, then for every $b, c \geq 1$, there is a sequence $\{G_\ell\}_{\ell \geq 1}$, where $G_\ell\colon \{0,1\}^\ell \to \{0,1\}^{\ell^b}$ is computable in time $2^{O(\ell)}$, such that for every language $L \in \mathsf{DTIME}^{\mathcal{O}}[n^c]$, there are infinitely many ℓ such that*

$$\left| \Pr_{x \sim \{0,1\}^{\ell^b}}[L(x) = 1] - \Pr_{z \sim \{0,1\}^{\ell}}[L(G_\ell(z)) = 1] \right| \leq \frac{1}{10}.$$

This completes the proof of Proposition 48. ◀

5.2 Bounded-Relativization Barriers for Explicit Constructions

5.2.1 Barriers for Almost-Everywhere Pseudodeterministic Constructions

We show that EXPH-relativizing techniques cannot prove *almost-everywhere* pseudodeterministic constructions, even if the construction algorithms are allowed $2^{o(n)}$ time. The underlying oracle uses the query complexity lower bounds proved by Goldwasser, Impagliazzo, Pitassi, and Santhanam [31] *in a black-box fashion.* (We thank Rahul Santhanam for pointing out that the query complexity lower bounds imply an EXPH-computable oracle without almost-everywhere pseudodeterministic constructions in a black-box fashion.)

▶ **Definition 50.** *A* pseudodeterministic *decision tree for a search problem S is a distribution $\mathcal{T}$ over decision trees with the following property: For every input x, there is a* canonical *value o such that with probability at least $2/3$, $\mathcal{T}(x) = o$. Let $\mathsf{psP}^{\mathsf{dt}}(S)$ denote the minimum depth of any pseudodeterministic decision tree for S.*

Consider the following search problem denoted as FIND1. The input is a string $x \in \{0,1\}^N$ where it is guaranteed that there are at least $N/2$ bits in x that are equal to 1. The problem is to find some $i \in [N]$ such that $x_i = 1$. The following lower bound on the pseudodeterministic query complexity of FIND1 was proved in [31]:

▶ **Theorem 51** ([31]). $\mathsf{psP}^{\mathsf{dt}}(\textsc{Find1}) = \Omega(\sqrt{N})$.

We use this result as a black box and construct an oracle $\mathcal{O} \in \mathsf{EXPH}$ without almost-everywhere pseudodeterministic constructions running in $2^{o(n)}$ time:

▶ **Theorem 52.** *There is an oracle $\mathcal{O} \in \mathsf{EXPH}$ and a dense property $Q \in \mathsf{P}^{\mathcal{O}}$ such that the following holds. For every randomized (and purportedly pseudodeterministic) algorithm $\mathcal{A}$ that runs in $2^{o(n)}$ time with oracle access to $\mathcal{O}$ and every infinite sequence of outputs $\{x_n\}_{n\in\mathbb{N}}$ where each $x_n \in Q \cap \{0,1\}^n$, there are infinitely many input lengths $n \in \mathbb{N}$ such that*

$$\Pr[\mathcal{A}(1^n) = x_n] \leq 3/4.$$

Proof. Let $M_1, M_2, \cdots$ be an enumeration of probabilistic machines running in $2^{0.1n}$ time. Let n_1 be a large enough constant and define the sequence $\{n_i\}$ where $n_i = 2^{n_{i-1}}$ for each $i \geq 2$. We let $Q = \mathcal{O}$ itself be the dense property without pseudodeterministic construction algorithms. In particular, we want that for every $i \in \mathbb{N}$, $M_i(1^{n_i})$ fails to generate a canonical string $x \in \mathcal{O} \cap \{0,1\}^{n_i}$. Naturally, our oracle construction proceeds in stages, where for each $i \in \mathbb{N}$, the n_i-th slice of $\mathcal{O}$ is constructed by carefully diagonalizing against M_i; on the other hand, if $n \in \mathbb{N}$ is not in the sequence $\{n_i\}$, then we simply let $\mathcal{O}$ accept every string of length n.

Now we show how to construct the n_i-th slice of $\mathcal{O}$ in the i-th stage. The idea is simple: construct the (purportedly pseudodeterministic) decision tree $\mathcal{T}$ corresponding to M_i, find an exponential-length input tt of Find1 on which $\mathcal{T}$ fails, and let the truth table of the n_i-th slice of $\mathcal{O}$ be tt. More precisely:

- **Converting M_i into a distribution of decision trees.** We define the distribution of decision trees $\mathcal{T}$. The input to Find1 has length 2^{n_i} and is considered as the truth table of the n_i-th slice of $\mathcal{O}$. Note that since $2^{0.1n_i} \ll n_{i+1}$, for every $n' \leq 2^{0.1n_i}$ such that $n' \neq n_i$, the n'-th slice of $\mathcal{O}$ are fixed. To sample a (deterministic) decision tree T from $\mathcal{T}$, sample a sequence of random coins fed to M_i and simulate M_i on the oracle $\mathcal{O}$. Whenever M_i makes a query $\mathcal{O}(x)$, if $|x| = n_i$, then the decision tree asks the x-th bit of our input; otherwise we return the already-fixed value of $\mathcal{O}(x)$.
- **Invoking the lower bound.** Note that $\mathcal{T}$ only makes $2^{0.1n_i} < o(\sqrt{2^{n_i}})$ queries. By Theorem 51, there exists an input $tt \in \{0,1\}^{2^{n_i}}$ such that $\mathcal{T}$ fails to solve Find1 on input tt pseudodeterministically. That is, for every valid output x such that $tt_x = 1$, $\Pr[\mathcal{T}(tt) = x] < 2/3$. Let the truth table of the n_i-th slice of $\mathcal{O}$ to be such a tt.
  It is easy to see that for every probabilistic machine $\mathcal{A}$ that runs in $2^{o(n)}$ time with oracle access to $\mathcal{O}$, and every infinite sequence of outputs $\{x_n\}_{n\in\mathbb{N}}$ where each $x_n \in \mathcal{O} \cap \{0,1\}^n$, for every i such that $\mathcal{A} = M_i$, the probability that $\mathcal{A}(1^{n_i})$ outputs x_{n_i} is at most $3/4$.
- **Complexity of $\mathcal{O}$.** Let $\mathcal{O}_{\leq n_{i-1}}$ denote the description of the oracle $\mathcal{O}$ up to the n_{i-1}-th slice (note that $\mathcal{O}_{\leq n_{i-1}}$ can be described in $\mathrm{poly}(2^{n_{i-1}}) \leq \mathrm{poly}(n_i)$ bits), and $x \in \{0,1\}^{n_i}$. Note that $\mathcal{O}_{\leq n_{i-1}}$ and tt together defines the oracle $\mathcal{O}$ up to input length $n_{i+1} - 1$, so the behavior of $M_i^{\mathcal{O}}$ is completely determined by $M_i, \mathcal{O}_{\leq n_{i-1}}$, and tt. Let $\mathsf{ProbEst}(\mathcal{O}_{\leq n_{i-1}}, tt, M_i, x)$ be an oracle such that

$$\Pr[M_i^{\mathcal{O}}(1^{n_i}) = x] > 3/4 \implies \mathsf{ProbEst}(\mathcal{O}_{\leq n_{i-1}}, tt, M_i, x) = 1,$$
$$\Pr[M_i^{\mathcal{O}}(1^{n_i}) = x] < 2/3 \implies \mathsf{ProbEst}(\mathcal{O}_{\leq n_{i-1}}, tt, M_i, x) = 0.$$

  By Theorem 23, the oracle ProbEst can be implemented in $\mathsf{DTIME}[2^{O(n_i)}]^{\mathsf{PH}}$. Consider the following algorithm with oracle access to ProbEst that prints a truth table tt. The algorithm maintains the prefix of a truth table which is initially the empty string, and extends this prefix bit by bit. Suppose that we have a prefix of length ℓ, denoted as $tt' \in \{0,1\}^{\ell}$. To fix the $(\ell+1)$-st bit of tt', we check if there exists a truth table tt

such that (1) for every $x \in \{0,1\}^{n_i}$, $\mathsf{ProbEst}(\mathcal{O}_{\leq n_{i-1}}, tt, M_i, x) = 0$, and (2) $tt' \circ 0$ (tt' concatenated with a bit 0) is a prefix of tt. If there is such a tt, then we append 0 to the end of tt'; otherwise we append 1 to the end of tt'.
It is clear that the algorithm always outputs a truth table tt on which M_i fails. Since $\mathsf{ProbEst}$ runs in $\mathsf{DTIME}[2^{O(n_i)}]^{\mathsf{PH}}$, the whole algorithm also runs in $\mathsf{DTIME}[2^{O(n_i)}]^{\mathsf{PH}}$. Using this algorithm, it is easy to construct the oracle $\mathcal{O}$ in EXPH. ◀

▶ **Corollary 53.** *If there is a* PSPACE*-relativizing proof that for every dense property Q computable in polynomial time, there is a pseudodeterministic construction for Q running in $2^{o(n)}$ time that is correct on* almost every input length*, then* $\mathsf{L} \neq \mathsf{NP}$*.*

5.2.2 Barriers for Deterministic Constructions and Lower Bounds for MKtP

In this section, we construct an oracle in EXP relative to which there is no deterministic construction in $2^n/n^{\omega(1)}$ time (even infinitely often), showing that any non-trivial deterministic constructions using PSPACE-relativizing techniques would separate PSPACE from EXP.

As an interesting corollary, it is easy to approximate the Kt complexity to an $(1+\varepsilon)$ factor in this oracle world in deterministic $n^{O(\log n)}$ time. Therefore, although it is EXP-hard to approximate the Kt complexity under $(\mathsf{P}/_{\mathrm{poly}})$-truth-table reductions and NP-Turing reductions [5], any PSPACE-relativizing proof that the Kt complexity requires deterministic $n^{\omega(\log n)}$ time to approximate would separate PSPACE from EXP.

▶ **Definition 54.** *For a constant $\varepsilon > 0$,* $\mathsf{Gap}_\varepsilon\mathsf{MKtP}$ *is defined to be the promise problem $(\mathcal{YES}_n, \mathcal{NO}_n)_{n\in\mathbb{N}}$, where*

$$\begin{aligned} \mathcal{YES}_n &:= \{(x,s) \in \{0,1\}^n \times \mathbb{N} : \mathrm{Kt}(x) \leq s\}, \\ \mathcal{NO}_n &:= \{(x,s) \in \{0,1\}^n \times \mathbb{N} : \mathrm{Kt}(x) > (1+\varepsilon)\cdot s\}. \end{aligned}$$

Our proof relies heavily on the equivalence between non-trivial derandomization and the hardness of $\mathsf{Gap}_\varepsilon\mathsf{MKtP}$ [34]. It is not hard to construct an oracle under which non-trivial derandomization is impossible, which means that a dense subset of the complement of MKtP can be accepted by an efficient algorithm A. Then, we use the worst-case to average-case reduction of [33] to transform A into a worst-case approximation algorithm for $\mathsf{Gap}_\varepsilon\mathsf{MKtP}$.

▶ **Theorem 55.** *There exists an oracle $\mathcal{O} \in \mathsf{EXP}$ such that*

1. *there is a dense property $Q \in \mathsf{P}^{\mathcal{O}}$ such that every deterministic algorithm that runs in time $2^n/n^{\omega(n)}$ fails to find a string in $Q \cap \{0,1\}^n$ on almost every input length n, and*
2. $\mathsf{Gap}_\varepsilon\mathsf{MKtP}^{\mathcal{O}} \in \mathsf{DTIME}^{\mathcal{O}}[n^{O(\log n)}]$ *for every constant $\varepsilon > 0$.*

To show the worst-case to average-case reduction, we use the following pseudorandom generator construction.

▶ **Lemma 56** (cf. [33])**.** *For any $d, m \leq 2n, \varepsilon > 0$, there exists a "pseudorandom generator construction"*

$$G\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

such that for any distinguisher $D : \{0,1\}^m \to \{0,1\}$, if

$$\Pr_{w\sim\{0,1\}^m}[D(w)=1] - \Pr_{z\sim\{0,1\}^d}[D(G(x,z))=1] \geq \frac{1}{2},$$

then

$$\mathrm{K}^{\mathrm{poly}(n),D}(x) \leq \exp(\ell^2/d) \cdot m + d + O(\log n),$$

where $\ell = O(\log n)$. *Moreover,* $G(x,z)$ *can be computed in time* $\mathrm{poly}(n)$.

Proof of Theorem 55. We construct an oracle $\mathcal{O} \in \mathsf{EXP}$ under which a dense subset of MKtP is in P. We start with $n := 1$, $\mathcal{O} := \emptyset$, and $F := \emptyset$, where F is a set of "frozen" strings. The construction of $\mathcal{O}$ in Stage n is as follows. Let $\mathcal{O}_n$ denote the state of the oracle $\mathcal{O}$ at the beginning of stage n. Let E_n be the set of all strings $x \in \{0,1\}^n$ such that $\mathrm{Kt}^{\mathcal{O}_n}(x) \leq n - c\log n$, where c is a sufficiently large constant (e.g., $c := 3$). Let F_n be the set of all strings $q \in \{0,1\}^*$ such that there exist $k \in [n - c\log n]$ and a description $d \in \{0,1\}^k$ of a Turing machine such that the universal Turing machine $U^{\mathcal{O}_n}$ on input d makes the query q in time $2^{n-k-c\log n}$. Then, we update $F := F \cup F_n$ and $\mathcal{O}_{n+1} := \mathcal{O}_n \cup (\{0,1\}^n \setminus (E_n \cup F))$ and move on to the next stage $n+1$. This completes the description of the oracle $\mathcal{O} := \bigcup_{n\in\mathbb{N}} \mathcal{O}_n$. It is easy to observe that $\mathcal{O} \in \mathsf{EXP}$.

The oracle $\mathcal{O}$ is a dense subset of $\left\{x : \mathrm{Kt}^{\mathcal{O}}(x) > n - c\log n\right\}$ in the following sense:

1. $x \notin \mathcal{O}$ for every string $x \in \{0,1\}^n$ such that $\mathrm{Kt}^{\mathcal{O}}(x) \leq n - c\log n$.
2. $x \in \mathcal{O}$ for at least half of the strings $x \in \{0,1\}^n$.

To see the first property, it suffices to show that if $x \in \{0,1\}^n$ satisfies $\mathrm{Kt}^{\mathcal{O}}(x) \leq n - c\log n$, then $\mathrm{Kt}^{\mathcal{O}_n}(x) \leq n - c\log n$ (which implies $x \in E_n$ and thus $x \notin \mathcal{O}$). Assuming that there exists a string d such that $U^{\mathcal{O}}(d)$ outputs x in time $2^{n-c\log n-|d|}$, we claim that $U^{\mathcal{O}_n}(d)$ also outputs x. If not, there exists a query in $\mathcal{O} \setminus \mathcal{O}_n$ made by $U^{\mathcal{O}}(d)$. Let q be the first query in $\mathcal{O} \setminus \mathcal{O}_n$. The same query is also queried during the computation of $U^{\mathcal{O}_n}(d)$. Thus, $q \in F_n$, which implies $q \notin \mathcal{O}$, which is a contradiction.

To see the second property, we bound the size of $E_n \cup F$ at Stage n. The size of E_n is at most $\sum_{k\in[n-c\log n]} 2^k \leq 2^n/n^{c-1}$. The number $|F_n|$ of frozen strings in Stage n is at most

$$\sum_{k\in[n-c\log n]} 2^k \cdot 2^{n-k-c\log n} \leq 2^n/n^{c-1}.$$

Thus, the number $|F|$ of frozen strings until Stage n is at most $\sum_{n'=1}^{n} |F_{n'}| \leq 2^n/n^{c-2}$. Overall, we obtain

$$|\mathcal{O} \cap \{0,1\}^n| \geq 2^n - |E_n| - |F| \geq \frac{1}{2} \cdot 2^n.$$

Let $Q := \mathcal{O}$. We prove that $Q \in \mathsf{P}^{\mathcal{O}}$ is a dense property such that every $2^n/n^{\omega(1)}$-time deterministic algorithm A fails to find a string in $Q \cap \{0,1\}^n$ on input 1^n. By the definition of Kt, we have $\mathrm{Kt}^{\mathcal{O}}(A(1^n)) \leq O(\log n) + \log(2^n/n^{\omega(1)}) \leq n - \omega(\log n)$. Thus, the output $A(1^n)$ of A is not in $\mathcal{O}$.

We now use the worst-case to average-case reduction of [34] to obtain an approximation algorithm $A^{\mathcal{O}}$ for $\mathsf{Gap}_\varepsilon\mathsf{MKtP}$. Let $x \in \{0,1\}^n$ and $s \in \mathbb{N}$ be an instance of $\mathsf{Gap}_\varepsilon\mathsf{MKtP}$. Let $d = k \cdot O(\log^2 n)$, where $k = k(\varepsilon)$ is a sufficiently large constant that will be chosen depending on $\varepsilon > 0$. We may assume without loss of generality that $O(k^2 \cdot \log^2 n) \leq s \leq n + O(\log n)$ because whether $\mathrm{Kt}(x) \leq O(k^2 \cdot \log^2 n)$ or not can be decided in time $n^{O(k^2 \cdot \log n)}$ by an exhaustive search. Let m be a parameter chosen later. Let G be the pseudorandom generator construction of Lemma 56. The algorithm $A^{\mathcal{O}}$ accepts (x,s) if and only if $G(x,z) \notin \mathcal{O}$ for every $z \in \{0,1\}^d$. The running time of $A^{\mathcal{O}}$ is $2^d\mathrm{poly}(n) = n^{O(\log n)}$.

We prove the correctness of $A^{\mathcal{O}}$. Assume that $\mathrm{Kt}^{\mathcal{O}}(x) \leq s$. Then, for every $z \in \{0,1\}^d$, we have

$$\mathrm{Kt}^{\mathcal{O}}(G(x,z)) \leq \mathrm{Kt}^{\mathcal{O}}(x) + O(d + \log n) \leq s + O(d + \log n) \leq m - c\log m,$$

where the last inequality holds by choosing a sufficiently large $m = s + O(d + \log n)$. It follows that $G(x,z) \notin \mathcal{O}$ and that $A^{\mathcal{O}}$ accepts (x,s). Conversely, assume that $A^{\mathcal{O}}$ accepts (x,s), which means that $\Pr_{z\sim\{0,1\}^d}[\mathcal{O}(G(x,z)) = 1] = 0$. We claim that $\mathrm{Kt}^{\mathcal{O}}(x) \leq (1+\varepsilon)\cdot s$. Since $\mathcal{O}$ is dense, we have

$$\Pr_{w\sim\{0,1\}^m}[\mathcal{O}(w) = 1] - \Pr_{z\sim\{0,1\}^d}[\mathcal{O}(G(x,z)) = 1] \geq \frac{1}{2}.$$

By Lemma 56, we obtain

$$\mathsf{K}^{\mathrm{poly}(n),\mathcal{O}}(x) \leq \exp(O(\log^2 n)/d)\cdot m + d + O(\log n) \leq \exp(1/k)\cdot m + O(k\cdot\log^2 n).$$

In particular,

$$\mathrm{Kt}^{\mathcal{O}}(x) \leq \exp(1/k)\cdot m + O(k\cdot\log^2 n) \leq (1+O(1/k))\cdot m + O(k\cdot\log^2 n) \leq (1+\varepsilon)\cdot m,$$

where we choose a sufficiently large $k := O(1/\varepsilon)$ and use that $m \geq O(k^2\cdot\log^2 n)$ in the last inequality. ◀

6 Barriers for Circuit Lower Bounds for Merlin–Arthur Classes

Buhrman, Fortnow, and Thierauf [13] showed that $\mathsf{MA\text{-}EXP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$ and Santhanam [65] proved that $\mathsf{MA}/_1 \not\subseteq \mathsf{SIZE}[n^k]$ for every constant k. Their techniques rely heavily on win-win analysis and thus only yield circuit lower bounds that hold infinitely often.

This section presents an EXPH-relativizing barrier for proving almost-everywhere versions of these lower bounds. In Section 6.1, we show that Santhanam's circuit lower bound is PSPACE-relativizing. In Section 6.2, we construct an EXPH oracle under which the almost-everywhere lower bound fails.

6.1 PSPACE-Relativizing Circuit Lower Bounds for Merlin–Arthur Classes

▶ **Theorem 57.** *Let $\mathcal{O} \in \mathsf{PSPACE}$. For every $k \in \mathbb{N}$, $\mathsf{MA}^{\mathcal{O}}/_1 \not\subseteq \mathsf{SIZE}^{\mathcal{O}}[n^k]$.*

We need the notion of an instance checker.

▶ **Definition 58** (Instance-Checkable Languages)**.** *A language L is said to be same-length instance-checkable if there is a probabilistic polynomial-time oracle machine $I^{(-)}$ with output in $\{0,1,\bot\}$ such that for any input x:*

1. *$I^{(-)}$ only makes oracle queries of length $|x|$.*
2. *$I^L(x) = L(x)$ with probability 1.*
3. *$I^A(x) \in \{L(x),\bot\}$ with probability at least $2/3$ for any oracle A.*

▶ **Lemma 59** ([70, 26])**.** *There is a PSPACE-complete language L_{hard} that is same-length instance-checkable.*

We now show Theorem 57. The proof follows closely to that of [65], which employs a win-win argument. Let L_{hard} be the language from Lemma 59. We will consider two cases: $L_{\mathsf{hard}} \in \mathsf{SIZE}^{\mathcal{O}}[\mathrm{poly}]$ and $L_{\mathsf{hard}} \notin \mathsf{SIZE}^{\mathcal{O}}[\mathrm{poly}]$.

▶ **Lemma 60.** *Let $\mathcal{O} \in \mathsf{PSPACE}$. If $L_{\mathsf{hard}} \in \mathsf{SIZE}^{\mathcal{O}}[\mathrm{poly}]$, then for every $k \in \mathbb{N}$, $\mathsf{MA}^{\mathcal{O}} \not\subseteq \mathsf{SIZE}^{\mathcal{O}}[n^k]$.*

Proof. We first note that $L_{\mathsf{hard}} \in \mathsf{SIZE}^{\mathcal{O}}[\mathrm{poly}]$ implies $L_{\mathsf{hard}} \in \mathsf{MA}^{\mathcal{O}}$. More specifically, the $\mathsf{MA}^{\mathcal{O}}$ protocol for L_{hard} on input $x \in \{0,1\}^n$ is as follows. Merlin sends to Arthur the polynomial-size $\mathcal{O}$-oracle circuit C_n that computes L_{hard} on inputs of length n. Then Arthur, which has access to the oracle $\mathcal{O}$, runs the (same-length) instance checker I for L_{hard} on x while answering its queries using C_n and outputs 1 if and only if $I^{C_n}(x) = 1$. The correctness follows from the property of the instance checker. Since L_{hard} is PSPACE-complete, we have $\mathsf{PSPACE} \subseteq \mathsf{MA}^{\mathcal{O}}$.

Following the (folklore) diagonalization argument showing that PSPACE does not have fixed-polynomial-size circuits, one can show that for every $k \geq 1$, there is a language $L \in \mathsf{PSPACE}$ that does not have $\mathcal{O}$-oracle circuits of size n^k. This argument works because $\mathcal{O} \in \mathsf{PSPACE}$ and we can simulate any size-(n^k) $\mathcal{O}$-oracle in polynomial space. Then by the previous paragraph, we have $L \in \mathsf{PSPACE} \subseteq \mathsf{MA}^{\mathcal{O}}$. ◀

▶ **Lemma 61.** *Let $\mathcal{O}$ be any oracle. If $L_{\mathsf{hard}} \notin \mathsf{SIZE}^{\mathcal{O}}[\mathrm{poly}]$, then for every $k \in \mathbb{N}$, $\mathsf{MA}^{\mathcal{O}}/_1 \not\subseteq \mathsf{SIZE}^{\mathcal{O}}[n^k]$.*

The proof of Lemma 61 is essentially the same as the original proof in [65]. We present the detail here for completeness.

Proof of Lemma 61. We define the following padded version of L_{hard}, called L_k, as follows.

> $x \in L_k$ iff $x = yz$, where $y \in L_{\mathsf{hard}}$, $|z| > |y|$, $|z| = 2^\ell$ for some integer ℓ and $(|y|+|z|)^{k+1} \leq s(|x|) < (|y|+2|z|)^{k+1}$, where $s(m)$ is the minimum size of an $\mathcal{O}$-oracle circuit that computes L_{hard} on inputs of length m.

We will show that L_k is in $\mathsf{MA}^{\mathcal{O}}/_1$, but does not have $\mathcal{O}$-oracle circuits of size n^k.

For the upper bound, we first specify the sequence of advice bits. We say that input length $n \in \mathbb{N}$ is *good* for L_k if $n = m + 2^\ell$ for non-negative integers m and ℓ, $n > 2m$ and $(m+2^\ell)^{k+1} \leq s(m) \leq (m+2\cdot 2^\ell)^{k+1}$, where $s(m)$ is the minimum size of an $\mathcal{O}$-oracle circuit that computes L_{hard} on inputs of length m. Note that if n is good for L_k, $m = m(n)$ and $\ell = \ell(n)$ are well-defined, since in this case, we can obtain ℓ by looking at the most significant bit of the binary representation of n. For input length n, we let the corresponding advice bit $b_n = 1$ iff n is good.

Consider the following procedure for deciding L_k. On input $x \in \{0,1\}^n$, Arthur first checks if the advice bit b_n is 1. If not, reject immediately, Otherwise, we have $n = m + 2^\ell$ for some $m, \ell \in \mathbb{N}$ and Arthur can obtain $yz := x$, where $|y| = m, |z| = 2^\ell$. Note that in this case, $x \in L_k$ if and only if $y \in L_{\mathsf{hard}}$. Then Merlin sends to Arthur the minimum $\mathcal{O}$-oracle circuit C_m that computes L_{hard} on inputs of length m. Note that the size of this circuit is $s(m) < (m+2\cdot 2^\ell)^{k+1} \leq n^{O(1)}$ since n is good. Then Arthur, which has oracle access to $\mathcal{O}$, runs the (same-length) instance checker I for L_{hard} on y while answering its queries using C_m, and he outputs 1 if and only if $I^{C_m}(y) = 1$. The correctness follows from the property of the instance checker.

For the lower bounds, suppose for the sake of contradiction that $L_k \in \mathsf{SIZE}^{\mathcal{O}}[n^k]$. Let $\{D_n\}_n$ be a circuit family that computes L_k, where each D_n is an $\mathcal{O}$-oracle circuit of size n^k.

For each $m \in \mathbb{N}$, let $s(m)$ be the minimum size of an $\mathcal{O}$-oracle circuit that computes L_{hard} on inputs of length m. By assumption, $L_{\mathsf{hard}} \notin \mathsf{SIZE}^{\mathcal{O}}[\mathrm{poly}]$, so there is an infinite set $\mathcal{I} \subseteq \mathbb{N}$ such that for each $m \in N$,

$$s(m) > (m+1)^{k+1}.$$

Consider the following sequence of circuits $\{C_m\}_{m\in\mathcal{I}}$ that computes L_{hard} on the input lengths in $\mathcal{I}$. For each $m \in \mathcal{I}$, consider the the unique integer ℓ such that

$$(m + 2^\ell)^{k+1} \leq s(m) < (m + 2 \cdot 2^\ell)^{k+1}. \tag{2}$$

Such an ℓ exists since $(m+1)^{k+1} < s(m) \leq 2^m$ for $m \in \mathcal{I}$. Then on input $x \in \{0,1\}^m$, C_m simulates D_{m+2^ℓ} on $x1^{2^\ell}$. Since $m + 2^\ell$ is a good length and D_{m+2^ℓ} computes L_k correctly on inputs of length $m + 2^\ell$, $C_{m+2^\ell}\left(x1^{2^\ell}\right) = 1$ if and only if $x \in L_k$. Note that the size of C_m is at most the size of D_{m+2^ℓ}, which is at most $(m + 2^\ell)^{k+1} < s(m)$. This contradicts Equation (2). ◀

Proof of Theorem 57. Theorem 57 follows directly from Lemma 60 and Lemma 61. ◀

6.2 Bounded-Relativization Barriers for Circuit Lower Bounds

We present an EXPH-relativization barrier for proving an almost-everywhere version of Santhanam's lower bound. The oracle construction is based on [13]. Although the construction in [13] does not seem to be in EXPH, by modifying the construction using approximation counting in PH, we obtain an EXPH-computable oracle under which $\mathsf{MA}_{/1}$ is computable by linear-sized circuits infinitely often.

▶ **Lemma 62.** *There exists an oracle $\mathcal{O} \in \mathsf{EXPH}$ such that*

$$\mathsf{MATIME}^{\mathcal{O}}[2^n]/_1 \subseteq \mathsf{i.o.SIZE}^{\mathcal{O}}[O(n)].$$

Proof. We prove $\text{pr-}\mathsf{MATIME}^{\mathcal{O}}[2^n] \subseteq \mathsf{i.o.SIZE}^{\mathcal{O}}[O(n)]$ for some oracle $\mathcal{O}$. We enumerate all the 2^n-time randomized machine $M_1, M_2, \cdots$, where each M_i takes an input x of length n and a certificate $y \in \{0,1\}^{2^n}$ and runs in time 2^n. Note that any MA-type algorithm that runs in time 2^n can be simulated by M_i for some i.

Let c be some universal constant ($c := 5$ suffices). Let $R_n := \{0,1\}^{cn}$. The input to $\mathcal{O}$ is of the form (r, i, x), where $r \in R_n$, $i \in [n]$, and $x \in \{0,1\}^n$ for some $n \in \mathbb{N}$. For any $r \in \{0,1\}^{cn}$, let S_r be the set of (r, i, x) such that $i \in [n]$ and $x \in \{0,1\}^n$.

We will construct an oracle $\mathcal{O}$ and a family of "advice strings" $r_n^* \in R_n$ for infinitely many n such that

1. if $\Pr_{M_i}[M_i^{\mathcal{O}}(x, y) = 1] \geq \frac{3}{4}$ for some y, then $(r_n^*, i, x) \in \mathcal{O}$, and
2. if $\Pr_{M_i}[M_i^{\mathcal{O}}(x, y) = 1] \leq \frac{1}{4}$ for any y, then $(r_n^*, i, x) \notin \mathcal{O}$.

Assuming this, it is easy to construct an $\mathcal{O}$-oracle linear-size circuit that simulates M_i as follows. The circuit takes r_n^* as hard-wired input and accepts an input $x \in \{0,1\}^n$ if and only if $(r_n, i, x) \in \mathcal{O}$.

Here is the construction of $\mathcal{O}$. We start with $\mathcal{O} := \emptyset$. Some pairs (i, x) will be marked "forced", meaning that M_i accepts on input x. Some strings r will be marked "frozen", meaning that the behavior of $\mathcal{O}$ on inputs in S_r will not be changed after r is frozen. Initially, no pair is forced and no advice string is frozen. We start with Stage $n := 1$. In Stage n, we construct an oracle as follows.

Stage n. Consider the following condition, which we call $(*)_\theta$ for a threshold $\theta \in (0, 1)$.

> There exist an unfrozen string $r \in R_n$, an unforced pair $(i, x) \in [n] \times \{0,1\}^n$, an oracle $B \subseteq S_r$, and a certificate $y \in \{0,1\}^{2^n}$ such that
>
> $$\Pr_{M_i}\left[M_i^{\mathcal{O}\cup B}(x, y) = 1\right] \geq \theta,$$
>
> where the probability is taken over the coin flip of the randomized machine M_i.

We need to argue that the final oracle $\mathcal{O}$ can be computed in EXPH. It may not be possible to check $(*)_{3/4}$ exactly in EXPH, but using approximate counting in PH (Theorem 23), we can check whether a promise variant of $(*)_{3/4}$ is satisfied or not in EXPH. Specifically, by Theorem 23, there exists an algorithm in PH that, given as input n and $\mathcal{O}$ (which can be encoded as a binary string of length $2^{O(n)}$), accepts if $(*)_{3/4}$ holds, and rejects if $(*)_{1/2}$ does not hold. By the standard search-to-decision reduction, we obtain a PH-oracle polynomial-time algorithm S that outputs $r \in R_n$, $(i,x) \in [n] \times \{0,1\}^n$, $B \subseteq S_r$, and $y \in \{0,1\}^{2^n}$ that satisfy $(*)_{1/2}$ if $(*)_{3/4}$ holds, and outputs $\perp$ if $(*)_{1/2}$ does not hold. (Note that S runs in time $2^{O(n)}$ on inputs of length $2^{O(n)}$ with a PH oracle.)

While the search algorithm S outputs a certificate (r,i,x,B) for $(*)_{1/2}$ on input $(n,\mathcal{O})$ (instead of $\perp$), we do the following. Update $\mathcal{O} := \mathcal{O} \cup B$. Let r be frozen and let (i,x) be forced. Let $\mathcal{R}_\theta$ be the set of $r' \in R_n$ such that (r',i',x') is queried during the computation of $M_i^{\mathcal{O}}(x,y)$ for some (i',x') with probability at least θ over the internal randomness of M_i. By Theorem 24, some set A such that $\mathcal{R}_{2^{-2n}} \subseteq A \subseteq \{0,1\}^* \setminus \mathcal{R}_{2^{-2n-1}}$ can be enumerated in EXPH. Note that $|A| \leq 2^{2n+1} \cdot 2^n$ because M_i can make at most 2^n queries on each computation path. Let r' be frozen for every $r' \in A$. If S outputs $\perp$ (and thus $(*)_{3/4}$ does not hold), then let $r_n^* \in R_n$ be the first string that is not frozen. (We will later claim that such a string r_n^* exists.) Then, we add to $\mathcal{O}$ all the tuples (r_n^*,i,x) such that $(i,x) \in [n] \times \{0,1\}^n$ is forced. This completes the description of Stage n. Then we move on to the next stage $n' := 2^n + 1$, so that the construction in each stage is independent.

It is evident from the construction that $\mathcal{O} \in \mathsf{EXPH}$.

Fix any n and consider Stage n. We claim that there exists a string $r_n^* \in R_n$ that is not frozen at the end of Stage n. We say that $(*)$ is satisfied if S does not output $\perp$. Observe that the number of times that $(*)$ is satisfied is at most $n2^n$. The reason is that there are at most $n2^n$ pairs $(i,x) \in [n] \times \{0,1\}^n$, and each time $(*)$ is satisfied, (i,x) becomes forced. Each time $(*)$ is satisfied, at most $1 + 2^{2n+1} \cdot 2^n \leq 2^{3n+2}$ strings $r \in R_n$ can be frozen. Thus, the number of frozen strings in R_n in Stage n is at most $n2^n \cdot 2^{3n+2}$. Since there are at most n stages before Stage n, in total, there are at most $n \cdot n2^n \cdot 2^{3n+2} < 2^{5n} \leq |R_n|$ frozen strings in R_n. (Here, we used that $c := 5$.) Thus, there exists some unfrozen string $r_n^* \in R_n$, which shows that the construction of $\mathcal{O}$ is well defined.

It remains to show that for every pair $(i,x) \in [n] \times \{0,1\}^n$,

1. if $\Pr_{M_i}[M_i^{\mathcal{O}}(x,y) = 1] \geq \frac{3}{4}$ for some y, then (i,x) is forced (and thus $(r_n,i,x) \in \mathcal{O}$), and
2. if $\Pr_{M_i}[M_i^{\mathcal{O}}(x,y) = 1] \leq \frac{1}{4}$ for any y, then (i,x) is not forced (and thus $(r_n,i,x) \notin \mathcal{O}$).

Fix any n and consider any unforced pair $(i,x) \in [n] \times \{0,1\}^n$. We claim that for every $y \in \{0,1\}^{2^n}$, it holds that

$$\Pr_{M_i}\left[M_i^{\mathcal{O}}(x,y) = 1\right] < \frac{3}{4}.$$

Fix any $y \in \{0,1\}^{2^n}$. Let $\mathcal{O}_n$ be the oracle $\mathcal{O}$ right after the while loop of Stage n. Since $(*)_{3/4}$ is not satisfied at that point, for every $r \in R_n$ and for every oracle $B \subseteq S_r$,

$$\Pr_{M_i}\left[M_i^{\mathcal{O}_n \cup B}(x,y) = 1\right] < \frac{3}{4}.$$

Let $B \subseteq S_{r_n^*}$ be the set of strings added to $\mathcal{O}$ at the end of Stage n. Then, the final oracle $\mathcal{O}$ coincides with $\mathcal{O}_n \cup B$ on any inputs of length at most 2^n; thus, we obtain

$$\Pr_{M_i}\left[M_i^{\mathcal{O}}(x,y) = 1\right] = \Pr_{M_i}\left[M_i^{\mathcal{O}_n \cup B}(x,y) = 1\right] < \frac{3}{4}.$$

Next, fix any n and consider any forced pair $(i,x) \in [n] \times \{0,1\}^n$. Let $\mathcal{O}_{i,x}$ be the oracle right after (i,x) is forced and y be the certificate that satisfies $(*)$. Then, we have

$$\Pr\left[M_i^{\mathcal{O}_{i,x}}(x,y)=1\right] \geq \frac{1}{2}.$$

We claim that this probability does not decrease by $\frac{1}{4}$ even if we replace $\mathcal{O}_{i,x}$ with the final oracle $\mathcal{O}$. After (i,x) is forced, $(*)$ can be satisfied at most $n2^n$ times, and each time $(*)$ is satisfied, at most one S_r whose subset is added to $\mathcal{O}$. Thus, there are at most $n2^n$ unfrozen strings $r \in R_n$ such that some strings in S_r are added to $\mathcal{O}$. Note that any string r that is queried by M_i with probability 2^{-2n} is frozen right after (i,x) is forced. Thus, each unfrozen string can decrease the probability that $M_i^{\mathcal{O}}(x,y)$ accepts by 2^{-2n}. In total, the unfrozen strings can affect the probability of acceptance by $n2^n \cdot 2^{-2n} < \frac{1}{4}$. Thus, we obtain

$$\Pr\left[M_i^{\mathcal{O}}(x,y)=1\right] > \frac{1}{2} - \frac{1}{4} \geq \frac{1}{4}.$$ ◀

▶ **Corollary 63.** *If there is a* PSPACE*-relativizing proof that for every constant* $k \geq 1$*,* $\mathsf{MA}/_1 \not\subseteq \mathsf{i.o.SIZE}[n^k]$*, then* $\mathsf{L} \neq \mathsf{NP}$ *follows.*

7 Open Problems

We believe that the perspective of bounded relativization will lead to a better understanding of current proof techniques, in particular interactive proofs. There are many interesting questions left open:

1. Find some reason that "current techniques" have not been able to separate PSPACE from EXPH. We interpret the results in this paper as: If "current techniques" are PSPACE-relativizing and cannot separate PSPACE from EXPH, then they also cannot prove "slight improvements of known results" for which an EXPH-relativization barrier exists.
   The "conventional wisdom" seems to indicate that it is difficult for "current techniques" to separate PSPACE from EXPH, but we have not been able to find any formal justification. Instead, an optimist might treat the oracles in this paper as first steps for attacking the PSPACE vs. EXPH problem – IP = PSPACE indicates some *weakness* of PSPACE, thus making it vulnerable to separate from EXPH.
2. Find new oracles in EXPH that rules out "slight" improvements of known results. For example, can we show that EXPH-relativizing techniques cannot prove $\mathsf{MA} \not\subseteq \mathsf{SIZE}[n^k]$? (Santhanam's lower bound [65] is for the class $\mathsf{MA}/_1$, and it has been an open problem since then to eliminate the one-bit advice in the lower bound.) Can we show that EXPH-relativizing techniques cannot yield fast derandomization from strong uniform lower bounds for EXP, such as $\mathsf{EXP} \not\subseteq \mathsf{BPSUBEXP} \implies \mathsf{BPP} \subseteq \mathsf{i.o.heur\text{-}QuasiP}$?[14]
3. What can bounded relativization say about the Algorithmic Method for proving circuit lower bounds? Williams [78, 79] showed that non-trivial circuit-analysis (e.g., satisfiability or derandomization) algorithms for polynomial-size circuits imply that $\mathsf{NEXP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$. The general implication from non-trivial algorithms to circuit lower bounds does not relativize as shown in [75, Theorem 11]; however, it is unclear whether the corresponding oracle is in EXPH.

[14] Note that we might need to assume $\mathsf{P} \neq \mathsf{L}$ for this problem, since if $\mathsf{P} = \mathsf{L}$ then $\mathsf{EXP}^{\mathcal{O}[\mathrm{poly}]} = \mathsf{PSPACE}^{\mathcal{O}}$ for every oracle $\mathcal{O}$. It is known that strong uniform lower bound for PSPACE implies fast derandomization of BPP [70, 20], and the proofs are likely PSPACE-relativizing.

4. Finally, is there an oracle world under which Σ_2^p has linear-size circuits on infinitely many input lengths? Our construction showed that if we replace Σ_2^p by pr-MA then such an oracle world exists and can be computed in EXPH. This problem is connected to the Missing-String problem studied in [75].

References

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

2 Scott Aaronson. The teaser. https://scottaaronson.blog/?p=3054, 2017. Accessed: Feb 6, 2023.

3 Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1(1):2:1–2:54, 2009. doi:10.1145/1490270.1490272.

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

5 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.

6 Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

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

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

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

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

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

12 Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 195–209, 2009. doi:10.1007/978-3-642-02927-1_18.

13 Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Conference on Computational Complexity* (CCC), pages 8–12, 1998. doi:10.1109/CCC.1998.694585.

14 Harry Buhrman and Leen Torenvliet. Randomness is hard. *SIAM J. Comput.*, 30(5):1485–1501, 2000. doi:10.1137/S0097539799360148.

15 Samuel R. Buss and R. Ryan Williams. Limits on alternation trading proofs for time-space lower bounds. *Comput. Complex.*, 24(3):533–600, 2015. doi:10.1007/s00037-015-0104-9.

16 Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *J. Comput. Syst. Sci.*, 49(1):24–39, 1994. doi:10.1016/S0022-0000(05)80084-4.

17 Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *Symposium on Foundations of Computer Science* (FOCS), pages 1–12. IEEE, 2020. doi:10.1109/FOCS46700.2020.00009.

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

19 Lijie Chen, Ron D. Rothblum, and Roei Tell. Unstructured hardness to average-case randomness. In *Symposium on Foundations of Computer Science* (FOCS), pages 429–437. IEEE, 2022. doi:10.1109/FOCS54457.2022.00048.

20 Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. On exponential-time hypotheses, derandomization, and circuit lower bounds. In *Symposium on Foundations of Computer Science* (FOCS), pages 13–23. IEEE, 2020. doi:10.1109/FOCS46700.2020.00010.

21 Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *Symposium on Foundations of Computer Science* (FOCS), pages 125–136. IEEE, 2021. doi:10.1109/FOCS52979.2021.00021.

22 Scott Diehl and Dieter van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM J. Comput.*, 36(3):563–594, 2006. doi:10.1137/050642228.

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

24 Lance Fortnow. Time-space tradeoffs for satisfiability. *J. Comput. Syst. Sci.*, 60(2):337–353, 2000. doi:10.1006/jcss.1999.1671.

25 Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, 2005. doi:10.1145/1101821.1101822.

26 Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Symposium on Foundations of Computer Science* (FOCS), pages 316–324, 2004. doi:10.1109/FOCS.2004.33.

27 Lance Fortnow, Rahul Santhanam, and R. Ryan Williams. Fixed-polynomial size circuit bounds. In *Computational Complexity Conference* (CCC), pages 19–26. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.21.

28 Lance Fortnow and Michael Sipser. Are there interactive protocols for coNP languages? *Inf. Process. Lett.*, 28(5):249–251, 1988. doi:10.1016/0020-0190(88)90199-8.

29 Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Inf. Process. Lett.*, 95(2):354–357, 2005. doi:10.1016/j.ipl.2005.03.009.

30 Oded Goldreich. *Computational complexity – A conceptual perspective*. Cambridge University Press, 2008.

31 Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. On the pseudo-deterministic query complexity of NP search problems. In *Computational Complexity Conference* (CCC), volume 200 of *LIPIcs*, pages 36:1–36:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.36.

32 Hans Heller. On relativized exponential and probabilistic complexity classes. *Inf. Control.*, 71(3):231–243, 1986. doi:10.1016/S0019-9958(86)80012-2.

33 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018. doi:10.1109/FOCS.2018.00032.

34 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Computational Complexity Conference* (CCC), volume 169 of *LIPIcs*, pages 20:1–20:47. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.20.

35 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Symposium on Theory of Computing* (STOC), pages 1038–1051, 2020. doi:10.1145/3357713.3384251.

36 Shuichi Hirahara. Symmetry of information from meta-complexity. In *Computational Complexity Conference* (CCC), volume 234 of *LIPIcs*, pages 26:1–26:41. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.26.

37 Shuichi Hirahara, Zhenjian Lu, and Hanlin Ren. Bounded relativization. *Electron. Colloquium Comput. Complex.*, 30:70, 2023. URL: https://eccc.weizmann.ac.il/report/2023/070.

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

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

40 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002. doi:10.1016/S0022-0000(02)00024-7.

41 Russell Impagliazzo and Avi Wigderson. $\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing* (STOC), pages 220–229. ACM, 1997. doi:10.1145/258533.258590.

42 Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.

43 Emil Jerábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Log.*, 129(1-3):1–37, 2004. doi:10.1016/j.apal.2003.12.003.

44 Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. $\mathsf{MIP}^* = \mathsf{RE}$. *CoRR*, abs/2001.04383, 2020. doi:10.48550/arXiv.2001.04383.

45 Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *J. Comput. Syst. Sci.*, 63(2):236–252, 2001. doi:10.1006/jcss.2001.1763.

46 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79. ACM, 2000. doi:10.1145/335305.335314.

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

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

49 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *Innovations in Theoretical Computer Science* (ITCS), volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.44.

50 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

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

52 Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1.

53 Richard J. Lipton and Anastasios Viglas. On the complexity of SAT. In *Symposium on Foundations of Computer Science* (FOCS), pages 459–464. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814618.

54 Zhenjian Lu, Igor Carboni Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Symposium on Theory of Computing* (STOC), pages 303–316. ACM, 2021. doi:10.1145/3406325.3451085.

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

56 Oleg B Lupanov. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR*, 119(1):23–26, 1958.

57 Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *International Computing and Combinatorics Conference* (COCOON), pages 210–220, 1999. doi:10.1007/3-540-48686-0_21.

58 Cody D. Murray and R. Ryan Williams. Easiness amplification and uniform circuit lower bounds. In *Computational Complexity Conference* (CCC), volume 79 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.8.

59 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

60 Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.32.

61 Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18.

62 Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017. doi:10.1145/3055399.3055500.

63 Hanlin Ren and Rahul Santhanam. A relativization perspective on meta-complexity. In *International Symposium on Theoretical Aspects of Computer Science* (STACS), volume 219 of *LIPIcs*, pages 54:1–54:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.54.

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

65 Rahul Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009. doi:10.1137/070702680.

66 Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992. doi:10.1145/146585.146609.

67 Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Syst. Tech. J.*, 28(1):59–98, 1949. doi:10.1002/j.1538-7305.1949.tb03624.x.

68 Michael Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory of Computing* (STOC), pages 330–335, 1983. doi:10.1145/800061.808762.

69 Larry J. Stockmeyer. The complexity of approximate counting (preliminary version). In *Symposium on Theory of Computing* (STOC), pages 118–126. ACM, 1983. doi:10.1145/800061.808740.

70 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

71 Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. doi:10.1016/S0022-0000(03)00046-1.

72 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

73 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.

74 Emanuele Viola. On approximate majority and probabilistic time. In *Conference on Computational Complexity* (CCC), pages 155–168. IEEE Computer Society, 2007. doi:10.1109/CCC.2007.16.

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

76 R. Ryan Williams. Inductive time-space lower bounds for SAT and related problems. *Comput. Complex.*, 15(4):433–470, 2006. doi:10.1007/s00037-007-0221-1.

77 R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Comput. Complex.*, 17(2):179–219, 2008. doi:10.1007/s00037-008-0248-y.

78 R. Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. doi:10.1137/10080703X.

79 R. Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.

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