Electronic Colloquium on Computational Complexity, Report No. 115 (2024)

On the Complexity of Avoiding Heavy Elements

Zhenjian Lu[*]
Department of Computer Science
University of Warwick

Igor C. Oliveira[†]
Department of Computer Science
University of Warwick

Hanlin Ren[‡]
Department of Computer Science
University of Oxford

Rahul Santhanam[§]
Department of Computer Science
University of Oxford

July 13, 2024

Abstract

We introduce and study the following natural total search problem, which we call the *heavy element avoidance* (Heavy Avoid) problem: for a distribution on N bits specified by a Boolean circuit sampling it, and for some parameter $\delta(N) \geq 1/\mathsf{poly}(N)$ fixed in advance, output an N-bit string that has probability less than $\delta(N)$. We show that the complexity of Heavy Avoid is closely tied to frontier open questions in complexity theory about uniform randomized lower bounds and derandomization. Among other results, we show:

1. For a wide range of circuit classes $\mathcal{C}$, including $\mathsf{ACC}^0, \mathsf{TC}^0$, NC^1 and general Boolean circuits, EXP does not have uniform randomized $\mathcal{C}$-circuits if and only if Heavy Avoid for uniform implicit $\mathcal{C}$-samplers has efficient deterministic algorithms infinitely often. This gives the first *algorithmic characterization* of lower bounds for EXP against uniform randomized low-depth circuits. We show similar algorithmic characterizations for lower bounds in PSPACE, NP and $\mathsf{EXP}^{\mathsf{NP}}$.
2. Unconditionally, there are polynomial-time *pseudodeterministic* algorithms that work infinitely often for several variants of Heavy Avoid, such as for uniform samplers of small randomness complexity. In contrast, the existence of a similar algorithm that solves Heavy Avoid for arbitrary polynomial-time samplers would solve a long-standing problem about hierarchies for probabilistic time.
3. If there is a time and depth efficient deterministic algorithm for Heavy Avoid, then $\mathsf{BPP} = \mathsf{P}$. Without the depth-efficiency requirement in the assumption, we still obtain a non-trivial form of infinitely-often deterministic simulation of randomized algorithms. These results are shown using *non-black-box* reductions, and we argue that the use of non-black-box reductions is essential here.

[*]Email: zhenjian.lu@warwick.ac.uk
[†]Email: igor.oliveira@warwick.ac.uk
[‡]Email: hanlin.ren@cs.ox.ac.uk
[§]Email: rahul.santhanam@cs.ox.ac.uk

ISSN 1433-8092

Contents

1 Introduction

Let C be a Boolean circuit sampling a distribution D on N-bit strings. Say that an N-bit string y is δ-heavy in D if y occurs with probability at least δ in D. Assuming that some 2δ-heavy string exists, for $\delta \geq 1/\mathsf{poly}(N)$, how hard is it to find a δ-heavy string given C as input?

We call this natural search problem the *heavy element finding* (Heavy Find) problem. It is not difficult to see that the complexity of Heavy Find is closely related to the complexity of derandomization. There is a simple randomized polynomial-time algorithm for Heavy Find: we use C to draw $O(N/\delta^2)$ independent samples from D and output the string that occurs with the greatest multiplicity in the multiset of samples. A standard application of Chernoff–Hoeffding bounds shows that assuming that a 2δ-heavy string exists, the output of the algorithm will be a string that is δ-heavy in D with high probability.

Moreover, a deterministic polynomial-time algorithm for Heavy Find implies $\mathsf{BPP} = \mathsf{P}$. Indeed, let M be a probabilistic polynomial-time Turing machine with error bounded by $1/4$ and x be an input to M. We can define a circuit sampler C_x which interprets its input as randomness r for the computation of M on x, outputting 1^N if M accepts on x using randomness r and 0^N otherwise. Observe that if M accepts x, the unique solution to Heavy Find on input C_x with parameter $\delta = 1/3$ is 1^N, and if M rejects x, the unique solution to Heavy Find on input C_x with parameter $1/3$ is 0^N. Thus, a deterministic polynomial-time algorithm for Heavy Find allows us to decide if M accepts x, also in deterministic polynomial time.[1]

We now turn our original question on its head: given C as input, how hard is it to find a string that is *not* δ-heavy? We call this the *heavy element avoidance* (Heavy Avoid) problem. Heavy Avoid is the complementary search problem to Heavy Find: a string $y \in \{0,1\}^N$ is a solution to Heavy Avoid if and only if it is not a solution to Heavy Find. The complexity of Heavy Avoid is the primary focus of this paper.

Superficially, Heavy Avoid seems to be a much simpler problem to solve than Heavy Find. First, when $\delta > 2^{-N}$, Heavy Avoid is a *total* search problem, i.e., the promise that a non-heavy N-bit string exists is automatically satisfied. In this paper, we mainly focus on the regime where $\delta \geq 1/\mathsf{poly}(N)$, hence this is always true if N is large enough. Second, there is a trivial algorithm that *list*-solves Heavy Avoid: Since the number of δ-heavy strings is at most $1/\delta$, at least one of the lexicographically first $\lceil 1/\delta \rceil + 1$ strings of length N is guaranteed to be a solution to Heavy Avoid. Third, there is a very efficient randomized algorithm for Heavy Avoid with overwhelming success probability: output a uniformly random string of length N. Note that by the previous observation that the number of δ-heavy strings is at most $1/\delta$, this randomized algorithm fails on at most $1/\delta \leq \mathsf{poly}(N)$ of its random choices.

Our main contribution is to introduce Heavy Avoid as a natural search problem of interest, and show that despite its seeming simplicity, Heavy Avoid has applications to several frontier questions in complexity theory regarding *uniform randomized lower bounds* and *derandomization*. Indeed, we show that in many settings the existence of algorithms for Heavy Avoid is *equivalent* to a complexity lower bound. The study of Heavy Avoid also illuminates recent *almost-all-inputs-hardness* assumptions in the theory of derandomization [CT21], and leads to novel *white-box* reductions in settings where black-box reductions are hard to show. Moreover, the connections between Heavy Avoid and complexity lower bounds can be used to derive efficient unconditional *pseudodeterministic* algorithms (in the sense of [GG11]) for Heavy Avoid in several settings, i.e., efficient randomized algorithms that output some *fixed* solution to Heavy Avoid with high probability.

[1]Readers who are familiar with derandomization might already see that the derandomization also holds for the *promise* version of BPP (prBPP). In fact, it is not hard to show that Heavy Find can be solved in deterministic polynomial time if and only if prBPP collapses to prP, the promise version of P.

1.1 Results

We now discuss our results in greater detail. We present algorithmic characterizations of uniform lower bounds via Heavy Avoid, unconditional pseudodeterministic algorithms for Heavy Avoid, and connections between derandomization with minimum assumptions and Heavy Avoid. Thus, our results can be divided naturally into three sets.

- In Section 1.1.1, we present algorithmic characterizations of lower bounds against uniform probabilistic circuits via Heavy Avoid. That is, deterministic algorithms for Heavy Avoid (in certain settings and with certain parameters) are *equivalent* to such lower bounds. In fact, we obtain very general characterizations that hold for classes such as $\mathsf{EXP}, \mathsf{PSPACE}, \mathsf{EXP}^{\mathsf{NP}}$ and NP, against uniform randomized circuit classes such as ACC^0, TC^0, or $\mathsf{SIZE}[\mathsf{poly}]$. This suggests that the analysis of Heavy Avoid could be useful in attacking frontier open questions such as $\mathsf{EXP} \not\subseteq \mathsf{BP}\text{-}\mathsf{ACC}^0$ and $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{BPP}$.

- In Section 1.1.2, we use our algorithmic characterizations together with other ideas to give *unconditional* pseudodeterministic algorithms for several variants of Heavy Avoid.

- In Section 1.1.3, we give applications of Heavy Avoid to derandomization, including novel *white-box* reductions from promise problems that are hard for prRP or prBPP to Heavy Avoid, as well as connections to "almost-all-inputs-hardness" assumptions that have been explored in recent work on derandomization.

We consider both *uniform* and *non-uniform* versions of Heavy Avoid. In the uniform version, the search algorithm is given N in unary, and needs to find a δ-light[2] element in D_N, where $\mathcal{D} = \{D_N\}_{N\in\mathbb{N}}$ is an ensemble of distributions over N-bit strings that are sampled by some *uniform* sequence of circuits from a circuit class. Since $\mathcal{D}$ is sampled by a uniform sequence of circuits, we do not need to give the circuit sampler explicitly to the search algorithm—the search algorithm can compute the circuit sampler by itself. In this uniform variant of the problem, fix a parameter $\delta \colon \mathbb{N} \to [0,1]$, $(\mathcal{D}, \delta)$-$\texttt{Heavy-Avoid}$ is the problem of finding a $\delta(N)$-light element in D_N, given 1^N as input. This variant is the one we consider in Sections 1.1.1 and 1.1.2.

In the non-uniform variant of the problem, the search algorithm is given as input a circuit sampler C from some circuit class $\mathcal{C}$, and needs to output a δ-light element in the distribution sampled by C. This is the version we mostly consider for the results in Section 1.1.3.

There are also two kinds of samplability we consider: *implicit* and *explicit*. In the implicit version, our sampler C is Boolean: given randomness r as input together with an index $i \in [N]$, it outputs the i'th bit of the string sampled on randomness r. In this setting, the circuit size is typically less than N. In the explicit version, the circuit C is given randomness r as input and has N output bits: it outputs the string sampled on randomness r. In this setting, the circuit size is at least N, since there are N output bits. Note that when we show an implication from solving Heavy Avoid to proving lower bounds, the implication is *stronger* when we consider implicit solvers, since the algorithmic problem is *easier* to solve for implicit samplers.[3] An implicit solver $C(r, i)$ can easily be converted to an equivalent explicit solver

$$C_{\mathsf{Explicit}}(r) := C(r,1)C(r,2)\ldots C(r,N).$$

[2]A δ-light element is one that is not δ-heavy.

[3]We measure the complexity of solving the search problem as a function of N, even in the implicit-sampler setting.

1.1.1 Equivalences Between Complexity Separations and Algorithms for Heavy Avoid

It is a long-standing open question to prove lower bounds against non-uniform circuits – we still have not ruled out the possibility that every language computable in exponential time with an NP oracle ($\mathsf{EXP}^{\mathsf{NP}}$) has polynomial-size circuits. What is more embarrassing is our inability to separate $\mathsf{EXP}^{\mathsf{NP}}$ from BPP (see, e.g., [Wil13b, Wil19] for discussions), despite the belief shared by many researchers that $\mathsf{BPP} = \mathsf{P}$ [NW94, IW97].[4] Moreover, the state of affairs is the same regarding lower bounds against uniform probabilistic circuits from *restricted* circuit classes: for example, it is open whether EXP can be simulated by $\mathsf{DLOGTIME}$-uniform probabilistic ACC^0 circuits or $\mathsf{EXP}^{\mathsf{NP}}$ can be simulated by $\mathsf{DLOGTIME}$-uniform probabilistic TC^0 circuits.[5]

Our first set of results gives *equivalences* between such explicit lower bounds against uniform probabilistic circuits and efficient deterministic algorithms for Heavy Avoid. The equivalences work in a wide variety of settings, for a range of circuit classes including $\mathsf{ACC}^0, \mathsf{TC}^0, \mathsf{NC}^1$ and general Boolean circuits, and for explicit lower bounds in several standard complexity classes of interest such as $\mathsf{EXP}, \mathsf{EXP}^{\mathsf{NP}}, \mathsf{PSPACE}$ and NP. Notably, these results give new *algorithmic characterizations* of uniform lower bound questions by the existence of efficient algorithms for a natural search problem. Thus they could potentially be useful in attacking frontier open questions such as the EXP vs (uniform probabilistic) ACC^0 question, or the $\mathsf{EXP}^{\mathsf{NP}}$ vs BPP question.

We use $\mathsf{BP}\text{-}\mathcal{C}$ to denote the set of languages computed by $\mathsf{DLOGTIME}$-uniform probabilistic $\mathcal{C}$-circuits.

Theorem 1.1 (Informal). *Let $\mathcal{C}$ be a nice*[6] *class of Boolean circuits. The following equivalences hold:*

(i) $\mathsf{EXP} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ *if and only if* $(\mathcal{D}, \delta)$-`Heavy-Avoid` *with* $\delta(N) = 1/\mathsf{polylog}(N)$ *can be solved in deterministic polynomial time on infinitely many input lengths for any* $\mathcal{D}$ *that admits implicit* $\mathsf{DLOGTIME}$*-uniform* $\mathcal{C}$*-samplers of size* $\mathsf{polylog}(N)$.

(ii) $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ *if and only if* $(\mathcal{D}, \delta)$-`Heavy-Avoid` *with* $\delta(N) = 1/\mathsf{polylog}(N)$ *can be solved in deterministic polynomial time with an* NP *oracle on infinitely many input lengths for any* $\mathcal{D}$ *that admits implicit* $\mathsf{DLOGTIME}$*-uniform* $\mathcal{C}$*-samplers of size* $\mathsf{polylog}(N)$.

(iii) $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ *if and only if* $(\mathcal{D}, \delta)$-`Heavy-Avoid` *with* $\delta(N) = 1/\mathsf{polylog}(N)$ *can be solved in deterministic logarithmic space on infinitely many input lengths for any* $\mathcal{D}$ *that admits implicit* $\mathsf{DLOGTIME}$*-uniform* $\mathcal{C}$*-samplers of size* $\mathsf{polylog}(N)$.

(iv) $\mathsf{NP} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ *if and only if* $(\mathcal{D}, \delta)$-`Heavy-Avoid` *with* $\delta(N) = 1/\mathsf{polylog}(N)$ *can be solved by* $\mathsf{DLOGTIME}$*-uniform unbounded fan-in circuits of quasi-polynomial size and constant depth on infinitely many input lengths for any* $\mathcal{D}$ *that admits implicit* $\mathsf{DLOGTIME}$*-uniform* $\mathcal{C}$*-samplers of size* $\mathsf{polylog}(N)$.

For the PSPACE lower bounds, analogous algorithmic characterizations hold for *almost everywhere* uniform lower bounds and for lower bounds against uniform randomized *sub-exponential* size

[4]Since BPP is strictly contained in $\mathsf{SIZE}[\mathsf{poly}]$ [Adl78], the open problem of separating $\mathsf{EXP}^{\mathsf{NP}}$ from BPP is *strictly more embarrassing* than separating $\mathsf{EXP}^{\mathsf{NP}}$ from $\mathsf{SIZE}[\mathsf{poly}]$! See also [Wil19, Table 1] for a related perspective.

[5]It follows from $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{ACC}^0$ [Wil14, CLW20], which is a *non-uniform* circuit lower bound, that $\mathsf{EXP}^{\mathsf{NP}}$ cannot be simulated by $\mathsf{DLOGTIME}$-uniform probabilistic ACC^0 circuits. (Note that we do not know how to prove such lower bounds by exploiting the circuit *uniformity* condition.)

[6]In brief, a nice circuit class is one that contains $\mathsf{AC}^0[\oplus]$, is closed under composition, and admits universal circuits for the corresponding class.

circuits. Perhaps interestingly, it follows from our arguments that the existence of efficient algorithms for $(\mathcal{D}, \delta)$-`Heavy-Avoid` in the settings considered in Theorem 1.1 is *robust* with respect to the threshold parameter $\delta(N)$: the existence of algorithms for any $\delta(N) = o(1)$ yields the existence of algorithms of similar complexity for $\delta(N) = 1/\mathsf{polylog}(N)$.

Theorem 1.1 has direct corollaries that characterize frontier open questions in complexity theory.

Corollary 1.2 (Informal)**.** *The following results hold:*

(*i*) $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{BP}\text{-}\mathsf{TC}^0$ *if and only if* Heavy-Avoid *for implicit* DLOGTIME*-uniform* TC^0*-samplers can be solved in deterministic polynomial time with access to and* NP *oracle on infinitely many input lengths.*

(*ii*) $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathsf{ACC}^0$ *if and only if* Heavy-Avoid *for implicit* DLOGTIME*-uniform* ACC^0*-samplers can be solved in logarithmic space on infinitely many input lengths.*

Previously, algorithmic characterizations of *non-uniform* lower bounds were known for classes such as NEXP [IKW02, Wil16] and $\mathsf{EXP}^{\mathsf{NP}}$ [Kor21, RSW22], and such characterizations for uniform randomized lower bounds against *general* circuits (that is, against BPP) were known for EXP [IW01] and NEXP [Wil16]. We are not aware of any previous algorithmic characterization of super-polynomial non-uniform or uniform randomized lower bounds for NP.

As a consequence of our results, we also observe a sharp threshold phenomenon in the setting of *quantified derandomization* of search problems (see Section 1.3 below for related work on quantified derandomization and some discussions, and Section 3.3 for a proof of this corollary).

Corollary 1.3. *For every function $e(N) = \omega(1)$, there is a unary* BPP *search problem S such that:*

- *$S(1^N)$ can be solved in randomized* $\mathsf{poly}(N)$ *time by an algorithm that uses N random bits and errs on at most $e(N)$ random bit sequences;*
- *If there exists a randomized polynomial-time algorithm for S with non-zero success probability that errs on at most $O(1)$ random bit sequences for any input, then* $\mathsf{EXP} \neq \mathsf{BPP}$.

1.1.2 Unconditional Pseudodeterministic Algorithms for Heavy Avoid

Recall that a randomized algorithm is *pseudodeterministic* if it outputs the same answer with high probability. In the next result, we use the results of the previous subsection together with other ideas to obtain unconditional pseudodeterministic algorithms for different variants of Heavy Avoid.

Theorem 1.4. *Let $\mathcal{D} = \{D_N\}_{N \geq 1}$ be a distribution ensemble, where each D_N is supported over $\{0,1\}^N$. The following results hold:*

(*i*) *Let $\mathcal{C}$ be a nice class of Boolean circuits, and suppose $\mathcal{D}$ admits implicit* DLOGTIME*-uniform $\mathcal{C}$-samplers of size* $\mathsf{polylog}(N)$*. Then, for every function $\delta(N) = 1/(\log N)^k$, where $k \in \mathbb{N}$, either $(\mathcal{D}, \delta)$-*`Heavy-Avoid` *can be solved in logarithmic space on infinitely many input lengths, or $(\mathcal{D}, \delta)$-*`Heavy-Avoid` *can be solved pseudodeterministically by* DLOGTIME*-uniform* BP-$\mathcal{C}$*-circuits of size* $\mathsf{polylog}(N)$ *on all input lengths.*[7]

In particular, when $\mathcal{C}$ is the class of general Boolean circuits and $\delta(N) = 1/(\log N)^k$, there is a polynomial-time pseudodeterministic algorithm for the $(\mathcal{D}, \delta)$-`Heavy-Avoid` *problem that succeeds on infinitely many inputs.*

[7]In other words, the corresponding DLOGTIME-uniform BP-$\mathcal{C}$-circuit $E(i)$ is given $i \in [N] = \{0,1\}^{\log N}$ and outputs the i-th bit of the solution with high probability.

(*ii*) *Suppose $\mathcal{D}$ admits a polynomial-time sampler of randomness complexity $(\log N)^{O(1)}$. Then, for every function $\delta(N) = 1/(\log N)^k$, where $k \in \mathbb{N}$, there is a polynomial-time pseudodeterministic algorithm for the $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem that succeeds on infinitely many inputs.*[8]

(*iii*) *Suppose $\mathcal{D}$ admits a polynomial-time sampler. Then, for every function $\delta(N) = 1/N^k$, where $k \in \mathbb{N}$, and for every constant $\varepsilon > 0$, there is a pseudodeterministic algorithm for the $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem that runs in time 2^{N^ε} and succeeds on infinitely many inputs.*

Moreover, in all items the corresponding algorithm behaves pseudodeterministically on every input.

While the pseudodeterministic algorithms above are for natural algorithmic problems about samplers, the design and analysis of the algorithms rely heavily on connections to complexity theory.

The first item of Theorem 1.4 provides a pseudodeterministic polynomial-time infinitely-often algorithm for Heavy Avoid for *implicit samplers*. Moreover, this algorithm computes pseudodeterministically on all input lengths. We observe that the existence of a pseudodeterministic algorithm with these properties for the larger class of *explicit samplers* (as in the third item of Theorem 1.4) would solve the longstanding open problem of showing a tight hierarchy theorem for probabilistic time [KV87, Bar02, FS04, FST05, vMP06]. This is captured by the following result (see Section 4.4 for a more detailed discussion).

Proposition 1.5. *Suppose that for every polynomial-time samplable distribution ensemble $\mathcal{D} = \{D_N\}_{N \geq 1}$, the corresponding $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem for $\delta(N) = 1/\log N$ admits a pseudodeterministic polynomial-time algorithm that succeeds on infinitely many input lengths and behaves pseudodeterministically on all input lengths. Then, for every constant $k \geq 1$, we have $\mathsf{BPE} \not\subseteq \mathsf{BPTIME}[2^{k \cdot n}]$ (in particular, for every constant $c \geq 1$, $\mathsf{BPP} \not\subseteq \mathsf{BPTIME}[n^c]$).*

1.1.3 Connections to Derandomization

Our final set of results explore relations between the complexity of Heavy Avoid and fundamental questions in derandomization. We consider the *non-uniform* variant of Heavy Avoid, where a Boolean circuit sampler is given as input to the algorithm solving Heavy Avoid. For $\delta\colon \mathbb{N} \to [0, 1]$, `Implicit-`$\delta$`-Heavy-Avoid` is the problem where we are given as input a circuit C implicitly sampling a distribution on N bits (as explained at the beginning of Section 1.1), and would like to output a δ-light element in the distribution.

Our first result shows that the existence of efficient deterministic algorithms for Heavy Avoid that in addition can be implemented by sub-polynomial depth uniform circuits leads to a complete derandomization of prBPP. Note that in this result, to obtain the desired conclusion it is sufficient for this algorithm to solve the problem for implicit samplers.

Theorem 1.6. *Let $\delta(N) = o(1)$ be any function. Suppose there is a constant $\epsilon > 0$ and a deterministic algorithm $\mathcal{A}$ that solves the `Implicit-`δ`-Heavy-Avoid` problem on implicit samplers of size N^ϵ. Moreover, assume that $\mathcal{A}$ can be implemented as a logspace-uniform circuit of size $\mathsf{poly}(N)$ and depth $N^{o(1)}$. Then $\mathsf{prBPP} = \mathsf{prP}$.*

If we could eliminate the circuit depth constraint from the statement of Theorem 1.6, it would be possible to establish an equivalence between the derandomization of prBPP and algorithms for

[8] Note that the result in this item subsumes the "in particular" result from the previous item.

Heavy Avoid (in both the implicit and explicit settings). While obtaining this strong characterization remains elusive, in the next result we get a non-trivial derandomization consequence from the existence of an efficient algorithm for Heavy Avoid without assuming a circuit depth bound.

Let $\mathsf{GAP\text{-}SAT}$ denote the promise problem where YES instances are Boolean circuits with at least half of assignments being satisfying, and NO instances are unsatisfiable Boolean circuits. It is well known that $\mathsf{GAP\text{-}SAT}$ is complete for the promise version of RP.

Theorem 1.7 (Informal)**.** *Let $\delta(N) = o(1)$ be any function. Suppose there is an algorithm for* `Implicit-`δ`-Heavy-Avoid` *on maps $G\colon \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^N$ (where $N = 2^{n^\epsilon}$) implicitly computable by an input circuit of size $\mathsf{poly}(n)$, where the Heavy Avoid algorithm runs in $\mathsf{poly}(N)$ time and is infinitely-often correct. Then there is an algorithm for $\mathsf{GAP\text{-}SAT}$ that runs in subexponential time and is infinitely-often* correct.*[9]

Theorem 1.6 and Theorem 1.7 are both established using *non-black-box reductions* that make use of recent hardness-randomness tradeoffs. In more detail, as explained in Section 1.2 below, Theorem 1.6 crucially relies on the instance-wise hardness-randomness tradeoff for low-depth circuits of Chen and Tell [CT21], while Theorem 1.7 combines the framework of [CT21] and the "leakage-resilient" hardness-randomness framework of Liu and Pass [LP23]. In contrast to the non-black-box nature of the proofs given for these two results, we show that it will be quite difficult to obtain them using *black-box* reductions. In particular, we show that improving Theorem 1.7 to a polynomial-time Levin reduction [Lev73] would derandomize prBPP.[10] Stated more precisely, if there is an efficient black-box Levin reduction from the search version of $\mathsf{GAP\text{-}SAT}$ to Heavy Avoid (even with respect to non-uniform explicit samplers), then $\mathsf{prBPP} = \mathsf{prP}$ holds *unconditionally*. We refer to Section 5.3 for more details.

Finally, we establish a deeper connection between the implicit non-uniform variant of Heavy Avoid considered in this section and the recent paradigm of *instance-wise* hardness-randomness tradeoffs alluded to above [CT21, LP22, LP23, CTW23]. Roughly speaking, in this paradigm, we convert a hard function $f : \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ with multiple output bits into pseudorandomness, where the obtained derandomization is *instance-wise*: for every $x \in \{0,1\}^n$, if f is hard to compute on x, then the derandomization of the corresponding computation over input x succeeds. Naturally, the derandomization assumptions used in these results need *almost-all-inputs hardness*, meaning that f is hard on *all but finitely many* inputs (instead of input lengths).[11] In Section 5.4, we prove that the existence of efficient deterministic algorithms for Heavy Avoid in the implicit non-uniform setting is *equivalent* to the existence of functions f with multiple output bits that are easy to compute deterministically but are hard against fixed polynomial-size randomized algorithms. This result sheds light into the relevance of the techniques that we employ to prove Theorem 1.6 and Theorem 1.7, and suggest that developing further connections between Heavy Avoid and these modern hardness-randomness tradeoffs paradigms could be a fruitful research direction.

[9] In Theorem 1.7, we only obtain $\mathsf{GAP\text{-}SAT}$ algorithms satisfying a technical condition called infinitely-often* correctness, which is a nonstandard variant of infinitely-often correctness. The crucial difference is that, for a sequence of inputs $\{x_n\}_{n\in\mathbb{N}}$, given 1^n, the algorithm is allowed to inspect every input $x_1, x_2, \ldots, x_{\mathsf{poly}(n)}$, and needs to provide a solution for x_n. In other words, the algorithm is correct *infinitely-often** if it outputs the correct answer on infinitely many input lengths n while having access to all input strings from the sequence that have length polynomial in n. We refer the reader to Definition 5.7 and to the proof of Theorem 5.8 for more details.

[10] Recall that in a Levin reduction between search problems we have a pair (f, g) of functions, where f maps to an instance of the other problem while g converts a given solution into a solution to the original problem.

[11] Compared with classical hardness-randomness frameworks such as [NW94, IW97, STV01], the advantage of the new paradigm is that lower bounds against *uniform algorithms* (instead of non-uniform circuits) suffice for *worst-case* derandomization.

1.2 Techniques

We now discuss the proofs of Theorem 1.1, Theorem 1.4, Theorem 1.6, and Theorem 1.7. We make use of a variety of techniques to establish these results:

- The proof of Theorem 1.1 Item (*iii*) relies on extremely efficient *instance checkers* for a special PSPACE-complete problem investigated in [Che23]. This allows us to establish equivalences for very weak circuits classes $\mathcal{C}$ at the frontier of existing separations. Extending the equivalence result to NP, EXP, and $\mathsf{EXP}^{\mathsf{NP}}$ in the context of weak circuit classes poses some additional challenges that we address through different ideas and techniques.

- In the proof of Theorem 1.4 Item (*i*), we use an instance checker to design a polynomial-time pseudodeterministic algorithm for the $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem. To our knowledge, this is the first application of instance checkers in the *design* of an algorithm for a natural problem. On the other hand, the algorithms in Items (*ii*) and (*iii*) of Theorem 1.4 explore a connection to *randomized time-bounded Kolmogorov complexity* [Oli19, LO22] and its *source coding theorem* [LO21, LOZ22].

- The proof of Theorem 1.6 relies on a novel application of the Chen–Tell *non-black-box hitting set generator* construction from [CT21, CLO$^+$23]. In contrast to previous applications, here the *reconstruction procedure* of the generator itself, as well as the assumed algorithm for Heavy Avoid, plays a key role in the specification of a "hard" function.

- Finally, the proof of Theorem 1.7 builds on the proof of Theorem 1.6. It combines for the first time the Chen–Tell derandomization framework [CT21] with the *leakage resilience* derandomization framework of [LP23], using a win-win analysis. We show that either the Heavy Avoid algorithm is leakage resilient, which allows us to use the framework of [LP23], or it can be implemented by a low-depth circuit, which allows us to use the framework of [CT21]. This enables us to derive a non-trivial derandomization consequence without the circuit depth constraint present in the hypothesis of Theorem 1.6.

Next, we describe some of our proofs and techniques in more detail.

Sketch of the Proof of Theorem 1.1. We first explain the proof of Item (*iii*), i.e., the equivalence between the complexity separation $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ and the existence of (infinitely often) logarithmic-space algorithms for $\mathsf{Heavy\text{-}Avoid}$ over implicit $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-samplers.

First, we show how to obtain the separation using algorithms for the implicit heavy avoid problem. Using standard arguments, it suffices to show that for every choice of $k \geq 1$, there is $L \in \mathsf{DSPACE}[n^2]$ such that L cannot be computed by $\mathsf{DTIME}[k \cdot \log n]$-uniform randomized $\mathcal{C}$-circuits of size n^k.

Let $N = 2^n$. We consider a map $G_N \colon \{0,1\}^{n^{O(k)}} \to \{0,1\}^N$ that views its input string x as a pair (M, r), where M is a short encoding (say, $\log n$ bits) of a clocked machine running in time $10k \cdot \log n$, and r is a random string. Let D_M be the $\mathcal{C}$-circuit of size at most n^{2k} whose direct connection language is encoded by the machine M. For $i \in [N]$, we define the i-th output bit of $G_N(x)$ as $D_M(r, i)$. Due to its running time, the computation of M can be uniformly converted into an AC^0 circuit of size at most n^{10k}. Using that $\mathcal{C}$ is a nice circuit class, G_N can be implicitly computed by a $\mathsf{DLOGTIME}$-uniform probabilistic $\mathcal{C}$-circuit C_N of size at most $n^{O(k)}$.

Let $B(1^N)$ be an algorithm of space complexity $O(\log N)$ that solves $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on infinitely many values of N for the sequence G_N, and let L_B be the language defined by B. Note that L_B is in $\mathsf{DSPACE}[O(n)] \subseteq \mathsf{DSPACE}[n^2]$. To argue that L_B cannot be computed by

$\mathsf{DTIME}[k \cdot \log n]$-uniform randomized $\mathcal{C}$-circuits of size n^k, it is enough to show that for every language L computed by such circuits, each string in the sequence $\{y_N^L\}_N$ of truth-tables obtained from L is δ-heavy in $G_N(\mathcal{U}_{m(N)})$. Since B solves $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` for the sequence $\{G_N\}$, it follows that $L_B \neq L$.

The proof that the sequence $\{y_N^L\}_N$ of truth-tables obtained from L is δ-heavy in $G_N(\mathcal{U}_{m(N)})$ relies on the definition of G_N. In more detail, under the assumption that L admits $\mathsf{DTIME}[k \cdot \log n]$-uniform randomized $\mathcal{C}$-circuits of size n^k, it is not hard to show that its truth-table is produced with probability comparable to $2^{-|M|}$. However, this probability is sufficiently large under the assumption that the encoding length $|M|$ is small in the definition of G_N.

The proof of the other direction in Theorem 1.1 is more interesting. We establish the contrapositive. Suppose that for some $G_N \colon \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size $\mathsf{poly}(n)$, every algorithm $A(1^N)$ running in space $O(\log N)$ fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on every large enough input length N. We employ this assumption to show that $\mathsf{PSPACE} \subseteq \mathsf{BP}\text{-}\mathcal{C}$. For this, we recall the notion of *instance checkers*. Let $L \subseteq \{0,1\}^*$ be a language, and let $\{C_n^{(-)}(x,z)\}_{n \in \mathbb{N}}$ be a family of probabilistic oracle circuits. We say that C is an *instance checker* for L if for every input $x \in \{0,1\}^*$:

1. $\mathbf{Pr}_z[C_{|x|}^L(x,z) = L(x)] = 1$, and
2. for every oracle $\mathcal{O}$, $\mathbf{Pr}_z[C_{|x|}^{\mathcal{O}}(x,z) \notin \{L(x), \perp\}] \leq 1/2^n$.

We will rely on an appropriate PSPACE-complete language $L^\star$ that admits highly efficient instance checkers computable in any nice circuit class. This is a consequence of a result from [Che23], as explained in Appendix B.

We then consider a candidate algorithm $A(1^N)$ that computes as follows. On input 1^N, define tt_N to be the truth table of $L^\star$ on n-bit inputs; we simply output tt_N. It is possible to show that A computes in space $O(\log N)$ after an appropriate scaling of parameters, which we we omit here for simplicity. Therefore, A fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on every large enough input length N. This means that for every large enough N, the probability of tt_N under the distribution $G_N(\mathcal{U}_{\mathsf{poly}(n)})$ from above is at least $\delta = 1/(\log N)^{O(1)} = 1/\mathsf{poly}(n)$.

To explain how we compute $L^\star$ on an input $x \in \{0,1\}^n$, assume for simplicity that the oracle instance checker circuit (call it IC) only queries its oracle on input length n. We sample $v := n^{O(1)}$ strings $z_1, \ldots, z_v \in \{0,1\}^{\mathsf{poly}(n)}$ uniformly and independently at random, and for each string z_i, we define an oracle $\mathcal{O}_i$ whose truth table is the string $G_N(z_i) \in \{0,1\}^N$. We run IC in parallel and obtain $b_i := \mathsf{IC}_n^{\mathcal{O}_i}(x)$ for each $i \in [v]$. We output 1 if at least one bit among $b_1, \ldots, b_v$ is 1, and 0 otherwise.

Next, we argue that A computes $L^\star$ with high probability. Let tt_N denote the truth table of $L^\star$ on input length n. By our choice of v, with high probability the string tt_N appears among the strings $G_N(z_1), \ldots, G_N(z_v)$, meaning that one of the oracles $\mathcal{O}_i$ computes $L^\star$ on inputs of length n. Consequently, in this case, if $L^\star(x) = 1$ then at least one bit $b_i = 1$, and the procedure outputs 1. On the other hand, if $L^\star(x) = 0$, then by a union bound over the internal randomness of IC, with high probability every bit $b_i \in \{0, \perp\}$. In this case, the procedure outputs 0. This establishes the correctness of A. Using the efficiency of the instance checker and that $\mathcal{C}$ is a nice circuit class, it is also possible to upper the circuit complexity of A and to analyze the uniformity of the corresponding circuits. This implies that $L^\star \in \mathsf{BP}\text{-}\mathcal{C}$. Since $L^\star$ is PSPACE-complete under $\mathsf{DLOGTIME}$-uniform projection reductions, we get that $\mathsf{PSPACE} \subseteq \mathsf{BP}\text{-}\mathcal{C}$, as desired.

We now briefly comment on the additional ideas needed for the proofs of the other items in Theorem 1.1. The proof of Item (*ii*) requires a different approach, since instance checkers for $\mathsf{EXP}^{\mathsf{NP}}$-complete languages are not known. We provide two different proofs in this case. In more detail, the result for $\mathsf{EXP}^{\mathsf{NP}}$ can be obtained using a win-win argument and a reduction to Item (iii), or through the use of *selectors* for $\mathsf{EXP}^{\mathsf{NP}}$-complete languages [Hir15]. These two approaches provide different extensions of the result, which we discuss in detail in Section 3.3. On the other hand, the proof of Item (iv) relies on a randomized depth-efficient version of the search-to-decision reduction for SAT based on the Valiant-Vazirani Isolation Lemma [VV86], as well as the equivalence between the polynomial hierarchy and $\mathsf{DLOGTIME}$-uniform constant-depth circuits of exponential size [BIS90].

Sketch of the Proof of Theorem 1.4. First, we discuss the proof of Item (i) in the case where $\mathcal{C}$ = "general Boolean circuits". Consider a map $G_N \colon \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^N$ that is implicitly computable in time $\mathsf{poly}(n)$. We consider two cases, based on whether $\mathsf{EXP} = \mathsf{BPP}$.

If $\mathsf{EXP} \not\subseteq \mathsf{BPP}$, then by Theorem 1.1 Item (i) (with circuit class $\mathcal{C}$ = "general Boolean circuits") the heavy avoid problem over G_N can be solved in deterministic polynomial time (i.e., in time $\mathsf{poly}(N)$) on infinitely many input lengths. Note that the correctness of the procedure obtained from Theorem 1.1 Item (i) relies on the existence of instance checkers for EXP-complete languages.

In the remaining case, assume that $\mathsf{EXP} \subseteq \mathsf{BPP}$. Let $B(j)$ be the following deterministic machine with input $j \in \{0,1\}^n$: It first goes over all choices of $x \in \{0,1\}^{\mathsf{poly}(n)}$ and computes $G_N(x)$, then calculates the probability of each string in $\{0,1\}^N$ produced in this way, and finally outputs the j-th bit of the lexicographic first string y such that $\mathbf{Pr}[G_N(\mathcal{U}_{\mathsf{poly}(n)}) = y] \leq \delta$. Note that B runs in time exponential in n, its input length. Therefore, it defines a language $L_B \in \mathsf{EXP}$. By the assumption, $L_B \in \mathsf{BPP}$. Consequently, we can compute $y \in \{0,1\}^N$ from 1^N in pseudodeterministic time $\mathsf{poly}(N)$. Note that in this case the algorithm succeeds on every input length.[12]

More generally, to obtain the result claimed in Item (i) for a nice circuit class $\mathcal{C}$, we use a similar approach but consider whether $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ instead (see Section 4.1).

In contrast, the proof of Theorem 1.4 Item (iii) relies on ideas from randomized time-bounded Kolmogorov complexity. More specifically, we consider the randomized Levin complexity of a string $y \in \{0,1\}^n$ [Oli19], denoted $\mathsf{rKt}(y)$. Roughly speaking, $\mathsf{rKt}(y)$ measures the minimum description length of a time-bounded machine that outputs y with high probability. Our key idea is that, by the coding theorem of [LO21], if a string y can be sampled in polynomial time with probability at least δ, then $\mathsf{rKt}(y) = O(\log 1/\delta)$. Consequently, to avoid the set of heavy strings produced by a polynomial-time samplable distribution $\mathcal{D}$, it is sufficient to construct a string z such that $\mathsf{rKt}(z) \geq C \cdot \log n$, where C is large enough. In order to implement this idea, we employ a related sub-exponential time pseudodeterministic construction of strings of large rKt complexity from [LOS21].

Finally, the proof of Theorem 1.4 Item (ii) is obtained via a translation to Item (iii), using a simple "prefix" reduction described in Section 4.3.

Sketch of the Proof of Theorem 1.6. Using existing results [BF99], in order to derandomize prBPP it is sufficient to describe an algorithm that, given an input circuit $D \colon \{0,1\}^M \to \{0,1\}$ of size $O(M)$ with the promise that $\mathbf{Pr}_y[D(y) = 1] \geq 1/2$, runs in deterministic time $\mathsf{poly}(M)$ and outputs a positive input of D. To achieve this, we will rely on a novel application of the Chen–Tell generator [CT21] (with the improved parameters from [CLO+23]). In more detail, given a function $f \colon \{0,1\}^n \to \{0,1\}^{T(n)}$ computed by logspace uniform circuits of size $T(n)$ and depth $d(n)$, and a parameter $M(n)$ such that $c \cdot \log T \leq M \leq T^{1/c}$ (for a constant c), [CT21, CLO+23] provides algorithms HSG_f and Recon_f depending on f such that:

[12] We remark that using a more involved construction that employs the instance checker as a subroutine, one can simultaneously consider both cases to describe a single explicit algorithm that succeeds infinitely often. We omit the details. For a similar situation where a non-constructive win-win argument can be turned into an explicit algorithm, see [OS17, Section 3.4].

- The algorithm $\mathsf{HSG}_f(x)$ runs in deterministic T^c time and outputs a set of M-bit strings.
- Given $x \in \{0,1\}^n$ and $i \in [T]$ as inputs, and oracle access to a candidate distinguisher $D\colon \{0,1\}^M \to \{0,1\}$, $\mathsf{Recon}_f^D(x,i)$ runs in randomized $(dnM)^c$ time. If D is dense and avoids $\mathsf{HSG}_f(x)$, then with probability $\geq 1 - 2^{-M}$, $\mathsf{Recon}_f^D(x,i)$ outputs the i-th bit of $f(x)$.

We consider an appropriate function $f'\colon \{0,1\}^{\widetilde{O}(M)} \to \{0,1\}^N$, where $N = M^{C_1}$ for a large enough constant C_1. We view the input of f' as the description of an arbitrary circuit $D\colon \{0,1\}^M \to \{0,1\}$ of size $O(M)$. In this construction, the parameter $T = M^{C_2}$ for a large enough constant $C_2 > C_1$, while $d = M^{o(1)} = N^{o(1)}$. Moreover, f' will be computed by a logspace-uniform family of circuits. We then show that $\mathsf{HSG}_{f'}(D)$ hits D if D is a dense circuit. Note that the generator runs in time $\mathsf{poly}(T) = \mathsf{poly}(M)$ by our choice of parameters.

The function f' makes use of the algorithm $\mathcal{A}$ that solves the $\texttt{Implicit-}\delta\texttt{-Heavy-Avoid}$ problem on instances $G\colon \{0,1\}^{N^\epsilon} \to \{0,1\}^N$ that are implicitly computable in N^ϵ size. In more detail, we let $f'(D) = \mathcal{A}(C_D)$, where C_D is an implicit (non-uniform) sampler of size N^ϵ for a map $G_D\colon \{0,1\}^{N^\epsilon} \to \{0,1\}^N$ described next.

First, we make a simplifying assumption: The sampler G_D has access to the code of a machine $M_{f'}$ that serves as a logspace-uniform description of a circuit family that computes f'. (Observe that this is self-referential, since we have defined $f'(D) = \mathcal{A}(C_D)$ above, while we will also use f' to define C_D. We will handle this issue later.)

The sampler G_D stores D as advice. This is possible because D is of size M, and if C_1 is large enough then $M \ll N^\epsilon$. The implicit sampler $C_D(r,i)$ for G_D then uses its random input string r of length N^ϵ and $i \in [\log N]$ to compute $\mathsf{Recon}_{f'}^D(D,i,r)$, where we have made explicit the random string r used by $\mathsf{Recon}_{f'}^D$. Since $d = M^{o(1)}$ and C_1 is large enough, we get that $\mathsf{Recon}_{f'}^D(D,i,r)$ can be computed in time $(d \cdot M^{1+o(1)} \cdot M)^c \leq M^{c+o(1)} \leq N^\epsilon$. This completes the definition of $f'(D)$ and of $\mathsf{HSG}_{f'}(D)$. We note that to establish the size, depth, and logspace-uniformity of the sequence of circuits computing f' we can rely on the fact that f' only needs to produce the *code* of C_D.[13]

Next, we argue that $\mathsf{HSG}_{f'}(D)$ hits any dense circuit D. Assume this is not the case. Then, since D avoids the generator, $\mathsf{Recon}_f^D(D,i)$ outputs the i-th bit of $f'(D)$ with probability at least $1-2^{-M}$. Consequently, by a union bound over $i \in [N]$, it follows that the string $\mathcal{A}(C_D) = f'(D) \in \{0,1\}^N$ is output by $\mathsf{Recon}_f^D(D,\cdot)$ with probability $1 - o(1)$. In other words, the string $f'(D)$ is sampled with high probability by the sampler G_D encoded by C_D. On the other hand, since $f'(D) = \mathcal{A}(C_D)$ and $\mathcal{A}$ solves the heavy avoid problem for G_D, we get that the string $f'(D)$ has probability $o(1)$ under G_D. This contradiction implies that $\mathsf{HSG}_{f'}(D)$ indeed hits D.

It remains to explain how to fix the self-referential nature of the definition of G_D via the implicit sampler C_D, which depends on f' (and which in turn depends on C_D). In more detail, the construction is self-referential due to the use of the routine $\mathsf{Recon}_{f'}^D$, which depends on f'. To patch the argument, we combine the following key points:

- There is a deterministic algorithm that, given the Turing machine $M_{f'}$ that prints the circuit for f' in logspace, outputs the description of $\mathsf{Recon}_{f'}$ in $\mathsf{poly}(|\langle M_{f'}\rangle|)$ time.
- We can combine constantly many samplers into a single sampler that produces the convex combination of the corresponding distributions. A string with weight $o(1)$ under the new distribution must have weight $o(1)$ under each original sampler.

[13] We make a brief comment about the novelty of this argument. In order to define the "hard" function f', here we make use of the *reconstruction procedure* of the generator. This is different from an application of this generator in [CLO+23], where the code of the *hitting set procedure* plays a key role in the definition of the hard function.

Therefore, we can change the description of G_D so that it interprets a small prefix of its random input string as the description of a Turing machine M_f that prints a circuit of the expected size using logarithmic uniformity, then use the first bullet above to produce the procedure Recon_f corresponding to f. Notice that with this change the sampler G_D no longer depends on f'. Moreover, since f' is encoded by some finite machine $M_{f'}$, using the second bullet the argument described above to reach a contradiction and establish the correctness of the hitting set generator still holds: When D avoids $\mathsf{HSG}_{f'}(D)$ the modified sampler G_D outputs the string $f'(D) = \mathcal{A}(C_D)$ with constant probability, while as a solution to the heavy avoid problem for G_D this string has probability $o(1)$. This completes the sketch of the argument.[14]

Sketch of the Proof of Theorem 1.7. Since this is a more sophisticated construction, we only provide a brief sketch of the idea. As alluded to above, the argument combines the two instance-wise hardness-randomness tradeoffs introduced by Chen and Tell [CT21] and by Liu and Pass [LP23], respectively.

We employ a win-win analysis based on whether the assumed algorithm for $\texttt{Implicit-}\delta\texttt{-Heavy-Avoid}$ (call it Avoid) is "*leakage resilient*" hard. In more detail, let $f\colon \{0,1\}^n \to \{0,1\}^T$ be a function, A be a randomized algorithm, and $x \in \{0,1\}^n$ be an input of f. We say that $f(x)$ is *ℓ-leakage resilient hard* against A if for every "leakage string" $\mathsf{leak} \in \{0,1\}^\ell$, there is some $i \in [T]$ such that $\mathbf{Pr}[A(x, \mathsf{leak}, i) = f(x)_i] \le 2/3$, where the probability is taken over the internal randomness of A. Liu and Pass [LP23] showed that leakage resilient hardness can be used for derandomization.

We can now explain the main idea behind the win-win analysis. If Avoid is leakage resilient hard, we use the hardness-randomness tradeoffs in [LP23]. If this is not the case, we show that Avoid can actually be implemented by a low-depth circuit. We can then use the hardness-randomness tradeoffs in [CT21], which requires the hard function to be computed by a low-depth circuit family.

Implementing this plan turns out to require a delicate construction and the notion of infinitely-often* correctness appearing in the statement of Theorem 1.7. We refer to Section 5.2 for more details.

1.3 Related Work

Our work relates to several recent lines of research in algorithms and complexity theory.

Algorithmic Characterizations of Uniform Lower Bounds. The algorithmic method of Williams [Wil13a], which derives $\mathcal{C}$-circuit lower bounds for NEXP or $\mathsf{EXP}^{\mathsf{NP}}$ from non-trivial algorithms for Satisfiability or $\mathsf{GAP\text{-}SAT}$ for $\mathcal{C}$ circuits, has been successful in showing several new circuit lower bounds [Wil14, Wil18b, Wil18a, MW20, CW19, CLW20]. However, in settings where *non-uniform* lower bounds are unknown, it is unclear how to use such methods to at least give uniform randomized lower bounds. A step towards such methods is to give *algorithmic characterizations* of uniform lower bounds, which show that certain algorithmic results are both necessary and sufficient for lower bounds. An algorithmic approach to the NEXP vs BPP problem is given in [Wil13b], but this does not seem to give a characterization. An algorithmic characterization of $\mathsf{EXP} \neq \mathsf{BPP}$ follows from [IW01], but this characterization does not extend to frontier lower bound questions such as $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{BP\text{-}TC}^0$ and $\mathsf{EXP} \not\subseteq \mathsf{BP\text{-}ACC}^0$. Theorem 1.1 gives generic characterizations that apply to these and other frontier questions – this showcases the benefits of considering

[14]We note that this argument is non-black-box. The code of a machine $M_{f'}$ that describes a uniform circuit family for f' is needed to instantiate the Chen-Tell generator. In the aforementioned construction, this means that black-box access to the algorithm $\mathcal{A}$ is not enough.

the Heavy Avoid problem rather than the previously studied Gap-SAT or Circuit Acceptance Probability Problem (CAPP). Theorem 1.1 also gives characterizations of uniform lower bounds for NP, where no algorithmic characterizations at all where known before. An algorithmic approach to uniform lower bounds for NP is given in [San23], but the method there does not seem to extend to randomized lower bounds, and moreover does not give unconditional algorithmic characterizations.

Range Avoidance. There have been several recent works on the Range Avoidance problem [KKMP21, Kor21, RSW22, GLW22, CHLR23, ILW23, GGNS23, CL24, Kra24], which is a total search problem where we are given a circuit C from n bits to m bits, $m > n$, and need to output an m-bit string that is not in the range of C.[15] The Range Avoidance problem is tightly connected to proving non-uniform lower bounds (see, e.g., [Jeřo4, Kor21, RSW22, CHR24, Li24, Kra24]). Heavy Avoid can be thought of as an easier version of Range Avoidance, where we are asked to output some m-bit string that does not have *many* pre-images, rather than a string that has no pre-images at all. Indeed, for δ that is inverse polynomial, Heavy Avoid with parameter δ is a BPP search problem, which is unknown for Range Avoidance and would have new circuit lower bound consequences if it were the case. We refer to Appendix A for reductions from both Heavy Avoid and Heavy Find to Range Avoidance. One of our motivations for defining and studying Heavy Avoid is that algorithms for this easier problem might give a way to exploit uniformity in the lower bound.

Quantified Derandomization. In the quantified derandomization setting [GW14, Tel22], we are interested in the possibility of derandomizing algorithms with overwhelming success probability, e.g., derandomization of randomized algorithms for a decision problem that err on at most $S(n)$ random bit sequences, for some $S(n)$ that is sub-exponential or even just slightly super-polynomial. Note that this derandomization setting is quite specialized, since bounded-error randomized algorithms are in general allowed to err on an inverse polynomial fraction of all random bit sequences. A naive way to derandomize such algorithms for decision problems is to run the randomized algorithm on the lexicographically first $2S(n) + 1$ random bit sequences, and take the majority answer. The question of when it is possible to do better has been studied extensively [GW14, Tel17, Tel18, CT19, Tel22]. Our Corollary 1.3 identifies an interesting phenomenon for quantified derandomization of BPP search problems, which has not been studied before. In this setting, the previously mentioned "naive" method to derandomize doesn't work, as verifying a candidate solution itself involves the use of randomness. We show that for any $\delta = o(1)$ there is a natural search problem, i.e., Heavy Avoid with parameter δ, such that the problem can be solved by a randomized algorithm which errs on at most $1/\delta = \omega(1)$ random bit sequences, but any efficient randomized algorithm which errs on at most $O(1)$ random bit sequences would imply $\mathsf{EXP} \neq \mathsf{BPP}$! Thus even slightly beating the performance of a known algorithm in a quantified derandomization sense would imply a breakthrough lower bound.

Pseudodeterministic Algorithms. The notion of *pseudodeterminism* was introduced in the influential work of [GG11]. A pseudodeterministic algorithm for a search problem is a randomized algorithm that outputs some fixed solution to the search problem with high probability. There has been a lot of recent work on pseudodeterminism in various settings, as well as connections to complexity theory (see, e.g., [GGR13, OS17, DPV18, OS18, GL19, DPV21, GIPS21, LOS21, DPWV22, CDM23]). In general, one might hope to show that every BPP search problem can

[15]The problem is closely related to the dual weak pigeonhole principle, which has been widely investigated in logic and bounded arithmetic (see [Kra95, Jeřo4, Jeřo5, Kra24] and references therein).

also be solved pseudodeterministically, but this is not known, though there are important special cases such as finding an N-bit prime for which efficient pseudodeterministic algorithms are known infinitely often [CLO+23]. Theorem 1.4 gives unconditional pseudodeterministic algorithms for Heavy Avoid that can be implemented in low depth when the circuit sampler for Heavy Avoid is itself low depth. Moreover, improving our strongest pseudodeterministic algorithms would have immediate consequences for the long-standing open problem of showing a hierarchy for randomized time (Proposition 1.5). Note that unlike the problem of finding an N-bit prime, Heavy Avoid is a fairly powerful BPP search problem, as evidenced by Theorems 1.6 and 1.7.

Minimal Assumptions for Derandomization. The standard "hardness vs. randomness" approach towards $\mathsf{prBPP} = \mathsf{prP}$ requires lower bounds against non-uniform circuits [NW94, IW97, Uma03]; another line of work employs uniform lower bounds such as $\mathsf{EXP} \neq \mathsf{BPP}$ to obtain heuristic (i.e., average-case) derandomization of BPP [IW01, TV07, CRTY23, CRT22]. A long-standing open problem is whether circuit lower bounds such as $\mathsf{EXP} \not\subseteq \mathsf{SIZE}[\mathsf{poly}]$ are indeed *necessary* for derandomization (see, e.g., [Gol11] and [CRTY23]). Recently, Chen and Tell [CT21] proposed an *instance-wise* hardness-randomness tradeoff and showed that *almost-all-inputs* hardness suffices for worst-case derandomization. The Chen-Tell result has already sparked a new line of research on instance-wise hardness-randomness tradeoffs and minimal assumptions for derandomization [LP22, LP23, CRT22, CTW23, vMS23] (see also the survey [CT23]). As demonstrated in Theorems 1.6 and 1.7, these instance-wise hardness-randomness tradeoffs can be used as *proof techniques* to connect Heavy Avoid to the problem of derandomizing prRP or prBPP. Moreover, as we show in Section 5.4, the *almost-all-inputs* hardness assumptions used in [CT21] are closely connected to Heavy Avoid. Given the rich interplay between Heavy Avoid and derandomization, we believe that investigating the complexity of Heavy Avoid is likely to shed further light on the minimal assumptions required for derandomization.

Acknowledgements. We thank the anonymous reviewers for their comments and suggestions regarding the presentation. Igor C. Oliveira received support from the Royal Society University Research Fellowship URF\R1\191059; the UKRI Frontier Research Guarantee Grant EP/Y007999/1; and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick.

2 Preliminaries

2.1 Notation

We use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$. For a distribution D and an element x, we use $D(x)$ to denote the probability of x under D.

We say that a probability distribution D contains a *δ-heavy element* if there is x in the support of D such that $D(x) \geq \delta$. Any such element x is said to be *δ-heavy*. In this case, we also say that the distribution D is δ-heavy. If an element x is not δ-heavy, then we say it is *δ-light*.

We will often consider a distribution ensemble $\mathcal{D} = \{D_n\}_{n \geq 1}$, where each D_n is a distribution supported over $\{0,1\}^n$. For convenience, we might simply refer to $\mathcal{D}$ as a distribution. We let PSAMP denote the set of polynomial-time samplable distributions.

We say a probabilistic algorithm $\mathcal{A}$ for a search problem $\mathcal{P}$ is *pseudodeterministic* [GG11], if for every input x, there is a *canonical* $\mathcal{P}$-solution y of x such that $\mathcal{A}(x)$ outputs y with probability

$\geq 2/3$. It is easy to see that the success probability can be amplified to $1 - \exp(-n)$ by parallel repetition.

2.2 The Heavy Avoid Problem

In general, given a distribution $\mathcal{D}$ over $\{0,1\}^n$ and a parameter $\delta \in (0,1)$, the Heavy Avoid problem asks to find a string $x \in \{0,1\}^n$ such that $\mathcal{D}(x) < \delta$. It is easy to see that such a string always exists as long as $2^n > 1/\delta$. Consequently, the Heavy Avoid problem is a *total* search problem. This paper mainly focuses on the regime where $1/\delta$ is significantly smaller than 2^n, such as $\delta = 1/\mathsf{poly}(n)$ or even just $\delta \approx 1/\log n$.

In this paper, we will consider the Heavy Avoid problem in different settings, depending on whether the sampler for $\mathcal{D}$ *implicitly* samples the distribution and whether it is computed *uniformly*.[16]

- We say a distribution $\mathcal{D}$ over $\{0,1\}^n$ is *implicit* (or, *locally-samplable*) if there is an efficient procedure that given an integer i and the randomness r used by the sampler, outputs the i-th bit of the sample according to r. In the typical parameter regime, $\mathcal{D}$ runs in time $t \approx \mathsf{poly}(\log n, |r|)$ which is much smaller than n. Depending on the context, "efficient procedure" could either mean Turing machines or circuits, as will be addressed in the next bullet. Note that the random string r is also short and the sampler has sequential access (instead of random access) to r.

  We use the word *explicit* to describe samplers that take $\mathsf{poly}(n)$ time, as opposed to implicit samplers.

- We say a family of distribution $\mathcal{D} = \{D_n\}_{n \in \mathbb{N}}$ is *uniformly samplable* if there is a Turing machine M that given 1^n (and access to uniformly random bits), samples from D_n. (Similarly, we often consider a Turing machine $M(1^n)$ that prints a circuit that samples D_n.) On the other hand, if we only have a (non-uniform) family of circuits $\{C_n\}$, where each C_n samples from D_n, then we say the distribution is *non-uniformly samplable.*

  We will also say that uniformly samplable distributions are sampled in *time* t, while non-uniformly samplable distributions are sampled in *size* t.

Explicit Maps

Definition 2.1 (Uniform Heavy Avoid)**.** Let $\mathcal{D} = \{D_n\} \in \mathsf{PSAMP}$, where each D_n is supported over $\{0,1\}^n$, and let $\delta(n) \in [0,1]$. In the $(\mathcal{D},\delta)$-`Heavy-Avoid` problem, given 1^n the goal is to output an element $x \in \{0,1\}^n$ such that $D_n(x) < \delta(n)$.

We say that the $(\mathcal{D},\delta)$-`Heavy-Avoid` problem can be solved in polynomial time if there is a deterministic algorithm $A(1^n)$ that runs in polynomial time and solves $(\mathcal{D},\delta)$-`Heavy-Avoid`. Similarly, the $(\mathcal{D},\delta)$-`Heavy-Avoid` problem can be solved in pseudodeterministic polynomial time if there is a pseudodeterministic algorithm $A(1^n)$ that runs in polynomial time and solves $(\mathcal{D},\delta)$-`Heavy-Avoid`.

Definition 2.2 (Non-Uniform Heavy Avoid)**.** Let $C\colon \{0,1\}^m \to \{0,1\}^n$ be a Boolean circuit, and let D_C be the distribution induced by $C(\mathcal{U}_m)$. Let $\delta \in [0,1]$. In the `Non-Uniform-Heavy-Avoid` problem, given C and δ, the goal is to output an element $x \in \{0,1\}^n$ such that $D_C(x) < \delta$.

[16]Do not confuse the uniformity of the sampler with the distribution D_n, which most often in this work will not be the uniform distribution.

We may also represent $1/\delta$ in unary when we want to emphasize that we consider the regime where $\delta \geq 1/\mathsf{poly}(n)$. In this case, the input consists of $(C, 1^t)$, let $\delta := 1/t$, and the goal is to output a δ-light element of D_C.

We say that `Non-Uniform-Heavy-Avoid` can be solved in polynomial time if for every constant $c \geq 1$, `Non-Uniform-Heavy-Avoid` over inputs where the circuit C is of size at most n^c and $\delta \geq 1/n^c$ can be solved in deterministic polynomial time.

Note that it is not hard to solve `Non-Uniform-Heavy-Avoid` with randomness.

Proposition 2.3. *Let $c \geq 1$. There is a probabilistic polynomial-time algorithm A such that, given a circuit $C\colon \{0,1\}^m \to \{0,1\}^n$ of size at most n^c and a parameter $\delta \geq 1/n^c$, A runs in time polynomial in n^c and outputs with high probability a set $T_{C,\delta}$ of size at most $2 \cdot (1/\delta)$ that contains all δ-heavy elements of D_C. Hence, if we output the lexicographically smallest string not in $T_{C,\delta}$, then we obtain a probabilistic polynomial-time algorithm solving* `Non-Uniform-Heavy-Avoid`.

Let $S_{C,\delta} = \{x \in \{0,1\}^c \mid D_C(x) \geq \delta\}$, where $D_C = C(\mathcal{U}_m)$. Note that the algorithm A outputs with high probability a set $T_{C,\delta}$ of bounded size such that $S_{C,\delta} \subseteq T_{C,\delta}$. However, different executions of A might produce different sets $T_{C,\delta}$. Consequently, this does not give rise to a *pseudodeterministic* algorithm for `Non-Uniform-Heavy-Avoid`.

Implicit Maps (Locally Samplable Distributions)

For locally samplable distributions (which will also be called "implicit maps" in this paper), it will be important to fix the following notation. A *map*, or *generator*, is a function $G\colon \{0,1\}^m \to \{0,1\}^N$ (typically $N \gg m$) such that our input distribution is $G(\mathcal{U}_m)$. We say the map is *implicitly computed* by a circuit C if $C\colon \{0,1\}^m \times [N] \to \{0,1\}$ satisfies that for every $r \in \{0,1\}^m$, $C(r,i)$ outputs the i-th bit of $G(r)$. The input of `Implicit-Heavy-Avoid` will be a circuit C even though we are actually solving Heavy Avoid on the corresponding instance G. (In the uniform case, the circuit C is generated by a uniform procedure, in which case the input to the problem is simply 1^N.)

Although the input length, $\mathsf{poly}(|C|)$, is usually much smaller than N, the output length is still N, hence we still measure the time complexity of algorithms solving `Implicit-Heavy-Avoid` by N. For example, we say `Implicit-Heavy-Avoid` can be solved in deterministic polynomial time if it can be solved by a deterministic machine that runs in time polynomial in N.

Definition 2.4 ($\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` for non-uniform samplers)**.** Let $\mathcal{C}$ be a circuit class, $\delta\colon \mathbb{N} \to [0,1]$, $m, N, s\colon \mathbb{N} \to \mathbb{N}$ be parameters. We define the *$\mathcal{C}$-implicit δ-heavy avoid problem* for maps that stretch $m(N)$ bits to N bits and are implicitly computed by $\mathcal{C}$-circuits of size at most $s(N)$. (A typical parameter regime is that $N = 2^n$ for some integer n, $\delta(N) = 1/\mathsf{poly}(n)$, and $m(N), s(N) \leq \mathsf{poly}(n)$. The parameters will be clear in each statement.)

The input of this problem is a size-s $\mathcal{C}$-circuit $C\colon \{0,1\}^m \times [N] \to \{0,1\}$. Recall that this circuit C implicitly defines a map $G\colon \{0,1\}^m \to \{0,1\}^N$ such that, for every $r \in \{0,1\}^m$, and $i \in [N]$, $C(r,i)$ outputs $G(r)_i$ (the i-th bit of the N-bit string $G(r)$). Given C, the goal is to output an element $y \in \{0,1\}^N$ such that $\mathbf{Pr}[G(\mathcal{U}_m) = y] < \delta$.

Similarly, we can also define $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` for uniformly-samplable maps. Here, we consider families of maps $\{G_N\}$ that are implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits $\{C_N\}$ of size at most $s(N)$. In other words, there is a $\mathsf{DLOGTIME}$-uniform sequence $\{C_N\}_{N\geq 1}$ of size-$s(N)$ $\mathcal{C}$-circuits such that, for every $N \geq 1$, $r \in \{0,1\}^{m(N)}$, and $i \in [N]$, $C_N(r,i)$ outputs $G_N(r)_i$, i.e., the i-th bit of the N-bit string $G_N(r)$. (Here, we say $\{C_N\}$ is $\mathsf{DLOGTIME}$-uniform if the *direct connection language* of C_N can be decided in $O(\log s(N))$ time.)

Definition 2.5 ($\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` for uniform samplers)**.** Let $m(N), s(N), \delta(N)$ be parameters as above, and $\{C_N\}$ be a $\mathsf{DLOGTIME}$-uniform sequence of $\mathcal{C}$-circuits that defines a family of maps $\{G_N\}$. That is, given $r \in \{0,1\}^m$ and $i \in [N]$, $C_N(r,i)$ outputs the i-th bit of $G_N(r)$. The $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem corresponding to $\{G_N\}$ is the following problem: Given 1^N the goal is to output a string $x \in \{0,1\}^N$ such that $\mathbf{Pr}_r[G_N(r) = x] < \delta(N)$.

Note that the input to the Heavy Avoid problem is given by a circuit when we consider the non-uniform formulations (in both the implicit and explicit settings), while the input to the problem is simply the input length when we consider uniform formulations (since the sampler can be efficiently obtained from the input length).

2.3 Time-Bounded Kolmogorov Complexity

This section reviews some notions from time-bounded Kolmogorov complexity (see, e.g., [LO22] for more details). Let U be a Turing machine. Given a positive integer t and a string $x \in \{0,1\}^*$, we let

$$\mathsf{K}^t_U(x) = \min_{p \in \{0,1\}^*} \Big\{ |p| \;\Big|\; U(p) \text{ outputs } x \text{ in at most } t \text{ steps} \Big\}.$$

We say that $\mathsf{K}^t_U(x)$ is the *t-time-bounded Kolmogorov complexity of x* (with respect to U). As usual, we fix U to be a time-optimal machine [LV19], i.e., a universal machine that is almost as fast and length efficient as any other universal machine, and drop the index U when referring to time-bounded Kolmogorov complexity measures.

For $x \in \{0,1\}^*$, the *probabilistic t-time-bounded Kolmogorov complexity* of x is defined as

$$\mathsf{pK}^t(x) = \min \left\{ k \in \mathbb{N} \;\middle|\; \mathop{\mathbf{Pr}}_{w \sim \{0,1\}^t} \Big[\exists\, p \in \{0,1\}^k,\, U(p,w) \text{ outputs } x \text{ within } t \text{ steps} \Big] \geq \frac{2}{3} \right\}.$$

In other words, if $k = \mathsf{pK}^t(x)$, then with probability at least $2/3$ over the choice of the random string w, given w the string x admits a t-time-bounded encoding of length k.

We can also consider the *randomized* Kt *complexity* of a string $x \in \{0,1\}^*$, defined as

$$\mathsf{rKt}(x) = \min_{t \in \mathbb{N},\, p \in \{0,1\}^*} \left\{ |p| + \lceil \log t \rceil \;\middle|\; \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^*} [U(p,r) \text{ outputs } x \text{ in } t \text{ steps}] \geq 2/3 \right\}.$$

All these notions of time-bounded Kolmogorov complexity can be generalized to capture the conditional complexity of x given y in the natural way, i.e., by providing y as an extra input string to the universal machine U.

2.4 Pseudorandomness and Derandomization

Fix an input length n. A *generator* is simply a multiset $G \subseteq \{0,1\}^n$. We will consider families of generators $\{G_n\}_{n \in \mathbb{N}}$ where each $G_n \subseteq \{0,1\}^n$ is a generator outputting n-bit strings. In the literature, it is also common to consider these generators as functions: let $\ell(n) < n$ denote the seed length of the generator, then the function $G_n \colon \{0,1\}^{\ell(n)} \to \{0,1\}^n$ is equivalent to the multiset

$$\{G_n(s) : s \in \{0,1\}^{\ell(n)}\}.$$

In this paper, we will use the subset- and functional-definitions of generators interchangeably.

Let $\mathcal{A} : \{0,1\}^n \to \{0,1\}$ be a function, $H \subseteq \{0,1\}^n$ be a generator, and $\epsilon > 0$ be a parameter. We say that $\mathcal{A}$ is *ϵ-dense* if $\mathbf{Pr}_{x \sim \{0,1\}^n}[\mathcal{A}(x) = 1] \geq \epsilon$. We say that $\mathcal{A}$ *ϵ-avoids* H if $\mathcal{A}$ is ϵ-dense, and for every string $x \in H$, we have $\mathcal{A}(x) = 0$. If $\mathcal{A}$ does not ϵ-avoid H, then we say that H *ϵ-hits* $\mathcal{A}$.

Let $\mathcal{A} : \{0,1\}^n \to \{0,1\}$ be a function, $G\colon \{0,1\}^\ell \to \{0,1\}^n$ be a generator, and $\epsilon > 0$ be a parameter. We say that $\mathcal{A}$ *ϵ-distinguishes* G, if

$$\left| \mathop{\mathbf{Pr}}_{x\sim\{0,1\}^n}[\mathcal{A}(x) = 1] - \mathop{\mathbf{Pr}}_{s\sim\{0,1\}^\ell}[\mathcal{A}(G(s)) = 1] \right| > \epsilon;$$

otherwise (if the above inequality does not hold), we say that G *ϵ-fools* A.

Like many papers in derandomization [Gol11, CT21, LP22, LP23], we will consider *promise* versions of randomized complexity classes, such as prRP and prBPP. A *promise problem* [ESY84] $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ is a pair of disjoint sets $(\Pi_{\mathsf{YES}} \cap \Pi_{\mathsf{NO}} = \varnothing)$. A machine solves the corresponding promise problem if given an input $x \in \{0,1\}^*$, it outputs 1 when $x \in \Pi_{\mathsf{YES}}$ and outputs 0 when $x \in \Pi_{\mathsf{NO}}$; note that there is no requirement on the behaviour of the machine when $x \notin (\Pi_{\mathsf{YES}} \cup \Pi_{\mathsf{NO}})$.

We also recall the definitions of the canonical prRP-complete problem $\mathsf{GAP\text{-}SAT}$ and the canonical prBPP-complete problem CAPP.[17]

Definition 2.6 ($\mathsf{GAP\text{-}SAT}$)**.** The problem $\mathsf{GAP\text{-}SAT}$ is the following promise problem $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$: Π_{YES} consists of all circuits $C : \{0,1\}^n \to \{0,1\}$ that are 1/10-dense, and Π_{NO} consists of all circuits $C : \{0,1\}^n \to \{0,1\}$ such that $C(x) = 0$ for every $x \in \{0,1\}^n$.

Definition 2.7 (CAPP)**.** The problem CAPP is the following promise problem $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$: On input (C, δ), where $C : \{0,1\}^n \to \{0,1\}$ is a circuit and $\delta \in (0,1)$ is a number, Π_{YES} consists of (C, δ) where $\delta \geq \mathbf{Pr}_{x\sim\{0,1\}^n}[C(x)] + 1/10$, and Π_{NO} consists of (C, δ) where $\delta \leq \mathbf{Pr}_{x\sim\{0,1\}^n}[C(x)] - 1/10$.

The constant $1/10$ in the above two definitions is arbitrary and can be amplified to $1/\mathsf{poly}(n)$ by parallel repetition.

3 Heavy Avoid and Lower Bounds Against Uniform Probabilistic Circuits

In this section, we study the connection between the `Heavy-Avoid` problem and the problem of proving lower bounds against *uniform probabilistic* circuits. Our main result is that in many settings, lower bounds against uniform probabilistic circuits are characterized by the existence of algorithms for `Implicit-Heavy-Avoid`.

Let $\mathcal{C}$ be a circuit class. A probabilistic $\mathcal{C}$-circuit $E(x; z)$ is a circuit from $\mathcal{C}$ that computes over an input x and an input z, where the latter corresponds to the random choice of E. We denote $\mathsf{BP}\text{-}\mathcal{C}$ the set of languages L that can be computed by a $\mathsf{DLOGTIME}$-uniform sequence of probabilistic $\mathcal{C}$-circuits of polynomial size. We stress that the uniform machine generating the $\mathcal{C}$ circuit is deterministic, while the circuit itself is allowed to make random choices (z).

Our results hold for any (uniform probabilistic) circuit class $\mathcal{C}$ that is *nice*, i.e., satisfies a few technical conditions. More precisely, we say a circuit class $\mathcal{C}$ is *nice* if the following holds:

- **($\mathcal{C}$ contains $\mathsf{AC}^0[\oplus]$.)** $\mathsf{AC}^0[\oplus] \subseteq \mathcal{C}$.
- **($\mathcal{C}$ is closed under composition.)** For every language $L \in \mathcal{C}$ and every $\mathsf{DLOGTIME}$-uniform family of oracle circuits $\{C_n^{(-)}\}_{n\in\mathbb{N}}$ making non-adaptive projection queries where the top (i.e., post-processing) circuit is in $\mathcal{C}$, the language computed by the circuit family $\{C_n^{L_n}\}_{n\in\mathbb{N}}$ is still in $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$.

[17] CAPP stands for "Circuit Acceptance Probability Problem."

- **($\mathcal{C}$ admits universal circuits.)** There is a $\mathsf{DLOGTIME}$-uniform family of $\mathcal{C}$-circuits Eval such that given the description of a $\mathcal{C}$-circuit C (i.e., the truth table of the direct connection language of C) and an input x, $\mathsf{Eval}(\langle C\rangle, x)$ outputs $C(x)$.

  In the case that $\mathcal{C}$ is the union of depth-d circuits for every constant d, such as AC^0 or TC^0, we allow Eval to have higher depth than C: for every fixed depth d, the circuit evaluation problem can be solved by a family of $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuit of constant depth.

It is not hard to check that many standard circuit classes considered in the literature are nice, e.g., $\mathsf{AC}^0[\oplus]$, ACC^0, TC^0, NC^1, $\mathsf{P/poly}$. For instance, universal circuits for NC^1 are constructed in [Bus87], while universal circuits for TC^0 can be built using the universal threshold function (see, e.g., [BW05]) and standard techniques.

A note on notation: throughout this section, when we use parameters n and N together, we implicitly assume $N = 2^n$. We switch back and forth between the two parameters based on which one is more natural in a given context.

3.1 Equivalences for PSPACE via Instance Checkers

Our equivalences for PSPACE follow from the existence of *instance checkers* [BK95, TV07] for PSPACE-complete languages. To establish our equivalences with respect to restricted circuit classes, we use a recent construction of $\mathsf{AC}^0[\oplus]$-computable instance checkers by Chen [Che23].

Below, we say that an oracle circuit $E^{(-)}(x,z)$ from $\mathsf{BP}\text{-}\mathcal{C}$ makes *projection* queries if every query it makes to the oracle can be computed by a projection over the inputs (x,z). After gathering the answers of the oracles, the final output is computed by a $\mathcal{C}$ circuit over (x,z) and these oracle answers. We stress that any oracle circuit that makes projection queries is *non-adaptive*. When we say such a circuit is $\mathsf{DLOGTIME}$-uniform, we mean that both the projection (computing the queries to the oracles) and the top $\mathcal{C}$ circuit are $\mathsf{DLOGTIME}$-uniform.

Theorem 3.1 (A PSPACE-Complete Language with Useful Properties)**.** *There is a language $L^\star \subseteq \{0,1\}^*$ with the following properties:*

1. **(Complexity Upper Bound)** *$L^\star \in \mathsf{PSPACE}$.*

2. **(Completeness)** *$L^\star$ is PSPACE-hard under $\mathsf{DLOGTIME}$-uniform projection reductions.*

3. **(Instance Checkability)** *There is a $\mathsf{DLOGTIME}$-uniform family of $\mathsf{BP}\text{-}\mathsf{AC}^0[\oplus]$ oracle circuits $\{\mathsf{IC}_n\}_{n\geq 1}$ making projection queries such that, on every input string $x \in \{0,1\}^n$ and for every oracle $\mathcal{O} \subseteq \{0,1\}^*$, the following holds:*

   - *$\mathsf{IC}_n^{\mathcal{O}}(x)$ only makes queries of length n to $\mathcal{O}$.*
   - *If $\mathcal{O}$ agrees with $L^\star$ on inputs of length n, then $\mathbf{Pr}_r[\mathsf{IC}_n^{\mathcal{O}}(x;r) = L^\star(x)] = 1$.*
   - *For every oracle $\mathcal{O}$, $\mathbf{Pr}_r[\mathsf{IC}_n^{\mathcal{O}}(x;r) \in \{\perp, L^\star(x)\}] \geq 1 - \exp(-n)$.*

Theorem 3.1 follows from [Che23, Section 7]; we refer the reader to Appendix B for more details.

We say that the $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem corresponding to a given family $\{G_N\}$ of implicitly computed maps can be solved in space $s(N)$ if there is an algorithm A of space complexity $s(N)$ such that, for every input length N, there is some $x \in \{0,1\}^N$ for which $A(1^N, i)$ outputs the i-th bit of x for all $i \in [N]$, and x is a solution to the $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem for G_N.

Theorem 3.2 (Equivalence for PSPACE)**.** *Let $\mathcal{C}$ be a nice class of Boolean circuits. The following statements are equivalent:*

(*i*) $\mathsf{PSPACE} \nsubseteq \mathsf{BP}\text{-}\mathcal{C}$.

(*ii*) *For every choice of $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and for every sequence $\{G_N\}$ of maps $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size at most n^c, the corresponding $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem can be solved in space $O(\log N)$ on infinitely many input lengths N.*

Proof. We consider each implication below.

$(ii) \Rightarrow (i)$. Using the assumption, we show below that for every choice of $k \geq 1$, there is $L \in \mathsf{DSPACE}[n^2]$ such that L cannot be computed by $\mathsf{DTIME}[k \cdot \log n]$-uniform randomized $\mathcal{C}$-circuits of size n^k.[18] Since there exist PSPACE-complete problems, a standard argument shows that this implies $\mathsf{PSPACE} \nsubseteq \mathsf{BP}\text{-}\mathcal{C}$.

Fix a large enough $k \geq 1$, and consider the map $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ defined as follows, where $m(N) := n^{3k}$. The map G_N views its input string x as a pair (M, r), where M is the description of a clocked deterministic machine running in time $10k \cdot \log n$, and r is the rest part of x, treated as a random string. We assume that this encoding satisfies that for a random x, every machine M of description length ℓ occurs with probability $\Theta(2^{-\ell}/\ell^2)$ (this is possible since $\sum_{\ell \geq 1} \frac{1}{\ell^2}$ is bounded). The important part is that if the description length of M is a constant, then it occurs with constant probability. Let $D_M \colon \{0,1\}^{n^{2k}} \times \{0,1\}^n \to \{0,1\}$ be the $\mathcal{C}$-circuit of size at most n^{2k} encoded by the machine $M(1^n, \cdot)$ (i.e., we assume that M computes the direct connection language of D_M). For $i \in \{0,1\}^n$, we define the i-th output bit of $G_N(x)$ as $D_M(r, i)$. Note that a uniform computation over inputs of length at most $10k \cdot \log n$ and running in time $10k \cdot \log n$ can be uniformly converted into an AC^0 circuit of size at most n^{10k}. Since $\mathcal{C}$ contains AC^0, admits universal circuits, and is closed under composition, G_N can be implicitly computed by a $\mathsf{DLOGTIME}$-uniform probabilistic $\mathcal{C}$-circuit C_N defined over $m(N) + n$ input bits and of size at most $n^{C \cdot k}$, where C is a large enough universal constant that depends only on the circuit class $\mathcal{C}$.

Let $B(1^N)$ be an algorithm of space complexity $O(\log N)$ that solves $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on infinitely many values of N for the sequence G_N, with parameters $c, d \leq C \cdot k$ and function $\delta(N) \leq o(1)$. Let L_B be the language defined by B, i.e., a string $z \in \{0,1\}^n$ is in L_B if and only if the z-th bit of $B(1^N)$ (with $N = 2^n$) is 1. Note that L_B is in $\mathsf{DSPACE}[O(n)]$.

We now argue that L_B cannot be computed by $\mathsf{DTIME}[k \cdot \log n]$-uniform randomized $\mathcal{C}$-circuits of size n^k. To prove this, it is enough to show that for every language L computed by such circuits, each string in the sequence $\{y_N^L\}_N$ of truth-tables obtained from L is δ-heavy in $G_N(\mathcal{U}_{m(N)})$ for every large enough N. Under this claim, since B solves $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` for the sequence $\{G_N\}$, it follows that $L_B \neq L$.

To see that the claim holds, recall that L is computed by $\mathsf{DTIME}[k \cdot \log n]$-uniform randomized $\mathcal{C}$-circuits of size n^k. Consequently, there is a deterministic machine M_L that runs in time $k \cdot \log n$ and decides the direct connection language of a corresponding randomized $\mathcal{C}$-circuit D_L of size at most n^{2k} and using at most n^k random bits that computes L on n-bit inputs. We now boost the success probability of the circuit D_L via repetition and (approximate) majority vote. More precisely, since the approximate majority function can be computed by $\mathsf{DLOGTIME}$-uniform AC^0 circuits [Ajt90, Vio09], $\mathcal{C}$ contains AC^0, and $\mathcal{C}$ is closed under composition, there is a deterministic machine $\widetilde{M}_L$ that runs in time $10k \cdot \log n$ and decides the direct connection language of a corresponding randomized $\mathcal{C}$-circuit $\widetilde{D}_L$ of size at most $n^{C \cdot k}$ and using at most n^{2k} random bits that computes

[18] If $\mathcal{C}$ is a constant-depth circuit class defined as a union of classes for each fixed depth k, the argument can be adapted accordingly.

L on each n-bit input string with probability at least $1 - 2^{-2n}$. Moreover, we can assume that the description length of $\widetilde{M}_L$ is a constant ℓ_{desc}, hence it occurs with constant probability. Let y_N^L be the truth-table of L on input length n, i.e., $|y_N^L| = N = 2^n$. By construction, using an union bound over all n-bit input strings, the probability that $G_N(\mathcal{U}_{m(N)}) = y_N^L$ is at least $\Omega(2^{-\ell_{\mathsf{desc}}}/\ell_{\mathsf{desc}}^2) \cdot (1 - 2^{-n}) \geq \Omega(1) > \delta$, for large enough N. This shows that y_N^L is δ-heavy, concluding the proof of this item.

$(i) \Rightarrow (ii)$. We argue in the contrapositive. In other words, suppose that there is a choice of constants c, d, and ℓ, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and a sequence $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size n^c such that every algorithm $A(1^N)$ running in space $O(\log N)$ time fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on every large enough input length N. Next, we use this assumption to establish that $\mathsf{PSPACE} \subseteq \mathsf{BP}\text{-}\mathcal{C}$, which concludes the proof.

We consider a candidate algorithm $A(1^N)$ that computes as follows. Consider the language $L^\star$ from Theorem 3.1, and assume that $L^\star \in \mathsf{DSPACE}[n^a]$, where $a \in \mathbb{N}$. For a given $N' = 2^{n'}$, we let $\mathsf{tt}^\star_{N'} \in \{0,1\}^{N'}$ denote the truth table of $L^\star$ over inputs of length n'. On input 1^N, algorithm A outputs the string $y_N = \mathsf{tt}^\star_{N'} 0^{u_N} \in \{0,1\}^N$, where $N' = 2^{n'}$ for $n' = n^{1/a}$, and $u_N = N - N'$.

Note that A computes in space $O(\log N)$, due to our choice of parameters. Therefore, A fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on every large enough input length N. This means that for every large enough N the probability of y_N under $G_N(\mathcal{U}_{m(N)})$ is at least $\delta = 1/n^\ell$.

We first describe a randomized algorithm that computes $L^\star$, deferring for now a discussion of its correctness, circuit complexity, and uniformity. Let $n = (n')^a$, as above. To compute $L^\star$ on a given input x of length $n' \in \mathbb{N}$, we sample $v = n^{3\ell}$ strings $z_1, \ldots, z_v \in \{0,1\}^{m(N)}$ uniformly and independently at random, and use the $2^{n'}$-bit prefixes $\mathcal{O}'_1, \ldots, \mathcal{O}'_v$ of the corresponding oracles $\mathcal{O}_1, \ldots, \mathcal{O}_v$ as candidate oracles for $L^\star$ on input length n', where each $\mathcal{O}_i$ is the oracle associated with the string $G_N(z_i) \in \{0,1\}^N$. In more detail, let $b_i = \mathsf{IC}_{n'}^{\mathcal{O}'_i}(x)$, where $\mathsf{IC}_{n'}$ is the algorithm from Theorem 3.1. We output 1 if at least one bit among $b_1, \ldots, b_v$ is 1, and 0 otherwise.

Next, we argue that A computes $L^\star$ with high probability. Consider an arbitrary input length n' and a given input string $x \in \{0,1\}^{n'}$. By our choice of v, with high probability the string y_N appears among the strings $G_N(z_1), \ldots, G_N(z_v)$. In particular, with high probability the truth table $\mathsf{tt}^\star_{N'}$ appears as an N'-bit prefix of one of these strings, meaning that one of the oracles $\mathcal{O}'_i$ computes $L^\star$ on inputs of length n'. Consequently, in this case, if $L^\star(x) = 1$ then at least one bit $b_i = 1$, and the procedure outputs 1. On the other hand, if $L^\star(x) = 0$, then by a union bound over the internal randomness of $\mathsf{IC}_{n'}$, with high probability every bit $b_i \in \{0, \bot\}$. In this case, the procedure outputs 0. This establishes the correctness of A.

It remains to establish an upper bound on the circuit complexity of A and to analyze the uniformity of the corresponding circuits. Note that each bit $b_i \in \{0, 1, \bot\}$ can be computed by a randomized $\mathcal{C}$-circuit of polynomial size, since G_N is implicitly computed by $\mathcal{C}$-circuits of polynomial size, $\mathsf{IC}_{n'}$ is computable by randomized $\mathcal{C}$-circuits of polynomial size, and $\mathcal{C}$ is closed under composition. Moreover, the disjunction of the bits b_i can also be computed in $\mathcal{C}$, since this class contains $\mathsf{AC}^0[\oplus]$. Therefore, A can be implemented by randomized $\mathcal{C}$-circuits of polynomial size. Finally, it is not hard to check that the corresponding sequence of randomized $\mathcal{C}$-circuits is $\mathsf{DLOGTIME}$ uniform, since IC is computed by $\mathsf{DLOGTIME}$-uniform randomized circuits, and G_N is implicitly computed by $\mathsf{DLOGTIME}$-uniform circuits.

The above discussion implies that $L^\star \in \mathsf{BP}\text{-}\mathcal{C}$. Since $L^\star$ is complete under $\mathsf{DLOGTIME}$-uniform projection reductions, we get that $\mathsf{PSPACE} \subseteq \mathsf{BP}\text{-}\mathcal{C}$, as desired. $\square$

Our characterizations also extend to *almost-everywhere* lower bounds and *sub-exponential* lower bounds, as demonstrated in the following theorems.

Theorem 3.3. *Let $\mathcal{C}$ be a nice class of Boolean circuits. The following statements are equivalent:*

(*i*) $\mathsf{PSPACE} \not\subseteq \mathsf{i.o.BP}\text{-}\mathcal{C}$.

(*ii*) *For every choice of $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and for every sequence $\{G_N\}$ of maps $G_N\colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size at most n^c, the corresponding $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem can be solved in $O(\log(N))$ space* for all large enough N.

Proof Sketch. The result follows from the same argument given for Theorem 3.2:

- $(ii) \Rightarrow (i)$: If our algorithm B is correct on input 1^N, then our language L is hard on input length n.
- $(i) \Rightarrow (ii)$: If the language $L^\star$ is hard on input length n', then our algorithm A is correct on input 1^N where $N = 2^{(n')^a}$. □

We use $\mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[f(n)]$ to denote the class of languages computable by $\mathsf{DLOGTIME}$-uniform $\mathsf{BP}\text{-}\mathcal{C}$ circuits of size $f(n)$.

Theorem 3.4. *Let $\mathcal{C}$ be a nice class of Boolean circuits. The following statements are equivalent:*

(*i*) *There is a constant $\epsilon > 0$ such that $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{n^\epsilon}]$.*

(*ii*) *There is a constant $\epsilon > 0$ such that for $\delta(N) := 2^{-n^\epsilon}$ and for every sequence $\{G_N\}$ of maps $G_N\colon \{0,1\}^{2^{n^\epsilon}} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size 2^{n^ϵ}, the corresponding $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem can be solved in $\mathsf{poly}(N)$ time on infinitely many input lengths N.*

Proof Sketch. The argument is an adaptation of the proof of Theorem 3.2 by adjusting a few parameters, so we refer the reader to that proof for more details.

To see that $(ii) \Rightarrow (i)$ holds, let $\epsilon' := \epsilon/4$, and consider the map $G_N\colon \{0,1\}^{m(N)} \to \{0,1\}^N$ where $m(N) := 2^{n^{2\epsilon'}}$ and the input bits are parsed into the description of a Turing machine M that encodes a size-$2^{n^{\epsilon'}}$ $\mathcal{C}$-circuit D_M and the rest random inputs (fed to D_M). Like in Theorem 3.2, we assume that every constant-size Turing machine occurs with constant probability. This map G_N can be implicitly computed by a $\mathsf{DLOGTIME}$-uniform probabilistic $\mathcal{C}$-circuit C_N of size 2^{n^ϵ}, in the sense that for every $x \in \{0,1\}^{m(N)}$ and $i \in [N]$, the i-th bit of $G_N(x)$ is equal to $C_N(x,i)$. We can see that for every language $L \in \mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{n^{\epsilon'}}]$, let y_N^L denote the truth table of L_n, then the probability that $G_N(\mathcal{U}_{m(N)}) = y_N^L$ is at least a constant. Hence, given an algorithm $B(1^N)$ that solves the $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem for $\delta = o(1)$ (on infinitely many N), the language whose truth table is the output of $B(1^N)$ is not in $\mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{n^{\epsilon'}}]$ (on infinitely many n). Since we further assumed that B runs in space $O(\log N)$, we obtain a hard language in $\mathsf{SPACE}[O(\log N)] = \mathsf{SPACE}[O(n)]$ that is not in $\mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{n^{\epsilon'}}]$.

To see that $(i) \Rightarrow (ii)$ holds, let $L^\star$ denote the PSPACE-complete language in Theorem 3.1 and let $\epsilon' := \epsilon/(5a)$, where $a \geq 1$ is a constant such that $L^\star \in \mathsf{SPACE}[n^a]$. Consider the following algorithm $A(1^N)$ for solving the $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem with parameter ϵ'. On input 1^N, let $n' := n^{1/a}$, $N' := 2^{n'}$, $y_{n'} \in \{0,1\}^{N'}$ be the truth table of $L^\star$ on input length n', then A outputs $y_{n'}0^{N-N'} \in \{0,1\}^N$. If A fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid`, then we can compute $L^\star$ in $\mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{n^\epsilon}]$ as follows. Let $x \in \{0,1\}^{n'}$ be an instance of $L^\star$, we set $n := (n')^a$ and $N' := 2^{n'}$. We sample $v := (1/\delta(N))^3 \leq 2^{3n^\epsilon}$ strings $z_1, \dots, z_v \in \{0,1\}^{2^{n^\epsilon}}$ uniformly and independently at random, and for each string z_i we define an oracle $\mathcal{O}_i \colon \{0,1\}^{n'} \to \{0,1\}$ whose truth table is the first N' bits of $G_N(z_i)$. We then run IC against each $\mathcal{O}_i$ and obtain $b_i := \mathsf{IC}_{n'}^{\mathcal{O}_i}(x)$ for each $i \in [v]$, and finally we output 1 if some b_i is equal to 1. This algorithm computes $L^\star$ because with high probability, the truth table $y_{n'}$ appears in these oracles, and also the instance checker will never output $1 - L(x)$ by mistake. Our algorithm can be implemented in $\mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{n^{5\epsilon'}}] \subseteq \mathsf{BP}\text{-}\mathcal{C}\text{-}\mathsf{SIZE}[2^{(n')^\epsilon}]$. □

Remark 3.5. In the proof of Theorem 3.2, we only need to solve the $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem for $\delta = o(1)$ to obtain the lower bound (i.e., Item (i)), while the latter implies algorithms for the $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem even when $\delta = 1/\mathsf{poly}(n) = 1/\mathsf{polylog}(N)$. This illustrates the *robustness* of the parameter δ in $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` with respect to $O(\log N)$-space algorithms: if the problem is solvable for $\delta = o(1)$, then it is also solvable for $\delta = 1/\mathsf{polylog}(N)$. Similarly, Theorem 3.4 shows that if we consider implicit maps computable in the 2^{n^ϵ} time regime, then this problem is solvable for $\delta = o(1)$ if and only if it is solvable for $\delta = 2^{-n^\epsilon}$. In fact, it is evident from the proofs that the robustness of the parameter δ holds in every characterization result in Section 3.

3.2 Equivalences for NP via Search-to-Decision Reductions

In this section, we show equivalences between uniform randomized lower bounds for NP and Heavy Avoid algorithms implementable by constant-depth circuits. NP is not known to be instance-checkable and so we cannot use the technique from the previous section. However, it turns out that search-to-decision reductions can also be used to argue the desired equivalences. The standard search-to-decision reduction is highly sequential, so in order to show equivalences that work for any nice circuit class, we use a depth-efficient version based on the Valiant-Vazirani Isolation Lemma [VV86] which exploits our access to randomness.

We first need a generalization of the standard result that $\mathsf{NP} \not\subseteq \mathsf{BPP}$ iff $\mathsf{PH} \not\subseteq \mathsf{BPP}$.

Lemma 3.6. *Let $\mathcal{C}$ be a nice circuit class.* $\mathsf{NP} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ *if and only if* $\mathsf{PH} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$.

Proof Sketch. The proof is essentially the same inductive argument as for the standard equivalence between $\mathsf{NP} \not\subseteq \mathsf{BPP}$ and $\mathsf{PH} \not\subseteq \mathsf{BPP}$. We must show that if $\mathsf{NP} \subseteq \mathsf{BPP}$ then $\mathsf{PH} \subseteq \mathsf{BPP}$. In order to implement that argument, we need to be able to do error reduction to exponentially small error by $\mathsf{DLOGTIME}$-uniform randomized $\mathcal{C}$ circuits, which holds since $\mathcal{C}$ contains AC^0, and Approximate Majority can be computed in $\mathsf{DLOGTIME}$-uniform AC^0 [Ajt90, Vio09]. We also need the closure of $\mathcal{C}$ under composition to be able to do induction, but that also holds by niceness of $\mathcal{C}$. □

Now we proceed to our equivalences for NP.

Theorem 3.7 (Equivalences for NP). *Let $\mathcal{C}$ be a nice circuit class. The following statements are equivalent:*

(i) $\mathsf{NP} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$.

(ii) *There are positive integers k and r such that for every $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and for every sequence $\{G_N\}$ of maps $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by* $\mathsf{DLOGTIME}$*-uniform $\mathcal{C}$-circuits of size at most n^c, the corresponding $\mathcal{C}$-*`Implicit`*-δ-*`Heavy-Avoid` *problem can be solved by* $\mathsf{DLOGTIME}$*-uniform unbounded fan-in circuits of size $2^{\log(N)^r}$ and depth k on infinitely many input lengths N.*

Proof. We consider each implication below.

$(ii) \Rightarrow (i)$. We will use the assumption to show that $\mathsf{PH} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$, and the desired implication then follows from Lemma 3.6 and the assumption that $\mathcal{C}$ is nice.

As in the proof of the analogous equivalence for PSPACE, fix a large enough $a \geq 1$, and consider the map $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ defined as follows, where $m(N) := n^{3a}$. The map G_N parses its input string x into (M, r), where M is the description of a clocked deterministic machine running in time $10a \cdot \log n$, and r consists of the remaining bits of x, treated as randomness. Let $D_M \colon \{0,1\}^{n^{2a}} \times \{0,1\}^n \to \{0,1\}$ be the randomized $\mathcal{C}$ circuit of size at most n^{2a} encoded by the machine $M(1^n, \cdot)$ (i.e., we assume that M computes the direct connection language of D_M). For $i \in \{0,1\}^n$, we define the i-th output bit of $G_N(x)$ as $D_M(r, i)$. Note that G_N can be implicitly computed by a $\mathsf{DLOGTIME}$-uniform randomized $\mathcal{C}$ circuit C_N defined over $m(N)+n$ input bits and of size at most $n^{C \cdot a}$, where C is a large enough universal constant.

As per assumption, let $\{C_N\}$ be a $\mathsf{DLOGTIME}$-uniform family of unbounded fan-in circuits of size $2^{\log(N)^r}$ and depth k such that for infinitely many N, C_N solves $\mathcal{C}$-$\texttt{Implicit}$-δ-$\texttt{Heavy-Avoid}$ on G_N, with parameters $c = C \cdot a$, $d = 2a+1$, and $\delta(N) = o(1)$. (Recall that each output bit of G_N is computed in time n^c, $m(N) \leq n^d$, and we want to find a $\delta(N)$-light element.) Let L be the language defined by $\{C_N\}$, i.e., a string $z \in \{0,1\}^n$ is in L if and only if the z-th bit of $C_N(1^N)$ (with $N = 2^n$) is 1. Note that there are integers s and k' (depending only on r and k) such that L is in $\Sigma_{k'}$-$\mathsf{TIME}[n^s]$ by the known equivalence [BIS90] between PH and $\mathsf{DLOGTIME}$-uniform circuits of exponential size in n (which is quasi-polynomial size in N).

We now argue that L cannot be computed by $\mathsf{DTIME}[a \cdot \log n]$-uniform randomized $\mathcal{C}$ circuits of size n^a. To prove this, it is enough to show that for every language L' computed by such circuits, each string in the sequence $\{y_N^{L'}\}_N$ of truth-tables obtained from L' is δ-heavy in $G_N(\mathcal{U}_{m(N)})$ for every large enough N. Under this claim, since B solves $\mathcal{C}$-$\texttt{Implicit}$-δ-$\texttt{Heavy-Avoid}$ for the sequence $\{G_N\}$, it follows that $L \neq L'$.

To see that the claim holds, suppose that L' is computed by $\mathsf{DTIME}[a \cdot \log n]$-uniform randomized $\mathcal{C}$ circuits $D_{L'}$ of size n^a. Consequently, there is a deterministic machine $M_{L'}$ that runs in time $a \cdot \log n$ and decides the direct connection language of a corresponding randomized $\mathcal{C}$ circuit $D_{L'}$, and the circuit $D_{L'}$ computes L' on n-bit inputs, has size at most n^{2a} and uses at most n^a random bits. We can then reduce the error of the circuit D_L to be exponentially small by using the facts that Approximate Majority is in $\mathsf{DLOGTIME}$-uniform AC^0 [Ajt90, Vio09] and that $\mathcal{C}$ is nice. Thus we obtain a family of randomized Boolean circuits $\widetilde{D}_L$ that has size at most $n^{C \cdot a}$, uses at most n^{2a} random bits, and computes L on each n-bit input string with probability at least $1 - 2^{-2n}$. Moreover, there is a deterministic machine $\widetilde{N}_{L'}$ that runs in time $10a \cdot \log n$ and decides the direct connection language of $\widetilde{D}_L$. Since the description length of $\widetilde{N}_{L'}$ is constant, it occurs with constant probability in the distribution sampled by G_N. Let $y_N^{L'}$ be the truth-table of L' on input length n, i.e., $|y_N^{L'}| = N = 2^n$. By construction, using an union bound over all n-bit input strings, the probability that $G_N(\mathcal{U}_{m(N)}) = y_N^{L'}$ is at least $\Omega(1) \cdot (1 - 2^{-n}) \geq \delta(N)$. This shows that $y_N^{L'}$ is δ-heavy, concluding the proof of the claim.

Note that the language L defined above depends on a, however by the standard fact that there is a language L_{comp} complete for $\Sigma_{k'}$-$\mathsf{TIME}[n^s]$ under $\mathsf{DLOGTIME}$-uniform projections of linear size, we get that that for each a, L_{comp} is not computed by $\mathsf{DTIME}[a \cdot \log n]$-uniform randomized $\mathcal{C}$ circuits of size n^a. This implies that $L_{\text{comp}} \notin \mathsf{BP}\text{-}\mathcal{C}$, and hence that $\mathsf{PH} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$. Therefore $\mathsf{NP} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ by Lemma 3.6, concluding the proof of this item.

$(i) \Rightarrow (ii)$. We argue in the contrapositive. Suppose that there is a choice of constants c,d, and ℓ, with $m(N) = n^d$ and $\delta(N) = 1/n^{\ell}$, and a sequence $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$ circuits of size n^c such that every $\mathsf{DLOGTIME}$-uniform sequence of unbounded fan-in circuits of size $2^{\log(N)^2}$ and depth 3 fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on every large enough input length N. We use this assumption to establish that $\mathsf{NP} \subseteq \mathsf{BP}\text{-}\mathcal{C}$, which concludes the proof.

We consider a candidate algorithm $A(1^N)$ that simply outputs tt_N, where tt_N is the truth table of SAT on n-bit inputs. We consider a standard encoding of SAT in which SAT is depth-efficiently paddable, i.e., there is an algorithm Pad implemented by $\mathsf{DLOGTIME}$-uniform AC^0 circuits which, given as inputs 1^t for positive integer t and a formula ϕ of length at most t, outputs an equisatisfiable formula ϕ' of length t. Note that A can be implemented by $\mathsf{DLOGTIME}$-uniform unbounded fan-in circuits of size $2^{\log(N)^2}$ and depth 3, using the known simulation of non-deterministic quasi-linear time by uniform unbounded fan-in circuits [BIS90]. By assumption, A fails to solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` on every large enough input length N. We show how to use this failure together with a depth-efficient randomized search-to-decision reduction based on the Valiant-Vazirani Isolation Lemma [VV86] and depth-efficient paddability of SAT to solve SAT in $\mathsf{BP}\text{-}\mathcal{C}$, which implies $\mathsf{NP} \subseteq \mathsf{BP}\text{-}\mathcal{C}$ by the NP-completeness of SAT with respect to $\mathsf{DLOGTIME}$-uniform projections.

By the failure of A, we have that for every large enough N, the probability of tt_N under $G_N(\mathcal{U}_{m(N)})$ is at least $\delta = 1/n^{\ell}$. First, we describe a polynomial-time randomized algorithm B to solve SAT. Let ϕ be a length-n input to SAT. For some $T = \mathsf{quasipoly}(N)$ and $t = \log T$ to be determined later, we sample $v := t^{5\ell}$ strings $z_1, \ldots, z_v \in \{0,1\}^{m(T)}$ uniformly and independently at random, and for each string z_i, we define an oracle $\mathcal{O}_i$ whose truth table is the string $G_T(z_i) \in \{0,1\}^T$. For each $i \leq v$, we try to use $\mathcal{O}_i$ and a depth-efficient search-to-decision reduction to find a satisfying assignment to ϕ, as follows. Assume without loss of generality that ϕ has n variables. We do the following for each i in parallel. We check if $\mathcal{O}_i(\mathsf{pad}(1^t, \phi)) = 1$. If this is not the case for any i, we reject. If $\mathcal{O}_i$ does evaluate to 1 on the padded version of ϕ, we use this oracle to find a candidate satisfying assignment w to ϕ as follows. The idea is to use the Valiant-Vazirani technique of intersecting the solution space of ϕ with k randomly chosen hyperplanes for $k = 1 \ldots n$ to obtain formulas $\phi_1, \ldots, \phi_n$. The Valiant-Vazirani Isolation Lemma [VV86] states that if ϕ is satisfiable, then with probability at least $1/4n$ over random choices of these formulas, some ϕ_j has a unique solution. Note that each ϕ_j can be constructed from ϕ by randomized constant-depth circuits. We would like to use O_i to find and check the unique satisfying assignment so that we can verify that ϕ is satisfiable. An issue is that the ϕ_j are in general of size larger than n, but they are still of size $\mathsf{poly}(n)$ and we choose t a large enough polynomial in n so that they can all be padded to length t. For each ϕ_j and each of the n original variables x_k in ϕ, we use an oracle call to O_i (using padding if necessary) to determine if there is a satisfying assignment to ϕ with the variable x_k set to 0. If yes, we set the wire $b_{j,k}$ to 1, else to 0. We check if there is a j such that the assignment $x_k = b_{j,k}$ for each k satisfies ϕ. If this is the case for some i, we accept, otherwise we reject. Note that all of the above can be implemented in constant-depth, apart form the oracle calls to O_i, which we simulate by evaluations of the implicit sampler for O_i.

By the niceness of $\mathcal{C}$, specifically the assumptions that $\mathsf{AC}^0[\oplus]$ is contained in $\mathcal{C}$ and $\mathcal{C}$ is closed under composition, as well as the fact that our sampler is implicitly computed by uniform $\mathcal{C}$ circuits, there are $\mathsf{DLOGTIME}$-uniform randomized $\mathcal{C}$ circuits of polynomial size implementing the procedure above. We need to argue that these circuits correctly solve SAT with high probability. Note that a circuit only accepts a formula ϕ if it verifies that some assignment satisfies ϕ. Hence it is enough to check that satisfiable ϕ is accepted with high probability. By our choice of v, with high probability the string tt_T appears with multiplicity $\omega(n)$ among the strings $G_T(z_1), \ldots, G_T(z_v)$, meaning that $\omega(n)$ of the oracles $\mathcal{O}_i$ compute SAT on inputs of length t. By the correctness of the paddability procedure, for all of these correct oracles, satisfiability questions about the randomized formulas

ϕ_j are all answered correctly. This together with the lower bound on probability that one of the ϕ_j is uniquely satisfiable implies that with probability at least $1 - o(1)$, oracle calls to some oracle $\mathcal{O}_i$ yield a satisfying assignment for ϕ, which is then correctly verified and results in acceptance of the circuit. $\square$

3.3 Equivalences for EXP and $\mathsf{EXP}^{\mathsf{NP}}$ via a Win-Win Argument and Selectors

In this section, we generalize our equivalence results to the classes EXP and $\mathsf{EXP}^{\mathsf{NP}}$.

We provide two proofs. The first proof uses a win-win argument and relies on the existing equivalence result for PSPACE (Theorem 3.2). The second proof uses *selectors* for $\mathsf{EXP}^{\mathsf{NP}}$-complete languages [Hir15] or *instance checkers* for EXP-complete languages [BFL91]. Each proof has its advantages and disadvantages, as will be discussed in Remark 3.16.

We start with the first proof. We first state the equivalence result for $\mathsf{EXP}^{\mathsf{NP}}$.

Theorem 3.8 (Equivalence for $\mathsf{EXP}^{\mathsf{NP}}$). *Let $\mathcal{C}$ be a nice class of Boolean circuits. The following statements are equivalent:*

(i) $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$.

(ii) *For every choice of $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and for every sequence $\{G_N\}$ of maps $G_N\colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size at most n^c, the corresponding $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem can be solved in deterministic time $\mathsf{poly}(N)$ with access to an NP oracle on infinitely many input lengths N.*

Proof. We consider each implication below.

$(ii) \Rightarrow (i)$. The proof is completely analogous to the implication from (ii) to (i) in Theorem 3.2. The only difference is that due to the access to an NP oracle provided to each deterministic polynomial-time algorithm for `Implicit`-δ-`Heavy-Avoid`, the resulting hard language is in $\mathsf{EXP}^{\mathsf{NP}}$ as opposed to PSPACE.

$(i) \Rightarrow (ii)$. If $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$ then either $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{PSPACE}$ or $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$. We show that the desired conclusion holds in each one of these cases.

First, assume that $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{PSPACE}$. Recall that if $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{SIZE}[\mathsf{poly}]$ then $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{PSPACE}$ [BH92]. Therefore, it follows that $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}[\mathsf{poly}]$. In particular, there is a language $L \in \mathsf{DTIME}[2^{O(n)}]^{\mathsf{NP}}$ such that $L \notin \mathsf{SIZE}[n^k]$ for every choice of k. Now fix a choice of $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and consider a sequence $\{G_N\}$ of maps $G_N\colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size at most n^c. Consider the following algorithm A: On input 1^N, where $N = 2^n$, A outputs the N-bit string w_N corresponding to the truth-table of L on n-bit strings. Since $L \in \mathsf{DTIME}[2^{O(n)}]^{\mathsf{NP}}$, A computes in time $\mathsf{poly}(N)$ with access to an NP oracle. Moreover, since L is not computed by (general) Boolean circuits of size n^k on infinitely many input lengths, where k is an arbitrary constant, it follows that w_N is not in the range of G_N for infinitely many values of N. Otherwise, there would be a choice a for the seed of G_N such that $w_N = G_N(a)$, which yields a bounded size circuit for the function encoded by w_N, given that G_N is implicitly computed by $\mathcal{C}$-circuits of bounded size. In particular, it follows that on infinitely many values of N, w_N is not δ-heavy for G_N, as desired.

Now consider the remaining case, i.e., assume that $\mathsf{PSPACE} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$. Then, by Theorem 3.2, we can solve the required $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem in space $O(\log N)$ on infinitely many input lengths N. Since $O(\log N)$ space algorithms can be simulated by $\mathsf{poly}(N)$ time algorithm (not to mention that we have access to an NP oracle), the desired conclusion also holds in this case. $\square$

It is easy to see that a similar equivalence result for EXP also holds. Actually, the proof is essentially the same as Theorem 3.8, with the only difference being that we use the Karp–Lipton theorem for EXP [KL80] instead of the one for $\mathsf{EXP}^{\mathsf{NP}}$ [BH92]. Hence we only state the result and omit the proof here.

Theorem 3.9 (Equivalence for EXP). *Let $\mathcal{C}$ be a nice class of Boolean circuits. The following statements are equivalent:*

(*i*) $\mathsf{EXP} \not\subseteq \mathsf{BP}\text{-}\mathcal{C}$.

(*ii*) *For every choice of $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and for every sequence $\{G_N\}$ of maps $G_N\colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-circuits of size at most n^c, the corresponding $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem can be solved in deterministic time $\mathsf{poly}(N)$ on infinitely many input lengths N.*

Using Theorem 3.9, it is not hard to prove Corollary 1.3 stated in Section 1.1.1. We restate the result below for convenience.

Corollary 3.10. *For every function $e(N) = \omega(1)$, there is a unary BPP search problem S such that:*[19]

- *$S(1^N)$ can be solved in randomized $\mathsf{poly}(N)$ time by an algorithm that uses N random bits and errs on at most $e(N)$ random bit sequences;*
- *If there exists a randomized polynomial-time algorithm for S with non-zero success probability that errs on at most $O(1)$ random bit sequences for any input, then $\mathsf{EXP} \neq \mathsf{BPP}$.*

Proof Sketch. To argue this, we can assume that $\mathsf{EXP} \subseteq \mathsf{BPP}$, since the claim is trivial otherwise. Then, by Theorem 3.9 with $\mathcal{C}$ = "general Boolean circuits" and $\mathsf{BP}\text{-}\mathcal{C} = \mathsf{BPP}$, and using Remark 3.5 on the robustness of δ, there is a uniform polynomial-time sampler and a corresponding distribution $\mathcal{D}$ for which $(\mathcal{D}, 1/e(N))$-`Heavy-Avoid` does not admit infinitely-often correct deterministic polynomial-time algorithms. Now note that the corresponding Heavy Avoid problem for $\mathcal{D}$ is a unary BPP search problem $S(1^N)$ that can be solved in randomized $\mathsf{poly}(N)$ time by an algorithm that uses N random bits and errs on at most $e(N)$ random bit sequences (indeed, a random string of length N satisfies this guarantee). On the other hand, if there is a randomized polynomial-time algorithm for S with non-zero success probability that only errs on constantly many random bit sequences for any input, then we argue that there is a *deterministic* polynomial-time algorithm for S that succeeds on infinitely many input lengths, which contradicts our assumption. Suppose k is a constant such that there is some randomized polynomial-time algorithm A for S which errs on at most k random bit sequences for any input 1^N. We define $k+1$ deterministic polynomial-time algorithms $A_1 \ldots A_{k+1}$ such that for some $i \in [k+1]$, A_i solves S on infinitely many input lengths. On input 1^N, A_i runs A with random bit sequence r_i, where r_i is the i-th random bit sequence used by A on input length N in lexicographical order. Note that since A only makes errors on at most k random bit sequences for any N and has non-zero success probability, for each N there is an i such that A_i outputs a correct solution to S on 1^N. Since this is the case for each N, there

[19]For the definition of BPP search problems and related discussions, see, e.g., [Gol11].

is some i such that A_i outputs a correct solution to S on 1^N for infinitely many N, and our result follows.[20] $\square$

Next, we present another proof of Theorem 3.8 for the special case where $\mathcal{C}$ is the class of general Boolean circuits. The proof is direct and does not go through win-win arguments. In one sentence, we use *selectors* for $\mathsf{EXP}^{\mathsf{NP}}$-complete problems [Hir15] and note that the proof strategy from Theorem 3.2 extends to selectors.

Definition 3.11. We say that a probabilistic polynomial-time oracle machine S is a *selector* for a language $L \subseteq \{0,1\}^*$ if the following holds. Let $\mathcal{O}_1, \mathcal{O}_2 \subseteq \{0,1\}^*$ be arbitrary oracles. Then, for any input $x \in \{0,1\}^*$, if $L \in \{\mathcal{O}_1, \mathcal{O}_2\}$ then

$$\Pr_S[S^{\mathcal{O}_1,\mathcal{O}_2}(x) = L(x)] \geq 2/3.$$

It is possible to boost the success probability of the selector using standard techniques. In addition, [Hir15] proved that if a language L admits a selector then it also admits a selector that succeeds when given access to polynomially many oracles provided that at least one of them correctly computes L. These are summarized in the following result.

Theorem 3.12 ([Hir15])**.** *Every $\mathsf{EXP}^{\mathsf{NP}}$-complete language admits a selector. Moreover, there is a paddable $\mathsf{EXP}^{\mathsf{NP}}$-complete language $L' \in \mathsf{DTIME}[2^{O(n)}]^{\mathsf{NP}}$, a polynomial q, and a probabilistic polynomial-time oracle algorithm S such that the following conditions hold:*

- *For every $n \geq 1$, $x \in \{0,1\}^n$, and $t \geq 1$, if $\mathcal{O}_1, \ldots, \mathcal{O}_t \subseteq \{0,1\}^*$ and $L' \in \{\mathcal{O}_1, \ldots, \mathcal{O}_t\}$, then*

$$\Pr_S[S^{\mathcal{O}_1,\ldots,\mathcal{O}_t}(x, 1^t) = L'(x)] \geq 1 - 2^{-n}.$$

- *Every oracle query of $S(x)$ has length exactly $q(n)$. Consequently, it is enough to assume that the oracles $\mathcal{O}_1, \ldots, \mathcal{O}_t \colon \{0,1\}^{q(n)} \to \{0,1\}$ and that $L'_{q(n)} \in \{\mathcal{O}_1, \ldots, \mathcal{O}_t\}$, where $L'_{q(n)} = L \cap \{0,1\}^{q(n)}$.*

The "moreover" part of the result follows from existence of a selector for every $\mathsf{EXP}^{\mathsf{NP}}$-complete problem [Hir15] combined with the paddability of L' and the discussion above.

Proof of Theorem 3.8, for the case that $\mathcal{C}$ is the class of general Boolean circuits.

$(ii) \Rightarrow (i)$. Again, the proof is completely analogous to the same implication in Theorem 3.2, and we omit the details.

$(i) \Rightarrow (ii)$. We argue in the contrapositive. Suppose there is a choice of constants c, d, and ℓ, with $m(N) = n^d$ and $\delta(N) = 1/n^{\ell}$, and a sequence $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed by general Boolean circuits of size at most n^c such that every deterministic algorithm $A(1^N)$ with access to an NP oracle that runs in $\mathsf{poly}(N)$ time fails to solve `Implicit-`δ`-Heavy-Avoid` on every large enough input length N. Next, we use this assumption to establish that $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{BPP}$.

We consider a candidate algorithm $A'(1^N)$ with access to an NP oracle that computes as follows. Consider the $\mathsf{EXP}^{\mathsf{NP}}$-complete language L' from Theorem 3.12. On input 1^N, A' outputs the truth-table $tt_N \in \{0,1\}^N$ of L' over strings of length n, where $N = 2^n$. Note that A' uses an

[20]While the argument presented here is somewhat non-constructive, we remark that it is possible to modify the proof so that for each $k \geq 1$, there is an explicit unary BPP search problem satisfying the same conditions except that one obtains the weaker conclusion that $\mathsf{DTIME}[2^n] \not\subseteq \mathsf{BPTIME}[n^k]$ in the second bullet. We omit the details.

NP oracle and computes in time $\mathsf{poly}(N)$, since $L' \in \mathsf{DTIME}[2^{O(n)}]^{\mathsf{NP}}$. Since A' fails to solve `Implicit-`δ`-Heavy-Avoid`, for every large enough N, the probability of tt_N under $G_N(\mathcal{U}_{m(N)})$ is at least $\delta = 1/n^{\ell}$.

The rest of the argument is similar to that of the proof of Theorem 3.2. To compute L' on some input $x \in \{0,1\}^{n'}$, we let $n := q(n')$, where q is the polynomial from Theorem 3.12. We sample $t := n^{3\ell}$ strings $z_1, \ldots, z_t \in \{0,1\}^{m(N)}$ uniformly and independently at random, and use the corresponding oracles $\mathcal{O}_1, \ldots, \mathcal{O}_t$ as candidate oracles for L'_n, where each $\mathcal{O}_i$ computes according to the string $G_N(z_i) \in \{0,1\}^N$. By our choice of t, with high probability the string tt_N is among the oracles obtained from $z_1, \ldots, z_t$. In this case, there is at least one correct oracle $\mathcal{O}_i$ among the oracles $\mathcal{O}_1, \ldots, \mathcal{O}_t$. Consequently, if we run $S(x, 1^t)$ with access to $\mathcal{O}_1, \ldots, \mathcal{O}_t$, we compute $L'(x)$ with high probability. Since $n = \mathsf{poly}(n')$, $t = \mathsf{poly}(n)$, the selector runs in time $\mathsf{poly}(n', t)$, and the simulation of each oracle query to $\mathcal{O}_i$ can be done using a computation of the corresponding bit of $G_N(z_i)$ in time $\mathsf{poly}(n)$, it follows that given x of length n' we can compute $L'(x)$ with high probability in time $\mathsf{poly}(n')$. Finally, since L' is complete for $\mathsf{EXP}^{\mathsf{NP}}$, it follows that $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{BPP}$, as desired. □

Inspecting the proof, it is not hard to see that it also extends to *almost-everywhere* and *sub-exponential* lower bounds. Since the proofs are straightforward modifications of the argument given above, we only state the results and omit the proof details.

Theorem 3.13. *The following statements are equivalent:*

(*i*) $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{i.o.BPP}$.

(*ii*) *For every choice of the parameters $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^{\ell}$, and for every sequence $\{G_N\}$ of maps $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed in time n^c, the* `Implicit-`δ`-Heavy-Avoid` *problem can be solved with access to an* NP *oracle in deterministic time* $\mathsf{poly}(N)$ for every large enough N.

Theorem 3.14. *The following statements are equivalent:*

(*i*) *There is a constant $\epsilon > 0$ such that* $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{BPTIME}[2^{n^{\epsilon}}]$.

(*ii*) *There is a constant $\epsilon > 0$ such that for $\delta(N) := 2^{-n^{\epsilon}}$ and for every sequence $\{G_N\}$ of maps $G_N \colon \{0,1\}^{2^{n^{\epsilon}}} \to \{0,1\}^N$ implicitly computed in time $2^{n^{\epsilon}}$, the* `Implicit-`δ`-Heavy-Avoid` *problem on $\{G_N\}$ can be solved with access to an* NP *oracle in deterministic time* $\mathsf{poly}(N)$ *on infinitely many values of N.*

One can also use the instance checkers for EXP-complete languages [BFL91] (which imply selectors for such languages [Hir15]) to prove similar characterizations for EXP. For example, the following theorem holds (we omit the details as it is the same as our second proof for $\mathsf{EXP}^{\mathsf{NP}}$):

Theorem 3.15. *The following statements are equivalent:*

(*i*) $\mathsf{EXP} \not\subseteq \mathsf{i.o.BPP}$.

(*ii*) *For every choice of the parameters $c, d, \ell \in \mathbb{N}$, with $m(N) = n^d$ and $\delta(N) = 1/n^{\ell}$, and for every sequence $\{G_N\}$ of maps $G_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed in time n^c, the* `Implicit-`δ`-Heavy-Avoid` *problem can be solved in deterministic* $\mathsf{poly}(N)$ *time for every large enough N.*

Remark 3.16 (Comparison between the two proof methods)*.* We presented two proofs for Theorem 3.8. Both proofs have their advantages and disadvantages, as we summarize below:

- The first proof only uses (non-adaptive) instance checkers [Che23] that have small circuit complexity overheads, hence it works for restricted circuit classes such as $\mathsf{AC}^0[\oplus]$ and ACC^0. On the other hand, the second proof needs to use the selectors for $\mathsf{EXP}^{\mathsf{NP}}$ [Hir15] which is highly adaptive, hence does not extend to smaller circuit classes such as $\mathcal{C} = \mathsf{NC}^1$ or $\mathcal{C} = \mathsf{TC}^0$.[21]

- The second proof proceeds by a direct argument and hence generalizes to almost-everywhere and sub-exponential time lower bounds (Theorems 3.13 and 3.14), while the first proof uses a win-win analysis and does not seem to generalize to these cases.

4 Unconditional Pseudodeterministic Algorithms for Heavy Avoid

4.1 A Pseudodeterministic Algorithm for Implicit Maps

We present an *unconditional* pseudodeterministic algorithm for $\texttt{Implicit-}\delta\texttt{-Heavy-Avoid}$ for general Boolean circuits on infinitely many input lengths. Our algorithm runs in $\mathsf{poly}(N)$ time and works for maps $G\colon \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^N$ (where $N = 2^n$ following our convention) implicitly computable in $\mathsf{poly}(n)$ time.

Theorem 4.1. *For every choice of positive constants c, d, and ℓ, with $m(N) = n^d$ and $\delta(N) = 1/n^\ell$, and for every sequence $\{G_N\}$ of maps $G_N\colon \{0,1\}^{m(N)} \to \{0,1\}^N$ implicitly computed in time n^c, the $\texttt{Implicit-}\delta\texttt{-Heavy-Avoid}$ problem can be solved pseudodeterministically in polynomial time on infinitely many input lengths. Moreover, the algorithm computes pseudodeterministically on every input length.*

Proof. Consider a map $G_N\colon \{0,1\}^{n^d} \to \{0,1\}^N$ that is implicitly computable in time n^c. We consider two cases, based on whether $\mathsf{EXP} = \mathsf{BPP}$.

If $\mathsf{EXP} \not\subseteq \mathsf{BPP}$, then by Theorem 3.9, the $\texttt{Implicit-}\delta\texttt{-Heavy-Avoid}$ problem can be solved in deterministic polynomial time (i.e., in time $\mathsf{poly}(N)$) on infinitely many input lengths. Since the algorithm is deterministic, it behaves pseudodeterministically on every input length.

In the remaining case, assume that $\mathsf{EXP} \subseteq \mathsf{BPP}$. Let $B(j)$ be the following Turing machine with input $j \in \{0,1\}^n$: It first goes over all choices of $x \in \{0,1\}^{n^d}$ and computes $G_N(x)$, then calculates the probability of each string in $\{0,1\}^N$ produced in this way, and finally outputs the j-th bit of the lexicographic first string y such that $\mathbf{Pr}[G_N(\mathcal{U}_{M(N)}) = y] < \delta$. Note that B runs in time exponential in n, its input length. Therefore, it defines a language $L_B \in \mathsf{EXP}$. By the assumption, $L_B \in \mathsf{BPP}$. Consequently, we can compute $y \in \{0,1\}^N$ from 1^N in pseudodeterministic time $\mathsf{poly}(N)$. Note that this algorithm succeeds on every input length. $\square$

The win-win argument presented in Theorem 4.1 is non-constructive, but it is possible to combine the two cases and the argument from Theorem 3.9 via instance checkers to give an explicit description of a procedure that solves $\mathcal{C}\texttt{-Implicit-}\delta\texttt{-Heavy-Avoid}$ on infinitely many input lengths.

[21]One can also construct non-adaptive instance checkers for EXP-complete languages from highly-efficient PCPs such as [BGH+06, BS08, BCGT13, BV14]. In fact, the results in [BV14] imply an instance checker whose circuit complexity is only a 3-CNF over its randomness r. However, it is unclear if that instance checker also has low circuit complexity over the input x. (Looking into the proof, it seems that one needs to at least compute a Reed–Solomon encoding of x). Therefore we chose to use the off-the-shelf $\mathsf{AC}^0[\oplus]$ instance checker in [Che23] for PSPACE and resort to a win-win argument for EXP. On the other hand, it is unclear to the authors whether the selectors in [Hir15] can be made non-adaptive.

While instance checkers are well know in the context of program verification and in complexity-theoretic results, to our knowledge this provides the first algorithm for a natural problem in a completely different context whose *design* makes use of instance checkers.

We can extend the proof of Theorem 4.1 to any nice circuit class $\mathcal{C}$. In this case, it is also possible to obtain a stronger conclusion using a more careful argument.

Theorem 4.2. *Let $\mathcal{C}$ be a nice circuit class. Let $\mathcal{D} = \{D_N\}_{N\geq 1}$ be a distribution ensemble, where each D_N is supported over $\{0,1\}^N$. Suppose that $\mathcal{D}$ admits implicit $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-samplers of size $\mathsf{poly}(n)$ (recall that $N = 2^n$). Then, for every function $\delta(N) = 1/n^k$, where $k \in \mathbb{N}$, either $(\mathcal{D}, \delta)$-`Heavy-Avoid` can be solved in space $O(\log N)$ on infinitely many input lengths, or $(\mathcal{D}, \delta)$-`Heavy-Avoid` can be solved pseudodeterministically by $\mathsf{DLOGTIME}$-uniform BP-$\mathcal{C}$-circuits of size $\mathsf{polylog}(N)$ on every input length.*[22]

Proof. Given an ensemble $\mathcal{D} = \{D_N\}_{N\geq 1}$ as in the statement, consider the corresponding map $G_N \colon \{0,1\}^{n^d} \to \{0,1\}^N$ implicitly computable by $\mathsf{DLOGTIME}$-uniform $\mathcal{C}$-samplers of size n^c. Let $k \geq 1$ be arbitrary, and set $\delta(n) = 1/n^k$. We consider two cases, based on whether $\mathsf{PSPACE} \subseteq \mathsf{BP}$-$\mathcal{C}$.

If $\mathsf{PSPACE} \nsubseteq \mathsf{BP}$-$\mathcal{C}$, then by Theorem 3.2, the $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem can be solved in space $O(\log N)$ on infinitely many input lengths.

In the remaining case, assume that $\mathsf{PSPACE} \subseteq \mathsf{BP}$-$\mathcal{C}$. We show how to solve the Heavy Avoid problem for G_N even for the exponentially smaller threshold $\delta(N) = 1/N^k$. Let $B(j)$ be the following Turing machine with input $j \in \{0,1\}^n$, which we also view as a natural number in $[N]$: if $j > 2k \cdot n$, it outputs 0; otherwise, it goes in lexicographic order over all strings of length $2k \cdot n$ and outputs the i-th bit of $G_N(y^\star)$, where $y^\star$ is the first string such that the $(2k \cdot n)$-prefix of $G_N(y^\star)$ is produced by $G_N(\mathcal{U}_{n^d})$ with probability less than δ. Note that B runs in space $\mathsf{poly}(n)$, since G_N is implicitly computable. Moreover, since $\delta = 1/N^k$, there exists some prefix of length $2k \cdot n$ that is produced by $G_N(\mathcal{U}_{n^d})$ with probability less than δ. In particular, this prefix concatenated with a sequence of 0's is a valid solution to the Heavy Avoid problem of D_N, the distribution sampled by G_N.

Since B runs in space polynomial in n, it defines a language $L_B \in \mathsf{PSPACE}$. By the assumption, $L_B \in \mathsf{BP}$-$\mathcal{C}$. Consequently, we can compute each bit of $G_N(y^\star) \in \{0,1\}^N$ given $i \in [N] = \{0,1\}^n$ using $\mathsf{DLOGTIME}$-uniform BP-$\mathcal{C}$-circuits of size $\mathsf{poly}(n) = \mathsf{polylog}(N)$. ☐

4.2 Universality and Connection to Time-Bounded Kolmogorov Complexity

In this section, we observe the existence of a "hardest" instance of uniform Heavy Avoid (Definition 2.1). We also point out a connection to time-bounded Kolmogorov complexity. These connections will be used, in Section 4.3, to obtain an improved pseudodeterministic algorithm as compared to Theorem 4.1.

First, we introduce a collection of sequences of probability distributions, where each sequence in the collection is associated with a time bound $t(n) = n^c$, $c \in \mathbb{N}$. We let $\mathcal{D}^t = \{D^t_n\}_{n\geq 1}$, where each sampler $D^t_n \colon \{0,1\}^{\log n + n + t + n} \to \{0,1\}^n$ is defined as follows. The input to D^t_n is a random string x that we view as a tuple (i, M, r, z), where $i \in [n]$, $M \in \{0,1\}^n$, $r \in \{0,1\}^t$, and $z \in \{0,1\}^n$. We let $M_{\leq i}$ denote the first i bits of M, which we view as an input to a universal Turing machine U. Let $y = U(1^t, M_{\leq i}, r)$ be the output of U on machine $M_{\leq i}$ with random string r after it computes for at most t steps. If y is an n-bit string, then D^t_n outputs y. Otherwise, D^t_n outputs z.

Observe that $\mathcal{D}^t \in \mathsf{PSAMP}$ for any polynomial t. Below we abuse notation and use $D^t_n(w)$ to denote the probability of a string w under the distribution induced by the sampler D^t_n.

[22] In other words, the corresponding $\mathsf{DLOGTIME}$-uniform BP-$\mathcal{C}$-circuit is given $i \in [N] = \{0,1\}^{\log N}$ and outputs the i-th bit of the solution with high probability.

Proposition 4.3 (Universality of $\mathcal{D}^t$ for Uniform Heavy Avoid). *Let $\mathcal{D} = \{D_n\}_{n \geq 1}$ be a distribution samplable in time n^c, where $c \in \mathbb{N}$. Let $\delta(n) \in [0,1]$, and consider the corresponding $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem. Let $t(n) = n^{c+1}$ and $\gamma(n) = \delta(n)/(Kn^2)$, where $K \geq 1$ is a constant that depends only on $\mathcal{D}$. Then for large enough n, every solution $w \in \{0,1\}^n$ to $(D_n^t, \gamma(n))$-`Heavy-Avoid` is a solution to $(D_n, \delta(n))$-`Heavy-Avoid`.*

Proof. We need to prove that if $D_n(w) > \delta$ then $D_n^t(w) > \delta/(Kn^2)$. Let $k \in \mathbb{N}$ denote the length of the time-n^c program M' that samples the ensemble $\mathcal{D}$. We let M be the program that samples from D_n. Note that M can be encoded with $k + \log n$ bits. Set $K = 2^k$.

Now with probability $1/n$ we obtain $i = k + \log n$ as input to sampler D_n^t, assuming that n is large enough. Moreover, with probability at least $1/(2^{k+\log n}) = 1/(Kn)$ the sampler D_n^t selects the description of M as input to the universal machine. Conditioned on these events, the output of D_n^t is distributed according to D_n. Consequently, if $D_n(w) > \delta$ we have $D_n^t(w) > \delta/(Kn^2)$. ☐

Let $\mathsf{pK}^t(z)$ denote the probabilistic t-time bounded Kolmogorov complexity of z [GKLO22]. Let rKt denote the randomized time-bounded Kolmogorov complexity of z [Oli19]. (These definitions are reviewed in Section 2.3.)

Lemma 4.4. *Let $t(n) = n^c$ for some constant $c \in \mathbb{N}$, and consider the distribution ensemble $\mathcal{D}^t = \{D_n^t\}_{n \geq 1}$ introduced above. There is a constant $d \geq 1$ such that, for every $n \in \mathbb{N}$ and for every $w \in \{0,1\}^n$, the following statements hold:*

(*i*) *If $D_n^t(w) \geq \delta$ then both $\mathsf{pK}^{n^d}(w) \leq \log\left(\frac{1}{\delta}\right) + d \cdot \log n$ and $\mathsf{rKt}(w) \leq d \cdot \log\left(\frac{1}{\delta}\right) + d \cdot \log n$.*

(*ii*) *If $\mathsf{pK}^t(w) = k$ then $D_n^t(w) \geq \frac{1}{2} \cdot \frac{1}{n} \cdot 2^{-k}$.*

Proof. The first item is an immediate consequence of the coding theorem for pK^t complexity from [LOZ22] and of the coding theorem for rKt complexity from [LO21]. The second item follows from the definitions of D_n^t and pK^t complexity. ☐

The next result shows that any solution $y \in \{0,1\}^n$ to the (D_n^t, δ)-`Heavy-Avoid` problem must have non-trivial pK^t complexity. In particular, if $\delta = n^{-c}$ then $\mathsf{pK}^t(y) \geq (c-1) \cdot \log n - 1$.

Proposition 4.5 (Uniform Heavy Avoid and Time-Bounded Kolmogorov Complexity). *Consider the distribution $\mathcal{D}^t = \{D_n^t\}_{n \geq 1}$ and a probability threshold δ. If $D_n^t(y) \leq \delta$ then $\mathsf{pK}^t(y) \geq \log(1/\delta) - \log(n) - 1$.*

Proof. This follows immediately from Lemma 4.4 Item (*ii*). ☐

Conversely, by Lemma 4.4 Item (*i*), in order to solve the (D_n^t, δ)-`Heavy-Avoid` problem for $t(n) = n^c$ it is enough to output a string $y \in \{0,1\}^n$ such that $\mathsf{pK}^{n^d}(y) > \log(1/\delta) + d \cdot \log n$. Similarly, it is enough that $\mathsf{rKt}(y) > d \cdot \log(1/\delta) + d \cdot \log n$.

We obtain the following consequence from the discussion in this section.

Proposition 4.6. *Let $\mathcal{D} = \{D_n\}_{n \geq 1}$ be a polynomial-time samplable ensemble of distributions, with D_n supported over $\{0,1\}^n$, and let $\delta(n) = 1/\mathsf{poly}(n)$. Then there is a constant $k \geq 1$ such that, for every large enough n, if $y \in \{0,1\}^n$ is a string such that $\mathsf{rKt}(y) \geq k \cdot \log n$, then y is a solution to the $(D_n, \delta(n))$-`Heavy-Avoid` problem.*

Proof. This follows from Proposition 4.3 and Lemma 4.4 Item (*i*). ☐

4.3 Subexponential-Time Pseudodeterministic Algorithms for Explicit Maps

In this section, we prove the following result.

Theorem 4.7. *Let $\mathcal{D} = \{D_n\}_{n \geq 1}$ be a polynomial-time samplable ensemble of distributions, with D_n supported over $\{0,1\}^n$, and let $\delta(n) = 1/\mathsf{poly}(n)$. Then, for every $\varepsilon > 0$, there is a pseudodeterministic algorithm running in time $O(2^{n^{\varepsilon}})$ that solves $(D_n, \delta(n))$-`Heavy-Avoid` problem on infinitely many values of n. Moreover, this algorithm behaves pseudodeterministically on every input.*

Proof. Let $\mathcal{D} = \{D_n\}_{n \geq 1}$ be a polynomial-time samplable ensemble of distributions, with D_n supported over $\{0,1\}^n$, and let $\delta(n) = 1/\mathsf{poly}(n)$. In order to solve the Heavy Avoid problem for this distribution ensemble, it is enough to pseudodeterministically construct an n-bit string y of large enough rKt complexity (Proposition 4.6).

For this, we rely on an unconditional result from [LOS21]. For every integer $d \geq 1$ and $\varepsilon > 0$, [LOS21, Theorem 39] pseudodeterministically constructs a string w of length $d \cdot \log n$ with $\mathsf{rKt}(w) \geq (d/2) \cdot \log n$ in time $2^{n^{\varepsilon}}$, for infinitely many values of n. Additionally, the corresponding procedure behaves pseudodeterministically on all input strings.[23]

Since we can efficiently recover w from a padded version of w, it is not hard to see that if we set $y = w0^{n-|w|}$, then $\mathsf{rKt}(y) \geq (d/8) \cdot \log n$, assuming that $d \geq 8$. Consequently, we can infinitely-often pseudodeterministically construct in time $2^{n^{\gamma}}$ an n-bit string of rKt complexity at least $k \cdot \log n$, where $\gamma > 0$ and $k \geq 1$ are arbitrary constants. $\square$

Comparison between Theorem 4.7 and Theorem 4.1. Next, we explain that Theorem 4.7 is stronger than Theorem 4.1. This is not immediately obvious, since the algorithms from Theorem 4.7 work in a more general setting but have a sub-exponential running time as a function of the output length, while the algorithms for the more restrictive implicit maps provided by Theorem 4.1 run in polynomial time as a function of the output length.

Consider a map $G_N \colon \{0,1\}^{n^d} \to \{0,1\}^N$ implicitly computed by a uniform circuit $C_N \colon \{0,1\}^{n^d+n} \to \{0,1\}$ of size at most n^c, where $N = 2^n$. Suppose that C_N is the output of $A(1^n)$, a uniform generating procedure that runs in time $O(n^k)$. Consider the associated $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` problem with threshold $\delta(N) = 1/n^{\ell}$, and let D_N be the probability distribution supported over $\{0,1\}^N$ that is induced by the map G_N implicitly computed by C_N.

For a given n, let $n' = n^a$, for a large enough constant $a \geq 1$, $m' = m = n^d$, and $\delta'(n') = \delta(N) = 1/n^{\ell}$. Consider the distribution $D'_{n'}$ supported over $\{0,1\}^{n'}$ and defined as follows: sample a random string $w \sim \{0,1\}^{m'}$, let $x^w = G_N(w) \in \{0,1\}^N$, and output the length-$n'$ prefix of x^w.

We make the following observations:

- The ensemble $\mathcal{D}' = \{D'_n\}_{n \geq 1}$ is polynomial-time samplable, since the circuit C_N that implicitly samples D_N is generated by $A(1^n)$ in time polynomial in n.
- If the constant a is large enough, by our choice of n', there is at least one element in the support of $D'_{n'}$ that is not δ'-heavy.
- If an element $y' \in \{0,1\}^{n'}$ is not δ'-heavy in $D'_{n'}$, then $y = y'0^{N-n'}$ is not δ-heavy in D_N.

Now suppose that a pseudodeterministic algorithm E solves $(D'_{n'}, \delta'(n'))$-`Heavy-Avoid` and runs in time $O(2^{n'^{\varepsilon'}})$, for a fixed but arbitrary $\varepsilon' > 0$, as in the case of Theorem 4.7. Let $y' =$

[23]This is implicit in the proof of [LOS21, Theorem 39]. In more detail, the proof of this result relies on the pseudodeterministic algorithm for CAPP given by [LOS21, Theorem 27], which has pseudodeterministic behavior on all input lengths [LOS21, Appendix A].

$E(1^{n'})$. Then, given 1^N, if we use E to compute y' then output $y = y'0^{N-n'}$, we can solve $\mathcal{C}$-`Implicit`-δ-`Heavy-Avoid` in pseudodeterministic time

$$O(N + 2^{n'^{\varepsilon'}}) = O(N),$$

assuming that $\varepsilon' < 1/a$.

More generally, using the "prefix" reduction described above, one can easily show that distributions D_n supported over $\{0,1\}^n$ that are sampled with $\mathsf{polylog}(n)$ random bits admit polynomial-time infinitely-often pseudodeterministic algorithms for the $(D_n, \delta(n))$-`Heavy-Avoid` problem, as long as $\delta(n) \geq 1/\mathsf{poly}(\log(n))$.[24] We summarize this discussion as follows.

Theorem 4.8. *Let $\mathcal{D} = \{D_n\}_{n \geq 1}$ be a distribution ensemble, where each D_n is supported over $\{0,1\}^n$, and suppose that $\mathcal{D}$ admits a polynomial-time sampler of randomness complexity $(\log n)^c$. Then, for every function $\delta(n) = 1/(\log n)^k$, there is a polynomial time pseudodeterministic algorithm for the $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem that succeeds on infinitely many values of n. Moreover, this algorithm behaves pseudodeterministically on every input.*

4.4 On Pseudodeterministic Algorithms and Hierarchies for Probabilistic Time

Theorem 4.1 provides a pseudodeterministic polynomial-time infinitely-often algorithm for heavy avoid for implicit maps. Moreover, this algorithm computes pseudodeterministically on all input lengths. In this section, we observe that the existence of a pseudodeterministic algorithm with these properties for the larger class of explicit maps (or even just for implicit maps with polynomial stretch instead of subexponential stretch) would solve a longstanding open problem related to hierarchies for probabilistic time.

To give more context, we know that BPP is strictly contained within $\mathsf{BPSUBEXP}$ [KV87], but the question of whether $\mathsf{BPTIME}[n]$ is strictly contained within $\mathsf{BPTIME}[T(n)]$, for any function T that remains sub-exponential even when composed with itself a constant number of times, remains open. Progress has been made in establishing hierarchies for variants of BPP. The papers [Bar02, FS04, vMP06] demonstrated hierarchies for $\mathsf{BPP}/_1$ (problems solvable in probabilistic polynomial time with 1 bit of advice) and for $\mathsf{Heur}\text{-}\mathsf{BPP}$ (problems solvable on average in probabilistic polynomial time). However, despite extensive efforts, establishing a hierarchy for BPP itself remains an open problem.

Proposition 4.9. *Suppose that for every polynomial-time samplable distribution ensemble $\mathcal{D} = \{D_N\}_{N \geq 1}$, there is a function $\delta(N) = o(1)$ such that the corresponding $(\mathcal{D}, \delta)$-`Heavy-Avoid` problem admits a pseudodeterministic polynomial-time algorithm that succeeds on infinitely many input lengths and behaves pseudodeterministically on all input lengths. Then, for every constant $k \geq 1$, we have $\mathsf{BPE} \not\subseteq \mathsf{BPTIME}[2^{k \cdot n}]$ (in particular, for every constant $c \geq 1$, $\mathsf{BPP} \not\subseteq \mathsf{BPTIME}[n^c]$).*

Proof. Let $k \geq 1$. Using the assumption, we show below that there is $L \in \mathsf{BPTIME}[2^{O(n)}]$ such that $L \notin \mathsf{BPTIME}[2^{k \cdot n}]$. Note that the additional conclusion in the theorem follows from a standard padding argument.

Let $N(n) = 2^n$, and consider the uniform sampler $S_N \colon \{0,1\}^{m(N)} \to \{0,1\}^N$ defined as follows, where $m(N) = N^{k+2} = 2^{(k+2) \cdot n}$. The sampler parses its input string x as a pair (M, r), where M describes a clocked probabilistic machine running in time $2^{(k+1) \cdot n}$, and the rest bits r are treated

[24] In order to maintain the infinitely-often guarantee in the prefix reduction, one needs to make sure that the map from the large output parameter n to the smaller output parameter n' is onto, which is easy to achieve.

as randomness. For $i \in [N]$, which we also view as an n-bit string, the i-th output bit of $S_N(x)$ is the output of the computation of M over the string i when running with the random string r. (For concreteness, if M on (i, r) does not produce an output bit, we assume its output is 0.) Note that the resulting distribution ensemble $\mathcal{D} = \{D_N\}$ obtained from the sampler S_N is indeed in PSAMP.

Under the assumption of the theorem, $(\mathcal{D}, \delta)$-`Heavy-Avoid` with $\delta(N) = o(1)$ admits a pseudodeterministic algorithm running in time $\mathsf{poly}(N) = 2^{O(n)}$ that succeeds on infinitely many input lengths N and behaves pseudodeterministically on all input lengths. Let $B(1^N)$ be such a pseudodeterministic algorithm. In addition, let L_B be the language defined by B, i.e., a string $z \in \{0,1\}^n$ is in L_B if and only if the z-th bit of $B(1^N)$ (with $N = 2^n$) is 1. Note that L_B is in $\mathsf{BPTIME}[2^{O(n)}]$, since by assumption B computes pseudodeterministically on every input 1^N.

It remains to argue that $L_B \notin \mathsf{BPTIME}[2^{k \cdot n}]$. To prove this, it is enough to show that for every language $L \in \mathsf{BPTIME}[2^{k \cdot n}]$, each string in the sequence $\{y_N^L\}_N$ of truth-tables obtained from L is $\delta(N)$-heavy in $D_N = S_N(\mathcal{U}_{m(N)})$, for every large enough N. Since B correctly solves $(\mathcal{D}, \delta)$-`Heavy-Avoid` on infinitely many values of N, we claim that L_B and L differ on each such value of N, provided that N is large enough. To see that the claim holds, note that if $L \in \mathsf{BPTIME}[2^{k \cdot n}]$ then by amplification there is a probabilistic machine M_L that runs in time at most $2^{kn} \cdot \mathsf{poly}(n) \leq 2^{(k+1) \cdot n}$ and computes M_L on every input string of length n with probability at least $1 - 2^{-2n}$. Moreover, the description length of M_L is a constant. Let y_N^L be the truth-table of L on input length n, then $|y_N^L| = N = 2^n$. By construction, using an union bound, the probability that $S_N(\mathcal{U}_{M(N)}) = y_N^L$ is at least $\Omega(1) \cdot (1 - 2^{-n}) > \delta(N)$, as long as N is large enough. This shows that y_N^L is $\delta(N)$-heavy in D_N. Since $B(1^n)$ avoids $\delta(N)$-heavy elements of D_N on infinitely many values of N, this concludes the proof. □

We note that, using a similar argument, the existence of polynomial-time algorithms for the heavy avoid problem for implicit maps with certain parameters would also lead to new hierarchy theorems. For instance, efficient algorithms for maps $G \colon \{0,1\}^{2^{n^\delta}} \to \{0,1\}^{2^n}$ that are implicitly computed by uniform circuits of size 2^{n^δ} would imply that $\mathsf{BPE} \not\subseteq \mathsf{BPTIME}[2^{n^{o(1)}}]$.

5 Heavy Avoid and Derandomization

In this section, we study the relation between algorithms for Heavy Avoid and derandomization, with connections to recent developments in *instance-wise* hardness-randomness tradeoffs [CT21, LP22, Kor22, LP23, CTW23]. This section mainly considers Heavy Avoid for *non-uniformly* and *implicitly* sampled distributions.

5.1 A Non-Black-Box Reduction

We show that in some scenarios, solving the `Implicit-Heavy-Avoid` problem on non-uniform samplers implies general derandomization of prBPP. Intriguingly, our reduction from prBPP to Heavy Avoid is *non-black-box* and relies on the *code* of an algorithm for `Implicit-Heavy-Avoid`.

We first introduce appropriate notation. We say that a Boolean circuit family $\{C_n\}$ has *dimension* $d(n) \times T(n)$ if for each $n \in \mathbb{N}$, the gates of C_n are partitioned into $d(n)$ layers, each layer contains at most $T(n)$ gates, and each gate on layer i only receives inputs from layer $i - 1$. The *depth* of the circuit is $d(n)$ and the *width* of the circuit is $T(n)$. We will always assume $d(n) \leq T(n)$. A circuit family of dimension $d(n) \times T(n)$ is *logspace-uniform* if there is a Turing machine that on input 1^n, uses at most $O(\log T(n))$ space, and prints the description of C_n.

Recall that for $\epsilon(n) > 0$, a Boolean function $f \colon \{0,1\}^M \to \{0,1\}$ is ϵ-*dense* if $\mathbf{Pr}_{x \sim \{0,1\}^M}[f(x) = 1] \geq \epsilon(M)$. We say f ϵ-*avoids* a hitting set $H \subseteq \{0,1\}^M$ if f is ϵ-dense and for every $y \in H$, $f(y) = 0$.

Now we are ready to state our main technical tool, the instance-wise hardness-randomness tradeoff in [CT21].

Theorem 5.1 ([CT21] with the improved parameters from [CLO$^+$23])**.** *There is an absolute constant $c \geq 1$ such that the following holds. Let $f\colon \{0,1\}^n \to \{0,1\}^{T(n)}$ be a multi-output function computable by a logspace-uniform circuit of dimension $d(n) \times T(n)$. Let $M(n)$ be a parameter such that $c \log T \leq M \leq T^{1/c}$. Then there are algorithms $\mathsf{CT21.HSG}_f$ and $\mathsf{CT21.Recon}_f$ depending on f, such that:*

- *The algorithm $\mathsf{CT21.HSG}_f(x)$ runs in deterministic T^c time and outputs a set of M-bit strings.*
- *Given $x \in \{0,1\}^n$ and $i \in [T]$ as inputs, and oracle access to a candidate distinguisher $D\colon \{0,1\}^M \to \{0,1\}$, $\mathsf{CT21.Recon}_f^D(x,i)$ runs in randomized $(dnM)^c$ time. If D $(1/M)$-avoids $\mathsf{CT21.HSG}_f(x)$, then with probability $\geq 1 - 2^{-M}$, $\mathsf{CT21.Recon}_f^D(x,i)$ outputs the i-th bit of $f(x)$.*

Moreover, there is a deterministic algorithm that, given the Turing machine M_f that prints the circuit for f in logspace, outputs the descriptions of $\mathsf{CT21.HSG}_f$ and $\mathsf{CT21.Recon}_f$ in $\mathsf{poly}(|\langle M_f \rangle|)$ time.

Remark 5.2. The hardness-randomness tradeoffs in [CT21, CLO$^+$23] were stated for hard functions of the form $f\colon \{0,1\}^n \to \{0,1\}^n$, where the reconstruction algorithm prints the entire string $f(x)$ in $\ll T$ time. Inspecting their proofs, it is easy to see that the same holds when we have a function $f\colon \{0,1\}^n \to \{0,1\}^T$ and the reconstruction algorithm prints the i-th bit of $f(x)$ in $\ll T$ time, given $i \in [T]$ as an input.

Theorem 5.3 (Non-black-box reduction under logspace-uniform sub-polynomial depth algorithms)**.** *Let $\delta(n) = o(1)$ be any function. Suppose there is a constant $\epsilon > 0$ and an algorithm $\mathcal{A}(\langle C \rangle)$ that solves the* `Implicit-`δ`-Heavy-Avoid` *problem on instances $G\colon \{0,1\}^{N^\epsilon} \to \{0,1\}^N$ that are implicitly computable by a circuit C of size N^ϵ, where a description of C is given as input. Moreover, assume that $\mathcal{A}$ can be implemented as a logspace-uniform circuit of size $\mathsf{poly}(N)$ and depth $N^{o(1)}$. Then $\mathsf{prBPP} = \mathsf{prP}$.*

Proof. Since $\mathsf{prBPP} \subseteq \mathsf{prRP}^{\mathsf{prRP}}$ [BF99], it suffices to prove that $\mathsf{prRP} = \mathsf{prP}$. That is, we want a deterministic algorithm that given as input a size-$2M$ circuit $D\colon \{0,1\}^M \to \{0,1\}$, distinguishes between the case that D rejects every input and the case that D is $1/2$-dense.

Let c be the constant in Theorem 5.1, and set $N := M^{3c/\epsilon}$. We recall our assumption: given an implicit map with parameters as above, for some $T \leq \mathsf{poly}(N)$, $d \leq N^{o(1)}$ and $\delta \leq o(1)$, there is a logspace-uniform circuit $\mathcal{A}$ of dimension $d \times T$ that solves the corresponding `Implicit-`δ`-Heavy-Avoid` problem deterministically.

Consider the implicit map $G_D\colon \{0,1\}^{N^\epsilon} \to \{0,1\}^N$, where the underlying circuit C_D for computing each output bit of G_D is as follows. Given $x \in \{0,1\}^{N^\epsilon}$ and $i \in [N]$, we parse x as (M_f, r), where M_f is a program, and r consists of the remaining bits of x (treated as randomness). We can encode x in such a way that every program of length ℓ appears with probability mass $\Theta(2^{-\ell}/\ell^2)$, hence every program with constant description length appears with probability mass $\Omega(1) > \delta$. Suppose that M_f is a logspace-uniform Turing machine that defines a circuit of size $T \cdot \mathsf{poly}(M)$ and depth $d + \mathsf{polylog}(M)$ that computes a function $f\colon \{0,1\}^{\tilde{O}(M)} \to \{0,1\}^N$ (this can be syntactically ensured by imposing a space constraint on M_f). We let

$$C_D(x,i) := \mathsf{CT21.Recon}_f^D(\langle D \rangle, i; r). \tag{1}$$

Here, $\mathsf{CT21.Recon}(\langle D\rangle, i; r)$ denotes the output of $\mathsf{CT21.Recon}(\langle D\rangle, i)$ *on randomness* r. Note that when we compute $C_D(x,i)$, we treat D both as the distinguisher (for the reconstruction algorithm $\mathsf{CT21.Recon}$) and as the input of f. We then run $\mathsf{CT21.Recon}$ on randomness r and (attempt to) reconstruct the i-th bit of $f(\langle D\rangle)$. If the length of $\langle M_f\rangle$ is a constant, then we have $|r| \geq N^\epsilon - O(1) \geq M^{2.9c} \geq (d^2 \cdot \tilde{O}(M)\cdot M)^c$, hence there is always enough randomness to feed into Recon. Also, by Theorem 5.1, the description $\langle C_D\rangle$ can be computed from the description $\langle D\rangle$ in $\mathsf{poly}(M)$ time and $\mathsf{polylog}(M)$ depth.[25] Note that G_D can be implicitly computed in time N^ϵ due to our choice of parameters.

Define a function $f'\colon \{0,1\}^{\tilde{O}(M)} \to \{0,1\}^N$ as follows: Given a size-$2M$ circuit $D\colon \{0,1\}^M \to \{0,1\}$ as input, f' computes the circuit C_D as in (1) and outputs $\mathcal{A}(\langle C_D\rangle)$. Recall that $\mathcal{A}$ is computed by a logspace-uniform circuit of dimension $d\times T$, hence f' is computed by a logspace-uniform circuit of size $T\cdot\mathsf{poly}(M)$ and depth $d+\mathsf{polylog}(M)$.

Assuming D is $1/2$-dense, we argue that $\mathsf{CT21.HSG}_{f'}(\langle D\rangle)$ hits D. Otherwise, D $(1/2)$-avoids $\mathsf{CT21.HSG}_{f'}(\langle D\rangle)$. By Theorem 5.1, for every $i\in[N]$, with probability at least $1-2^{-M}$ over the randomness r, $\mathsf{CT21.Recon}^D_{f'}(\langle D\rangle, i; r)$ is equal to the i-th output bit of $f'(\langle D\rangle)$. By a union bound, w.p. at least $1 - N\cdot 2^{-M}$ over the randomness r, we have that for every $i\in[N]$, $\mathsf{CT21.Recon}^D_{f'}(\langle D\rangle, i; r) = f'(\langle D\rangle)_i$. Since $f'(\langle D\rangle) = \mathcal{A}(\langle C_D\rangle)$ by definition, we have:

$$\Pr_x[G_D(x) = \mathcal{A}(\langle C_D\rangle) \mid \langle M_f\rangle = \langle M_{f'}\rangle \text{ where } (M_f, r) = x] \geq 1 - N\cdot 2^{-M} \geq 1/2.$$

Since the description length of $M_{f'}$ is a constant, it follows that $\langle M_f\rangle = \langle M_{f'}\rangle$ with constant probability. Hence,

$$\Pr_x[G_D(x) = \mathcal{A}(\langle C_D\rangle)] \geq \Omega(1),$$

contradicting our assumption that $\mathcal{A}$ solves the `Implicit-δ-Heavy-Avoid` problem.

We have shown that if D is $1/2$-dense, then $\mathsf{CT21.HSG}_{f'}(\langle D\rangle)$ hits D. We can compute $\mathsf{CT21.HSG}_{f'}(\langle D\rangle)$ in $\mathsf{poly}(T,N) \leq \mathsf{poly}(M)$ time, hence we can solve the $\mathsf{GAP\text{-}UNSAT}$ problem in deterministic polynomial time. This implies that $\mathsf{prRP} = \mathsf{prP}$, as desired. □

Remark 5.4. The above reduction is non-black-box for two reasons. First, the statement $\mathsf{prBPP} \subseteq \mathsf{prRP}^{\mathsf{prRP}}$ can be seen as a non-black-box reduction from prBPP to prRP [BF99]. Second, and perhaps more interestingly, the reduction from prRP to `Implicit-Heavy-Avoid` is also non-black-box, as it requires the algorithm $\mathcal{A}$ for `Implicit-Heavy-Avoid` to be a *logspace-uniform circuit of low depth* and relies on an application of Theorem 5.1 over this low-depth circuit. Indeed, the proof of Theorem 5.1 performs arithmetization on this low-depth circuit.

Remark 5.5. It is also interesting to compare Theorem 5.3 with [Kor22, Theorem 8]. The latter result is a *black-box* reduction from prBPP to a problem called R-Lossy Code. In contrast, our Theorem 5.3 needs additional constraints on the algorithm solving `Implicit-Heavy-Avoid` and makes non-black-box use of that algorithm.

Our `Implicit-Heavy-Avoid` is a special case of R-Lossy Code in the following sense. Recall that R-Lossy Code is the problem where, given circuits $\mathsf{Comp} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^{n-1}$ and $\mathsf{Decomp} : \{0,1\}^{n-1} \to \{0,1\}^n$ and a parameter $\delta > 0$, one needs to find some $x\in\{0,1\}^n$ such that $\Pr_{r\leftarrow\{0,1\}^m}[\mathsf{Decomp}(\mathsf{Comp}(x,r)) = x] < \delta$. Given an implicit map $C : \{0,1\}^r \times [N] \to \{0,1\}$ as the input of `Implicit-δ-Heavy-Avoid` (recall that $r < N$ in the typical parameter setting), we can reduce it to the R-Lossy Code instance $(\mathsf{Comp}, \mathsf{Decomp})$ where $\mathsf{Comp}(x,r)$ simply outputs its

[25]The depth upper bound is dominated by computing the description of $\mathsf{CT21.Recon}_f$ from M_f, which takes time $\mathsf{polylog}(M)$. Although $\mathsf{CT21.Recon}_f$ is an *adaptive* oracle algorithm, to compute the *code* of $\mathsf{CT21.Recon}^D_f$ from $\langle D\rangle$ we only need to concatenate the codes of $\mathsf{CT21.Recon}_f$ and D together, hence this step is depth-efficient.

randomness r, and $\mathsf{Decomp}(r) = C(r, 1)C(r, 2) \ldots C(r, N)$. In this sense, `Implicit-Heavy-Avoid` is no more than a special case of R-Lossy Code where the compressor circuit is *trivial*.

5.2 Getting Rid of the Depth Assumption

In this section, we show how to get rid of the low-depth assumption in Theorem 5.3 in a weaker but non-trivial setting. An ideal statement would be: a deterministic polynomial-time algorithm for `Implicit-Heavy-Avoid` implies a deterministic polynomial-time algorithm for GAP-SAT or CAPP. Compared to the ideal statement, the actual result that we prove only holds for subexponential-time infinitely-often algorithms, as we shall explain later.

We note that it should not be too surprising that a subexponential-time infinitely-often algorithm for `Implicit-Heavy-Avoid` implies a subexponential-time infinitely-often algorithm for GAP-SAT. In fact, combining Theorem 3.9 and [IW01], it is easy to show that this holds for *heuristic* algorithms.[26]

Theorem 5.6. *The following items are equivalent.*

- (*Infinitely-often subexponential-time heuristic derandomization.*)

  For every language $L \in \mathsf{BPP}$, every ensemble of polynomial-time samplable distributions $\mathcal{D} = \{D_n\} \in \mathsf{PSAMP}$, every polynomial $p(\cdot)$, and every constant $\delta > 0$, there exists a deterministic Turing machine M running in 2^{n^δ} time, such that for infinitely many input lengths n,

$$\Pr_{x \sim D_n}[L(x) \neq M(x)] \leq 1/p(n).$$

- (*Infinitely-often polynomial-time algorithms for uniform* `Implicit-Heavy-Avoid`.)

  For every $\delta(N) = 1/\mathsf{polylog}(N)$, and for every sequence $\{G_N\}$ of maps $G_N : \{0,1\}^{\mathsf{polylog}(N)} \to \{0,1\}^N$ where each bit of G_N is implicitly computed in $\mathsf{polylog}(N)$ time, there is a deterministic $\mathsf{poly}(N)$-time algorithm that solves the `Implicit-`δ`-Heavy-Avoid` *problem for $\{G_N\}$ on infinitely many input lengths N.*

Proof Sketch. In fact, both items are equivalent to $\mathsf{EXP} \neq \mathsf{BPP}$. The equivalence between the first item and $\mathsf{EXP} \neq \mathsf{BPP}$ is shown in [IW01], and the equivalence between the second item and $\mathsf{EXP} \neq \mathsf{BPP}$ follows from Theorem 3.9. □

We now attempt to prove a version of Theorem 5.6 with respect to *worst-case* algorithms, instead of *heuristics*. Our worst-case version of Theorem 5.6 also has many caveats such as being infinitely-often and requiring subexponential time, but the biggest caveat might be that we could only obtain infinitely-often algorithms in the following, somewhat artificial, setting: For a sequence of inputs $\{x_n\}_{n\in\mathbb{N}}$, the algorithms read many inputs $x_1, x_2, \ldots, x_{\mathsf{poly}(n)}$ but are only required to solve x_n. We call this "infinitely-often*" algorithms. Formally, we have:

Definition 5.7. Let $\mathcal{P}$ be a computational problem and $\{x_n\}_{n\in\mathbb{N}}$ be a sequence of inputs. We say an algorithm $\mathcal{A}$ *infinitely-often* solves $\mathcal{P}$ on $\{x_n\}$* if there is a polynomial $p(\cdot)$ such that for infinitely many integers n, $\mathcal{A}(1^n, x_1, x_2, \ldots, x_{p(n)})$ outputs a valid $\mathcal{P}$-solution for x_n.

[26]Note that Theorem 5.6 refers to `Implicit-Heavy-Avoid` for *uniformly samplable* distributions, which is different from most results in this section. We believe that more connections between `Implicit-Heavy-Avoid` for *uniformly samplable* distributions and *average-case* derandomization can be obtained (e.g., using the recent "unstructured hardness to average-case randomness" [CRT22]), but we do not pursue this direction here.

In other words, an infinitely-often* algorithm has access to the (non-uniform) sequence of inputs around x_n, as opposed to the usual setting where the algorithm only has access to the given input string. We remark that our algorithm in Theorem 5.8 works in a weaker model where the machine only reads x_n and $x_{p(n)}$ and outputs an answer for x_n. However, we believe that Definition 5.7 is a more accurate model to capture win-win analyses in complexity theory, hence choose to define it in this way.

For ease of notation, in what follows, we will denote the sequence $x_1, x_2, \ldots, x_\ell$ simply by $x_{1\sim\ell}$. We are now able to state our main result in this section.

Theorem 5.8 (A non-black-box reduction for `Implicit-Heavy-Avoid`)**.**
Assume there is an infinitely-often polynomial-time algorithm for `Implicit-Heavy-Avoid` *with subexponential stretch. That is, for every constants $a \geq 1$, $\epsilon > 0$, and function $\delta \leq o(1)$, there exists a deterministic algorithm* Avoid *running in* $\mathsf{poly}(N)$ *time such that for infinitely many $n \in \mathbb{N}$ and $N := 2^{n^\epsilon}$, and for every generator $G\colon \{0,1\}^{n^a} \to \{0,1\}^N$ implicitly described by a size-n^a circuit $C\colon \{0,1\}^{n^a} \times [N] \to \{0,1\}$,* $\mathsf{Avoid}(\langle C\rangle)$ *solves* `Implicit-`δ`-Heavy-Avoid` *on G.*

Then there is an infinitely-often subexponential-time algorithm for* $\mathsf{GAP\text{-}SAT}$*. That is, for every $\epsilon > 0$ and $c \geq 1$, there exists a deterministic algorithm* Derand *running in 2^{n^ϵ} time such that the following holds: For every sequence of circuits $\{D_n\}_{n\in\mathbb{N}}$, where each $D_n : \{0,1\}^{n^c} \to \{0,1\}$ is a circuit of size $2n^c$,* $\mathsf{Derand}(1^n, \langle D_{1\sim\mathsf{poly}(n)}\rangle)$ *infinitely-often* solves* $\mathsf{GAP\text{-}SAT}$ *on $\{D_n\}$.*

Our proof combines the two instance-wise hardness-randomness tradeoffs introduced by Chen and Tell [CT21] and Liu and Pass [LP23] recently. Since the hardness-randomness tradeoff in [CT21] is already summarized in Theorem 5.1, in what follows we summarize the hardness-randomness tradeoff in [LP23].

Definition 5.9. Let $f\colon \{0,1\}^n \to \{0,1\}^{T(n)}$ be a function, A be a randomized algorithm, and $x \in \{0,1\}^n$ be an input of f. We say that $f(x)$ is *ℓ-leakage resilient hard* against A if for every "leakage string" $\mathsf{leak} \in \{0,1\}^\ell$, there is some $i \in [T]$ such that $\mathbf{Pr}[A(x, \mathsf{leak}, i) = f(x)_i] \leq 2/3$, where the probability is over the internal randomness of A.

We need the following result by Liu and Pass [LP23] showing that leakage resilient hardness can be used for derandomization.

Theorem 5.10 ([LP23])**.** *There are algorithms* $\mathsf{LP23.PRG}$ *and* $\mathsf{LP23.Recon}$ *and an absolute constant $c \geq 1$ such that the following holds. Let $f\colon \{0,1\}^n \to \{0,1\}^{T(n)}$ be a function, $D\colon \{0,1\}^M \to \{0,1\}$ be a distinguisher, and $x \in \{0,1\}^n$ be an input of f. Let $\ell := (M \log T)^c$ and $r := O(\log^2 T/\log M)$. Then:*

- $\mathsf{LP23.PRG}\colon \{0,1\}^T \times \{0,1\}^r \to \{0,1\}^M$ *runs in deterministic* $\mathsf{poly}(T, M)$ *time.*
- $\mathsf{LP23.Recon}^{(-)}\colon \{0,1\}^\ell \times [T] \to \{0,1\}$ *runs in randomized* $\mathsf{poly}(\ell, \log T)$*.*
- *If $f(x)$ is ℓ-leakage resilient hard against* $\mathsf{LP23.Recon}^D$*, then* $\mathsf{LP23.PRG}(f(x), -)$ *is a (targeted) PRG that $(1/10)$-fools D.*

Proof Sketch. This follows by observing that the leakage resilient hardness-randomness tradeoff in [LP23] holds *instance-wise*. In particular, if we let g be the "k-reconstructive PRG" described in [LP23, Theorem 3.11] (which follows from [STV01]), then $\mathsf{LP23.PRG}(f(x), z) = g^{f(x)}(1^{m(n)}, 1^{M(n)}, z)$; the algorithm $\mathsf{LP23.Recon}$ is simply the corresponding "reconstruction algorithm" R as defined in [LP23, Definition 3.10]. □

Suppose that $f\colon \{0,1\}^n \to \{0,1\}^{m(n)}$ is computable by a deterministic Turing machine M running in time $T(n)$. We can define $f^{\mathsf{hist}}\colon \{0,1\}^n \to \{0,1\}^{T'(n)}$ for some function $T'(n) \le \mathsf{poly}(T(n), m(n))$ such that $f^{\mathsf{hist}}(x)$ outputs the *computational history* of $f(x)$, i.e., the sequence of configurations of M when computing over the input string x. It will be useful to consider the leakage-resilience hardness of f^{hist}, since if f^{hist} is not leakage resilient hard, then f can be computed by a *low-depth* circuit:

Claim 5.11. *Let $f\colon \{0,1\}^n \to \{0,1\}^{T(n)}$ be a function computable in time $T(n)$, $\ell = \ell(n)$ be a parameter, and A be a randomized algorithm running in time $T_A(n)$. Then, for $d(n) := O(T_A(n) + \ell(n) + \log T(n))$, there is a logspace-uniform circuit $C\colon \{0,1\}^n \to \{0,1\}^{T(n)}$ of dimension $d(n) \times 2^{d(n)}$ such that the following holds. For every input $x \in \{0,1\}^n$, if $f^{\mathsf{hist}}(x)$ is* not *$\ell(n)$-leakage resilient hard against A, then $f(x) = C(x)$.*

Proof. The circuit $C(x)$ enumerates all strings $\mathsf{leak} \in \{0,1\}^\ell$ and computes $A(x, \mathsf{leak}, i)$ for each i. Although A is a randomized algorithm, we can compute $A(x, \mathsf{leak}, i)$ by brute force using a logspace-uniform circuit of width $2^{O(T_A(n))}$ and depth $O(T_A(n))$. Then, (for each leak) C verifies whether the computational history $H(x, \mathsf{leak})$ defined by $H(x, \mathsf{leak})_i = A(x, \mathsf{leak}, i)$ is indeed the correct history for $f(x)$; this can be done by a logspace-uniform circuit of width $\mathsf{poly}(T(n))$ and depth $O(\log T(n))$, by checking that every local step in $H(x, \mathsf{leak})$ is correct. Whenever there is some leak that corresponds to the correct history for $f(x)$, the circuit C outputs the value of $f(x)$ according to this history. The depth of C is $d(n) \le O(T_A(n) + \ell(n) + \log T(n))$ and the size of C is exponential in $d(n)$. □

Now we are ready to prove Theorem 5.8.

Proof of Theorem 5.8. Let $c \ge 1$ and $\epsilon > 0$ be constants. Let $\{D_n\}_{n\in\mathbb{N}}$ be a sequence of circuits where each $D_n\colon \{0,1\}^{n^c} \to \{0,1\}$ is of size $2n^c$. Recall that we want a deterministic 2^{n^ϵ}-time algorithm that infinitely-often* solves $\mathsf{GAP\text{-}SAT}$ on $\{D_n\}$. Let c_1 be the constant in Theorem 5.1 (the tradeoff in [CT21]) or Theorem 5.10 (the tradeoff in [LP23]), whichever is larger. Let $\kappa := \lceil \max\{4c/\epsilon, 8c_1/\epsilon\} \rceil$; the meaning of this constant is that we will perform a win-win analysis over input lengths m and m^κ. Finally, let $a := 4\kappa c c_1$ and $L(t) := 2^{t^{\epsilon/4}}$. The input instances of our Heavy Avoid algorithm will be generators $G_t\colon \{0,1\}^{t^a} \to \{0,1\}^{L(t)}$ whose output bits are implicitly computable in t^a size.

c, ϵ	We want a $\mathsf{GAP\text{-}SAT}$ algorithm on n^c-size circuits in 2^{n^ϵ} time
c_1	Overheads of the tradeoffs in Theorems 5.1 and 5.10
$\kappa = O(c/\epsilon)$	Our win-win analysis is over input lengths m and m^κ
$a = O(c^2/\epsilon)$	We need a Heavy Avoid algorithm for generators $G\colon \{0,1\}^{t^a} \to \{0,1\}^{L(t)}$, implicitly computable in t^a size
$L(t) = 2^{t^{\epsilon/4}}$	

Table 1: Constants used in this proof.

For any integer M, its *index* $idx(M)$ is defined as the largest integer such that $M = m^{\kappa^{idx(M)}}$ for some integer m. (Note that most integers have index 0.) We say an input length M is *big* if $idx(M)$ is odd, and is *small* if $idx(M)$ is even. For convenience, we will always use upper case M to denote big input lengths and lower case m to denote small input lengths. Looking ahead, each small input length m will be paired with a big input length m^κ and vice versa; if our Heavy Avoid algorithm succeeds on input length m, then our $\mathsf{Gap\text{-}SAT}$ algorithm succeeds on either length m or length m^κ.

We now define the sequence of implicit descriptions $C_t\colon \{0,1\}^{t^a} \times [L(t)] \to \{0,1\}$ for each $t \in \mathbb{N}$, which we feed into our Heavy Avoid algorithm. Each C_t also defines the generator $G_t : \{0,1\}^{t^a} \to \{0,1\}^{L(t)}$. Let M be the t-th smallest big input length. Note that $M \le O(t^\kappa)$. Let x denote the input of G_t. We parse x into $(\langle M_f\rangle, r)$, where $\langle M_f\rangle$ is the description of a Turing machine M_f and the remaining $M^{(2c+1)c_1}$ bits r are treated as randomness. As in the proof of Theorem 5.3, we encode x in such a way that every constant-length program M_f occurs with constant probability. Now, let $d := M^{2\epsilon/3}$ and f be the $d \times 2^d$ circuit outputted by M_f within space constraint d. Then

$$C_t(x,i) := \mathsf{CT21.Recon}_f^{D_M}(\langle D_M\rangle, i; r).$$

Recall that $\mathsf{CT21.Recon}$ uses at most $(d \cdot |\langle D_M\rangle|^2)^{c_1} \le (dM^{2c})^{c_1}$ random bits, hence we have enough random bits to feed into $\mathsf{CT21.Recon}$. We can see that each bit of G_t can be computed in $M^{3cc_1} \le t^{4\kappa cc_1} \le t^a$ size.

Let Avoid be our algorithm for $\texttt{Implicit-}\delta\texttt{-Heavy-Avoid}$ that runs in deterministic $\mathsf{poly}(L(t))$ time. Let I be the set of input lengths t for which $\mathsf{Avoid}(\langle C_t\rangle)$ correctly outputs a δ-light element of G_t. Using the same reasoning as Theorem 5.3, one can see that:

Claim 5.12. *Let $t \in I$, M be the t-th big input length, and let $d := M^{\epsilon/2}$. Suppose there is a constant-length Turing machine that outputs a circuit f' of dimension $d \times 2^d$ in $O(d)$ space such that $f'(\langle C_t\rangle) = \mathsf{Avoid}(\langle C_t\rangle)$. Then $\mathsf{CT21.HSG}_{f'}(\langle C_t\rangle)$ hits D_M.* ◇

Let $\mathsf{Avoid}^{\mathsf{hist}}$ denote the algorithm that outputs the computational history of Avoid. Note that $\mathsf{Avoid}^{\mathsf{hist}}(\langle C_t\rangle)$ runs in $T_{\mathsf{hist}}(t) \le \mathsf{poly}(L(t))$ time. Now let $m \in \mathbb{N}$ be the t-th small input length, $M := m^\kappa$ be the t-th big input length, and $\ell(t) := (m \cdot \log T_{\mathsf{hist}}(t))^{c_1}$. Consider the following criterion for our win-win analysis:

- ▶ $\underline{\mathsf{Crit}(t)\text{: } \mathsf{Avoid}^{\mathsf{hist}}(\langle C_t\rangle) \text{ is } \ell(t)\text{-leakage resilient hard against } \mathsf{LP23.Recon}^{D_m}.}$

Our algorithm Derand works as follows. On input $(1^n, \langle D_{1\sim n^\kappa}\rangle)$:

- Suppose n is small, $m := n$, and $M := n^\kappa$. Assume that m is the t-th small input length and assume that $\mathsf{Crit}(t)$ holds. Then by Theorem 5.10, $\mathsf{LP23.PRG}(\mathsf{Avoid}^{\mathsf{hist}}(\langle C_t\rangle), -)\colon \{0,1\}^r \to \{0,1\}^m$ is a PRG that (1/10)-fools D_m, where $r := O(\log^2 T_{\mathsf{hist}}(t)/\log m) \le O(t^{\epsilon/2}/\log m) < m^{\epsilon/2}$. By enumerating this PRG, we can solve the $\mathsf{GAP\text{-}SAT}$ (in fact, CAPP) problem on input D_m in deterministic 2^{m^ϵ} time.

  Note: The definition of C_t involves $D_M = D_{n^\kappa}$; this is why we need our infinitely-often* algorithm to have access to inputs on a larger length.

- Suppose n is big, $M := n$, and $m := n^{1/\kappa}$. Suppose that M is the t-th big input length and assume that $\mathsf{Crit}(t)$ does not hold. Then it follows from Claim 5.11 that there is a logspace-uniform circuit $\widetilde{\mathsf{Avoid}}$ of dimension $d' \times T'$ such that $\widetilde{\mathsf{Avoid}}(\langle C_t\rangle) = \mathsf{Avoid}(\langle C_t\rangle)$, where $d' = O(m^c + \ell(m) + \log T_{\mathsf{hist}}(t)) \le O(m^c + (mt^{\epsilon/4})^{c_1}) < M^{\epsilon/2}$ and $T' = 2^{d'}$. By Claim 5.12, $\mathsf{CT21.HSG}_{\widetilde{\mathsf{Avoid}}}(\langle C_t\rangle)$ hits D_M. Since the size of this HSG is $2^{O(d')} < 2^{M^{0.9\epsilon}}$, we can solve the $\mathsf{GAP\text{-}SAT}$ problem on D_M in 2^{M^ϵ} time.

For every $t \in I$, let m be the t-th small input length and $M = m^\kappa$. If $\mathsf{Crit}(t)$ holds, then our algorithm solves $\mathsf{GAP\text{-}SAT}$ on input D_m; otherwise our algorithm solves $\mathsf{GAP\text{-}SAT}$ on input D_M. Since $|I|$ is infinite, it follows that our algorithm infinitely-often* solves $\mathsf{GAP\text{-}SAT}$ on $\{D_n\}$. □

5.3 On Black-Box Reductions to Heavy-Avoid

We complement the previous non-black-box reductions by showing that if there is a black-box reduction from $\mathsf{GAP\text{-}SAT}$ to $\texttt{Heavy-Avoid}$ of a certain type, then we would have $\mathsf{prBPP} = \mathsf{prP}$ *unconditionally*. In more detail, we consider the natural notion of *Levin reductions* [Lev73], i.e., "witness-mapping" reductions between search problems, and show that a Levin reduction from $\mathsf{search\text{-}GAP\text{-}SAT}$ to $\texttt{Heavy-Avoid}$ implies $\mathsf{prBPP} = \mathsf{prP}$.

Intriguingly, these results together *separate* the notion of (weak) non-black-box reductions and black-box (i.e., Levin) reductions between two natural problems w.r.t. current techniques! That is, improving the weak non-black-box reductions in Theorems 5.3 and 5.8 to Levin reductions (which is a stronger notion of black-box reduction) would imply breakthroughs in complexity theory.

As the notion of Levin reductions is standard in complexity theory (for a recent example, see [MP24]), we only recall its definition in the special case of reducing $\mathsf{search\text{-}GAP\text{-}SAT}$ to $\texttt{Heavy-Avoid}$:

Definition 5.13. We say there is a *Levin reduction* from $\mathsf{search\text{-}GAP\text{-}SAT}$ to $\texttt{Heavy-Avoid}$ if there are functions f, g computable in deterministic polynomial time such that the following holds:

- For every circuit $C : \{0,1\}^n \to \{0,1\}$ that is $1/2$-dense (i.e., that is a valid $\mathsf{search\text{-}GAP\text{-}SAT}$ instance), $f(C) = (D, 1^t)$ is a $\texttt{Heavy-Avoid}$ instance where we want to find a $(1/t)$-light string in D.
- For every string y that is $(1/t)$-light for D, $g(C, y)$ is a valid solution for $\mathsf{search\text{-}GAP\text{-}SAT}$ for C. That is, if C is $1/2$-dense, then $C(g(C,y)) = 1$.

Theorem 5.14. *If there is a polynomial-time Levin reduction (f, g) from* $\mathsf{search\text{-}GAP\text{-}SAT}$ *to* $\texttt{Heavy-Avoid}$*, then* $\mathsf{prP} = \mathsf{prBPP}$*.*

Proof. First, we describe the main idea. The crucial observation is that there is a trivial algorithm that "*list-solves*" any $\texttt{Heavy-Avoid}$ instance, that is, outputs a list of solutions such that some element in the list is not heavy. In particular, let $(D, 1^t)$ be an input instance of $\texttt{Heavy-Avoid}$, and consider the trivial algorithm that outputs an arbitrary list of $t+1$ distinct strings. By an averaging argument, at least one string in this list will be a $(1/t)$-light element of D. This means that using the list and a witness-mapping reduction we can produce at least one satisfying assignment if the input circuit is dense.

We proceed to the formal proof. Again, by [BF99], it suffices to prove that $\mathsf{prP} = \mathsf{prRP}$. Let $C : \{0,1\}^n \to \{0,1\}$ be an input to $\mathsf{GAP\text{-}SAT}$. We first reduce C to a $\texttt{Heavy-Avoid}$ instance $(D, 1^t) := f(C)$. Then we compute an arbitrary list of $t+1$ distinct strings $x_1, x_2, \ldots, x_{t+1}$. We are guaranteed that some x_i is a δ-light element of D, hence if C is $1/2$-dense, then $g(C, x_i)$ would be a satisfying assignment of C. Consequently, if there is an index $i \in [\ell]$ such that $C(g(C, x_i)) = 1$, then we output 1, otherwise we output 0. It is easy to see that if C is unsatisfiable then we always output 0, while if C is $1/2$-dense then we always output 1. Hence we have $\mathsf{prP} = \mathsf{prRP}$ and this concludes the proof. $\square$

5.4 Heavy Avoid versus Almost-All-Inputs Hardness

Finally, we show connections between $\texttt{Implicit-Heavy-Avoid}$ and the *almost-all-inputs* hardness assumptions, introduced recently in [CT21].

The results in this section are motivated by two conjectures. Given the non-black-box reductions in Theorems 5.3 and 5.8, it seems natural to conjecture that $\texttt{Implicit-Heavy-Avoid}$ is complete for prBPP under "the most natural notion of non-black-box reductions":

Conjecture 5.15 (Informal)**.** *If* `Implicit-Heavy-Avoid` *for non-uniform samplers admits a deterministic polynomial-time algorithm, then* $\mathsf{prBPP} = \mathsf{prP}$*.*

On the other hand, there is another intriguing conjecture implicit in the work of Chen–Tell [CT21]. Recall that Chen and Tell [CT21] showed how to derive $\mathsf{prBPP} = \mathsf{prP}$ given a multi-output function $f\colon \{0,1\}^n \to \{0,1\}^n$ that is *almost-all-inputs* hard against randomized algorithms, *provided that f can be computed in low depth.* They also showed that $\mathsf{prBPP} = \mathsf{prP}$ *necessitates* the existence of multi-output function with almost-all-inputs hardness, but the hard function they construct might require high depth. Still, this demonstrates that almost-all-inputs hardness might be *the right hardness assumption* for derandomization. It is tempting to conjecture that low-depth constraints are, in fact, not necessary:

Conjecture 5.16 (Informal)**.** *If there is a multi-output function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in deterministic polynomial time that is almost-all-inputs hard against fixed-polynomial time randomized algorithms, then* $\mathsf{prBPP} = \mathsf{prP}$*.*

In this section, we show that in an "implicit" setting, Conjecture 5.15 and Conjecture 5.16 are equivalent! In fact, we show that `Implicit-Heavy-Avoid` is the computational problem characterizing the task of "creating" almost-all-inputs hardness (against randomized algorithms). We also show that creating such hardness is equivalent to generating strings with high conditional sublinear-time probabilistic Kolmogorov complexity.

Our "implicit" setting. We consider functions with a possibly long output, i.e., $f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$ where $\ell(n)$ is larger than the running time of our adversaries. Consequently, it makes sense to only require our adversaries to output each bit of $f(x)$ given x and the index of that bit. It is worth noting that the hardness-randomness tradeoffs in both [CT21] and [LP23] holds for such implicit adversaries.

Formally, let $\mathcal{A}$ be a randomized algorithm, we say that $\mathcal{A}$ *locally* computes $f(x)$ if for every $i \in [\ell(n)]$, it holds that $\mathbf{Pr}[\mathcal{A}(x,i) = f(x)_i] \geq 2/3$, where the probability is over the internal randomness of $\mathcal{A}$. An equivalent way of saying this is that $f(x)$ is not 0-leakage resilient hard against $\mathcal{A}$ in the sense of Definition 5.9.

We also consider the task of generating strings with high (conditional) *sublinear-time* probabilistic Kolmogorov complexity. (Our equivalence results hold for both $\mathsf{pK}^{\mathsf{poly}}$ and $\mathsf{rK}^{\mathsf{poly}}$.) Roughly speaking, fix a universal Turing machine U, a time bound t, and strings x, y, where $|x| \ll t \ll |y|$. The conditional complexity of y given x is the length of the shortest program p such that $U(p, w, x, i)$ outputs the i-th bit of y in t steps, with w being the randomness. However, there is a technical detail that is worth stressing: In our definition, we require the resources (x and r) to have length at most t, hence the universal Turing machine U has time to read them in their entirety. A possible alternative definition would be that U has *oracle access* to strings x and r (whose lengths might be $\gg t$), but it is unclear if we can extend our equivalence result (Theorem 5.18) to these alternative definitions.

Definition 5.17 (Sublinear-time probabilistic Kolmogorov complexity)**.** Let U be a universal Turing machine, $x, y \in \{0,1\}^*$, and $t \in \mathbb{N}$. (Think of $|x| \ll t$ and $|y| \gg t$.) We artificially define $y_{|y|+1} = \star$.

- **Sublinear-time pK^t complexity:**

$$\mathsf{pK}^t_U(y \mid x) := \min\left\{k \in \mathbb{N} \;\middle|\; \mathop{\mathbf{Pr}}_{w \sim \{0,1\}^t}\left[\exists p \in \{0,1\}^k, \forall i \in [|y|+1], U(1^t, p, w, x, i) = y_i\right] \geq \frac{2}{3}\right\}.$$

In other words, if $k = \mathsf{pK}^t_U(y \mid x)$, then with probability at least 2/3 over the choice of the *length-t* random string w, given w and x, the string y admits a t-time-bounded *local* encoding of length k.

- **Sublinear-time rK^t complexity:**

$$\mathsf{rK}^t_U(y \mid x) = \min_{p \in \{0,1\}^*} \left\{ |p| \;\middle|\; \forall i \in [|y|+1], \Pr_{w \sim \{0,1\}^t}[U(1^t, p, w, x, i) = y_i] \geq \frac{2}{3} \right\}.$$

Now we are ready to present our equivalence result.

Theorem 5.18. *The following are equivalent.*

(1) **(Almost-all-inputs hardness.)** *For every polynomial $p(\cdot)$, there exists a polynomial $q(\cdot)$ and a function $f\colon \{0,1\}^n \to \{0,1\}^{q(n)}$ computable by a deterministic polynomial-time algorithm, such that every algorithm running in randomized $p(n)$ time only locally computes $f(x)$ on finitely many inputs $x \in \{0,1\}^*$.*

(2) **(Deterministic algorithms for `Heavy-Avoid`.)** *For every polynomial $p(\cdot)$, there exists a polynomial $q(\cdot)$ such that the following holds. Let $\mathcal{C}$ denote the class of circuits with n input bits and $q(n)$ output bits where each output bit is implicitly computed by a size-$p(n)$ circuit, then $\mathcal{C}$-`Implicit`-$(1/p(n))$-`Heavy-Avoid` can be solved in deterministic polynomial time.*

(3) **(Finding strings with large conditional $\mathsf{pK}^{\mathsf{poly}}$-complexity.)** *For every polynomial $p(\cdot)$, there exists a polynomial $q(\cdot)$ and a deterministic polynomial-time algorithm that given an input $x \in \{0,1\}^n$, finds a string $y \in \{0,1\}^{q(n)}$ such that $\mathsf{pK}^{p(n)}(y \mid x) \geq \log p(n)$.*

(4) **(Finding strings with large conditional $\mathsf{rK}^{\mathsf{poly}}$-complexity.)** *For every polynomial $p(\cdot)$, there exists a polynomial $q(\cdot)$ and a deterministic polynomial-time algorithm that given an input $x \in \{0,1\}^n$, finds a string $y \in \{0,1\}^{q(n)}$ such that $\mathsf{rK}^{p(n)}(y \mid x) \geq \log p(n)$.*

Arguably, the most interesting implication above is (1) $\Rightarrow$ (2), which can be interpreted as *instance-wise hardness vs. randomness* for solving `Implicit-Heavy-Avoid` without depth constraints.

Lemma 5.19. *In Theorem 5.18, (1) $\Rightarrow$ (2) holds. That is, almost-all-inputs hardness (of any kind, without depth restrictions) can be used to solve* `Implicit-Heavy-Avoid`*.*

Proof. Let $T(n)$ and $q'(n)$ be two polynomials such that there is a multi-output function $f\colon \{0,1\}^n \to \{0,1\}^{q'(n)}$ computable in deterministic $T(n)$ time that is almost-all-inputs hard against randomized algorithms running in time $p(n)^6$. Let $q(n) := \mathsf{poly}(T(n), q'(n))$, and $f_{\mathsf{PCP}}\colon \{0,1\}^n \to \{0,1\}^{q(n)}$ denote the PCP of f: On input $x \in \{0,1\}^n$, $f_{\mathsf{PCP}}(x)$ outputs the concatenation of strings z, pcp_1, $\mathsf{pcp}_2, \ldots, \mathsf{pcp}_{q'(n)}$, where $z = f(x)$ and each pcp_i is a length-$\mathsf{poly}(T(n))$ PCP proof for the assertion that the i-th output bit of $f(x)$ is equal to z_i. (Any efficiently-computable PCP with polynomial length and constant query complexity works here, e.g., [ALM+98, BGH+06, Din07].) We claim that f_{PCP} is an algorithm that solves $\mathcal{C}$-`Implicit`-$(1/p(n))$-`Heavy-Avoid`.

Suppose, towards a contradiction, that f_{PCP} does not solve $\mathcal{C}$-`Implicit`-$(1/p(n))$-`Heavy-Avoid` on an input $\langle C \rangle \in \{0,1\}^\ell$. The input $\langle C \rangle$ encodes a circuit C of size $p(n)$ (hence $\ell = \tilde{O}(p(n))$) that implicitly represents a generator $G\colon \{0,1\}^n \to \{0,1\}^{q(n)}$. Given $\langle C \rangle$, $r \in \{0,1\}^n$, and $i \in [q(n)]$, the i-th output bit of $G(r)$ can be computed in $\tilde{O}(p(n))$ time. Since f_{PCP} fails on $\langle C \rangle$, we have

$$\Pr_{r \sim \{0,1\}^n}[f_{\mathsf{PCP}}(\langle C \rangle) = G(r)] \geq 1/p(n). \tag{2}$$

Now we present a randomized algorithm $\mathcal{A}$ running in $p(n)^5$ time that locally computes f on input $\langle C \rangle$. Given an integer i, we want to compute the i-th bit of $f(\langle C \rangle)$. We repeat the following $O(p(n)^2)$ times:

1. Sample a random string $r \sim \{0,1\}^n$.
2. Parse $G(r)$ as the concatenation of $z, \mathsf{pcp}_1, \mathsf{pcp}_2, \dots, \mathsf{pcp}_{q'(n)}$.
3. Invoke the PCP verifier $O(p(n))$ times to verify that pcp_i is indeed a correct PCP proof that $f(\langle C \rangle)_i = z_i$.
4. If all invocations of the PCP verifier are successful, then we output z_i and halt.

If we have not outputted anything after these $O(p(n)^2)$ iterations, then we output a random bit.

Let $\mathcal{A}$ denote the above randomized algorithm. We now analyze $\mathcal{A}$.

- **(Running time.)** Since each bit of pcp_i can be retrieved in $\tilde{O}(p(n))$ time, Step 3 above takes $\tilde{O}(p(n)^2)$ time, hence the whole algorithm runs in at most $\tilde{O}(p(n)^4) \le p(n)^5$ time.
- **("Soundness.")** At each iteration where we parse $G(r)$ as $z, \mathsf{pcp}_1, \dots, \mathsf{pcp}_{q(n)}$, if $f(\langle C \rangle)_i \neq z_i$, then no matter what pcp_i is, the PCP verifier will catch an error with probability at least $1 - \exp(-p(n))$. Hence, the probability that $\mathcal{A}(\langle C \rangle, i)$ halts in Step 4 above and outputs $1 - f(\langle C \rangle)_i$ is at most $\exp(-p(n))$.
- **("Completeness.")** On the other hand, by Eq. (2), with probability at least $1 - (1 - 1/p(n))^{p(n)^2} \ge 1 - \exp(-p(n))$, there is some iteration of the above algorithm in which $G(r) = f_{\mathsf{PCP}}(\langle C \rangle)$. During this iteration, it will be the case that $z_i = f(\langle C \rangle)_i$ and pcp_i is a valid PCP proof for this, therefore we will output $f(\langle C \rangle)_i$ and halt. It follows that the probability that none of the $p(n)^2$ iterations succeed and we output a random bit at the end is also upper bounded by $\exp(-p(n))$.

The "Soundness" and "Completeness" above imply that $\mathcal{A}$ locally computes $f(\langle C \rangle)$. To conclude, if f is indeed almost-all-inputs hard against randomized algorithms running in time $p(n)^6$, then f_{PCP} solves the $\mathcal{C}$-$\texttt{Implicit}$-$(1/p(n))$-$\texttt{Heavy-Avoid}$ problem on all but finitely many inputs. $\square$

Now we present the complete proof for Theorem 5.18.

Proof of Theorem 5.18. We consider each implication below.

(1) $\Rightarrow$ (2): This follows from Lemma 5.19.

(2) $\Rightarrow$ (3): Let $p(n)$ be a polynomial and $p'(n) := p(n)^2$. By (2), for some polynomial $q(n)$, there is a polynomial-time algorithm Avoid solving the $\mathcal{C}$-$\texttt{Implicit}$-$(1/p'(n))$-$\texttt{Heavy-Avoid}$ problem, where the generators have output length $q(n)$ and each bit can be computed in size $p'(n)$. Let U be a universal Turing machine and suppose that given input $x \in \{0,1\}^n$, we want to find a string $y \in \{0,1\}^{q(n)}$ such that $\mathsf{pK}_U^{p(n)}(y \mid x) \ge \log p(n)$.

Consider the generator $G_x : \{0,1\}^{p(n)+\log p(n)} \to \{0,1\}^{q(n)}$ that is implicitly computed by the following circuit C_x: given (z,i) as input, where $z \in \{0,1\}^{p(n)+\log p(n)}$ and $i \in [q(n)]$, $C_x(z,i)$ outputs the i-th bit of $G(z)$. We parse z into $(\langle M \rangle, r)$, where M is a Turing machine of description length $\log p(n)$, and $r \in \{0,1\}^{p(n)}$ is treated as randomness. Then, $C_x(z,i)$ outputs $U(1^{p(n)}, \langle M \rangle, r, x, i)$. Clearly, C_x can be implemented by a circuit of size $\tilde{O}(p(n)) < p'(n)$.

Let $y \in \{0,1\}^{q(n)}$ be any string that is $\frac{1}{p'(n)}$-light for G_x, we claim that $\mathsf{pK}_U^{p(n)}(y \mid x) \geq \log p(n)$. This is easily shown by contradiction. Suppose that $\mathsf{pK}_U^{p(n)}(y \mid x) < \log p(n)$, then w.p. at least 2/3 over $r \sim \{0,1\}^{p(n)}$, there is a program M of description length $\log p(n)$ such that for every $i \in [n+1]$, $U(1^{p(n)}, \langle M \rangle, r, x, i) = y_i$; in other words, $G_x(\langle M \rangle, r) = y$. This contradicts our assumption that y is $\frac{1}{p'(n)}$-light for G_x.

Hence, solving the $\mathcal{C}$-`Implicit`-$(1/p'(n))$-`Heavy-Avoid` problem on G_x will give us a string $y \in \{0,1\}^{q(n)}$ such that $\mathsf{pK}_U^{p(n)}(y \mid x) \geq \log p(n)$.

(3) $\Rightarrow$ (4): This follows from the fact ([LO22, Fact 2]) that for every universal Turing machine U, every $x, y \in \{0,1\}^*$ and every time bound t, we have $\mathsf{pK}_U^t(y \mid x) \leq \mathsf{rK}_U^t(y \mid x)$. It is easy to verify that this is still true with respect to our notions of sublinear-time-bounded Kolmogorov complexity.

(4) $\Rightarrow$ (1): Let $q(n)$ be any polynomial, $f\colon \{0,1\}^n \to \{0,1\}^{q(n)}$ be any function, and $\mathcal{A}$ be a randomized algorithm running in $p(n)$ time. We claim that for every input $x \in \{0,1\}^n$, if $\mathsf{rK}^{p(n)^2}(f(x) \mid x) > |\mathcal{A}| + \omega(1)$, then $\mathcal{A}$ fails to locally compute $f(x)$. Indeed, if $\mathcal{A}$ locally computes $f(x)$, then

$$\forall i \in [|x|+1], \Pr_{r \sim \{0,1\}^{p(n)}}[\mathcal{A}(x, i; r) = U(1^{p(n)}, \langle \mathcal{A} \rangle, r, x, i) = f(x)_i] \geq 2/3.$$

Clearly, this means that $\mathsf{rK}^{p(n)^2}(f(x) \mid x) \leq |\mathcal{A}| + O(1)$.

Suppose that f is a deterministic polynomial-time algorithm that given an input $x \in \{0,1\}^n$, outputs a string $y \in \{0,1\}^{q(n)}$ such that $\mathsf{rK}^{p(n)^2}(y \mid x) \geq 2 \log p(n)$. It follows directly that every randomized algorithm running in $p(n)$ time only locally computes $f(x)$ for finitely many inputs $x \in \{0,1\}^*$ (as long as the algorithm admits a constant-size description). Hence f is almost-all-inputs hard against $p(n)$-time randomized algorithms. $\square$

Given the above equivalence, it is easy to see that Conjectures 5.15 and 5.16 are equivalent, since Conjecture 5.15 asserts the equivalence between (2) and $\mathsf{prBPP} = \mathsf{prP}$, while Conjecture 5.16 asserts the equivalence between (1) and $\mathsf{prBPP} = \mathsf{prP}$.

References

[Adl78] Leonard M. Adleman. Two theorems on random polynomial time. In *Symposium on Foundations of Computer Science* (FOCS), pages 75–83, 1978. 3

[Ajt90] Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990*, pages 1–20, 1990. 19, 22, 23

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. 43

[ALR99] Eric Allender, Michael C. Loui, and Kenneth W. Regan. Reducibility and completeness. In *Algorithms and Theory of Computation Handbook*, Chapman & Hall/CRC Applied Algorithms and Data Structures series. CRC Press, 1999. 55

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 194–208, 2002. 5, 33

[BCGT13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In *Symposium on Theory of Computing* (STOC), pages 585–594. ACM, 2013. 29

[BF99] Harry Buhrman and Lance Fortnow. One-sided versus two-sided error in probabilistic computation. In *Symposium on Theoretical Aspects of Computer Science* (STACS), pages 100–109, 1999. 9, 35, 36, 41

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.*, 1:3–40, 1991. 25, 28

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006. 29, 43

[BH92] Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 116–127, 1992. 25, 26

[BI94] David A. Mix Barrington and Neil Immerman. Time, hardware, and uniformity. In *Structure in Complexity Theory Conference* (CCC), pages 176–185, 1994. 56

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990. 9, 23, 24

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995. 18

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008. 29

[Bus87] Samuel R. Buss. The Boolean formula value problem is in ALOGTIME. In *Symposium on Theory of Computing* (STOC), pages 123–131, 1987. 18

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 163–173, 2014. 29

[BW05] Amos Beimel and Enav Weinreb. Monotone circuits for weighted threshold functions. In *Conference on Computational Complexity* (CCC), pages 67–75, 2005. 18

[CDM23] Arkadev Chattopadhyay, Yogesh Dahiya, and Meena Mahajan. Query complexity of search problems. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), volume 272, pages 34:1–34:15, 2023. 12

[Che23] Lijie Chen. New lower bounds and derandomization for ACC, and a derandomization-centric view on the algorithmic method. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 34:1–34:15, 2023. 7, 8, 18, 29, 52, 53, 54, 55, 56

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *Symposium on Theory of Computing* (STOC), pages 1058–1066, 2023. 12

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *Symposium on Theory of Computing* (STOC), pages 1990–1999, 2024. 12

[CL24] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In *Symposium on Theory of Computing* (STOC), pages 620–629, 2024. 12

[CLO$^+$23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *Symposium on Foundations of Computer Science* (FOCS), pages 1261–1270, 2023. 7, 9, 10, 13, 35

[CLW20] Lijie Chen, Xin Lyu, and Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *Symposium on Foundations of Computer Science* (FOCS), pages 1–12, 2020. 3, 11

[CRT22] 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, 2022. 13, 37

[CRTY23] Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. On exponential-time hypotheses, derandomization, and circuit lower bounds. *J. ACM*, 70(4):25:1–25:62, 2023. 13

[CT19] Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits "just beyond" known lower bounds. In *Symposium on Theory of Computing* (STOC), pages 34–41, 2019. 12

[CT21] 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, 2021. 1, 6, 7, 9, 11, 13, 17, 34, 35, 38, 39, 41, 42

[CT23] Lijie Chen and Roei Tell. Guest column: New ways of studying the BPP = P conjecture. *SIGACT News*, 54(2):44–69, 2023. 13

[CTW23] Lijie Chen, Roei Tell, and Ryan Williams. Derandomization vs refutation: A unified framework for characterizing derandomization. In *Symposium on Foundations of Computer Science* (FOCS), pages 1008–1047, 2023. 6, 13, 34

[CW19] Lijie Chen and Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via PCPs of proximity. In *Computational Complexity Conference* (CCC), pages 19:1–19:43, 2019. 11

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. 43

[DPV18] Peter Dixon, Aduri Pavan, and N. V. Vinodchandran. On pseudodeterministic approximation algorithms. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 61:1–61:11, 2018. 12

[DPV21] Peter Dixon, Aduri Pavan, and N. V. Vinodchandran. Complete problems for multi-pseudodeterministic computations. In *Innovations in Theoretical Computer Science* (ITCS), 2021. 12

[DPWV22] Peter Dixon, Aduri Pavan, Jason Vander Woude, and N. V. Vinodchandran. Pseudodeterminism: promises and lowerbounds. In *Symposium on Theory of Computing* (STOC), pages 1552–1565, 2022. 12

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inf. Control.*, 61(2):159–173, 1984. 17

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Symposium on Foundations of Computer Science* (FOCS), pages 316–324, 2004. 5, 33

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Symposium on Theory of Computing* (STOC), pages 348–355, 2005. 5

[GG11] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, TR11-136, 2011. 1, 12, 13

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 65:1–65:18, 2023. 12

[GGR13] Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Innovations in Theoretical Computer Science* (ITCS), pages 127–138, 2013. 12

[GIPS21] Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. On the pseudodeterministic query complexity of NP search problems. In *Computational Complexity Conference* (CCC), pages 36:1–36:22, 2021. 12

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), pages 16:1–16:60, 2022. 31

[GL19] Ofer Grossman and Yang P. Liu. Reproducibility and pseudo-determinism in Log-Space. In *Symposium on Discrete Algorithms* (SODA), pages 606–620, 2019. 12

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 20:1–20:21, 2022. 12

[Gol11] Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 191–232. Springer, 2011. 13, 17, 26

[GW14] Oded Goldreich and Avi Wigderson. On derandomizing algorithms that err extremely rarely. In *Symposium on Theory of Computing* (STOC), pages 109–118, 2014. 12

[Hir15] Shuichi Hirahara. Identifying an honest $\mathrm{EXP}^{\mathrm{NP}}$ oracle among many. In *Computational Complexity Conference* (CCC), pages 244–263, 2015. 9, 25, 27, 28, 29

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Symposium on Theoretical Aspects of Computer Science* (STACS), pages 672–683, 2006. 53

[IKW02] 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. 4

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Symposium on Theory of Computing* (STOC), pages 1076–1089, 2023. 12

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing* (STOC), pages 220–229. ACM, 1997. 3, 6, 13, 51

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. 4, 11, 13, 37

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Log.*, 129(1-3):1–37, 2004. 12

[Jeř05] Emil Jeřábek. *Weak pigeonhole principle and randomized computation.* PhD thesis, Charles University in Prague, 2005. 12

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 44:1–44:18, 2021. 12

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Symposium on Theory of Computing* (STOC), pages 302–309, 1980. 26

[Kor21] Oliver Korten. The hardest explicit construction. In *Symposium on Foundations of Computer Science* (FOCS), pages 433–444, 2021. 4, 12, 51

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *Computational Complexity Conference* (CCC), volume 234, pages 37:1–37:26, 2022. 34, 36

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995. 12

[Kra24] Jan Krajícek. Proof complexity generators. Monograph (In Progress), 2024. 12

[KV87] Marek Karpinski and Rutger Verbeek. On the Monte Carlo space constructible functions and seperation results for probabilistic complexity classes. *Inf. Comput.*, 75(2):178–189, 1987. 5, 33

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. 6, 41

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *Symposium on Theory of Computing* (STOC), pages 2000–2007, 2024. 12

[LO21] Zhenjian Lu and Igor C. Oliveira. An efficient coding theorem via probabilistic representations and its applications. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 94:1–94:20, 2021. 7, 9, 31

[LO22] Zhenjian Lu and Igor C. Oliveira. Theory and applications of probabilistic Kolmogorov complexity. *Bull. EATCS*, 137, 2022. 7, 16, 45

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *ACM Symposium on Theory of Computing* (STOC), pages 303–316, 2021. 9, 12, 32

[LOZ22] Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 92:1–92:14, 2022. 7, 31

[LP22] Yanyi Liu and Rafael Pass. Characterizing derandomization through hardness of Levin-Kolmogorov complexity. In *Computational Complexity Conference* (CCC), pages 35:1–35:17, 2022. 6, 13, 17, 34

[LP23] Yanyi Liu and Rafael Pass. Leakage-resilient hardness vs randomness. In *Computational Complexity Conference* (CCC), pages 32:1–32:20, 2023. 6, 7, 11, 13, 17, 34, 38, 39, 42

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition.* Texts in Computer Science. Springer, 2019. 16

[MP24] Noam Mazor and Rafael Pass. Gap MCSP is not (Levin) NP-complete in Obfustopia. In *Computational Complexity Conference* (CCC), 2024. to appear. 41

[MW20] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020. 11

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. 3, 6, 13, 51

[Oli19] Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019. 7, 9, 31

[OS17] Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017. 9, 12

[OS18] Igor C. Oliveira and Rahul Santhanam. Pseudo-derandomizing learning and approximation. In *International Conference on Randomization and Computation* (RANDOM), pages 55:1–55:19, 2018. 12

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *Symposium on Foundations of Computer Science* (FOCS), pages 640–650, 2022. 4, 12, 51

[San23] Rahul Santhanam. An algorithmic approach to uniform lower bounds. In *Computational Complexity Conference* (CCC), pages 35:1–35:26, 2023. 12

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. 6, 38

[Tel17] Roei Tell. Improved bounds for quantified derandomization of constant-depth circuits and polynomials. In *Computational Complexity Conference* (CCC), pages 13:1–13:48, 2017. 12

[Tel18] Roei Tell. Quantified derandomization of linear threshold circuits. In *Symposium on Theory of Computing* (STOC), pages 855–865, 2018. 12

[Tel22] Roei Tell. Quantified derandomization: How to find water in the ocean. *Found. Trends Theor. Comput. Sci.*, 15(1):1–125, 2022. 12

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Comput. Complex.*, 16(4):331–364, 2007. 13, 18

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. 13

[Vio09] Emanuele Viola. On approximate majority and probabilistic time. *Comput. Complex.*, 18(3):337–375, 2009. 19, 22, 23

[vMP06] Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy for semantic models with one bit of advice. In *Conference on Computational Complexity* (CCC), pages 129–144, 2006. 5, 33

[vMS23] Dieter van Melkebeek and Nicollas M. Sdroievski. Instance-wise hardness versus randomness tradeoffs for Arthur-Merlin protocols. In *Computational Complexity Conference* (CCC), pages 17:1–17:36, 2023. 13

[VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. 9, 22, 24

[Wil13a] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. 11

[Wil13b] Ryan Williams. Towards NEXP versus BPP? In *International Computer Science Symposium in Russia* (CSR), pages 174–182, 2013. 3, 11

[Wil14] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. 3, 11

[Wil16] Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. 4

[Wil18a] Ryan Williams. Limits on representing Boolean functions by linear combinations of simple functions: Thresholds, ReLUs, and low-degree polynomials. In *Computational Complexity Conference* (CCC), pages 6:1–6:24, 2018. 11

[Wil18b] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory Comput.*, 14(1):1–25, 2018. 11

[Wil19] Ryan Williams. Some estimated likelihoods for computational complexity. In *Computing and Software Science - State of the Art and Perspectives*, pages 9–26. Springer, 2019. 3

A A Reduction to Range Avoidance

Recall that in the range avoidance problem (referred to as Avoid), we are given an input Boolean circuit $F\colon \{0,1\}^m \to \{0,1\}^n$, with $m < n$, and the goal is to output a string $y \in \{0,1\}^n$ such that $y \notin \mathsf{Range}(F) = \{F(x) \mid x \in \{0,1\}^m\}$ (see, e.g., [Kor21, RSW22][27]).

Let $C\colon \{0,1\}^m \to \{0,1\}^n$ be a sampler, where m and n are arbitrary. Note that non-uniform heavy avoid for $0 < \delta < 2^{-m}$ is precisely the range avoidance problem, since a string is not in the range of the map C if and only if its probability under D_C is strictly less than 2^{-m}. In this work, we are mostly concerned with the regime where $\delta = 1/\mathsf{poly}(n)$, i.e., when the goal is to avoid heavy elements. If $n > m$, then it is trivial to reduce `Heavy-Avoid` to Avoid; however, the goal of this section is to show that even in the regime that $n \leq m$, `Heavy-Avoid` and `Heavy-Find` are not harder than range avoidance.

Theorem A.1. *For any constant $u \geq 1$, there is a deterministic polynomial-time algorithm that, given oracle access to* Avoid*, an input circuit $C\colon \{0,1\}^m \to \{0,1\}^n$ of size at most n^u, and parameters $\delta > \varepsilon > 1/n^u$, outputs a pair (α, β) of n-bit strings such that the following hold:*

- *If there is a string $z \in \{0,1\}^n$ that is δ-heavy under D_C, then α is $(\delta - \varepsilon)$-heavy under D_C.*
- *If there is a string $z \in \{0,1\}^n$ that is not $(\delta - \varepsilon)$-heavy under D_C, then β is not δ-heavy under D_C.*

Proof. In order to prove Theorem A.1, we rely on the fact that a query to Avoid allows us to obtain the description of a Boolean function $h\colon \{0,1\}^{\log n} \to \{0,1\}$ of circuit complexity $\Omega(n/\log n)$ (see, e.g., [Kor21]). In turn, this implies that, for every constant $k \geq 1$ and $\varepsilon \geq 1/\mathsf{poly}(n)$, we can compute in time $\mathsf{poly}(n, 1/\varepsilon)$ a pseudorandom generator $G\colon \{0,1\}^{O(\log n)} \to \{0,1\}^n$ that ε-fools circuits of size n^k [NW94, IW97]. Next, we observe that this is sufficient to solve in polynomial time the non-uniform variants of both `Heavy-Avoid` and `Heavy-Find`.

We will need the following lemma.

Lemma A.2. *There is a constant $b \geq 1$ such that the following holds. Let $C\colon \{0,1\}^m \to \{0,1\}^n$ be a map computed by a circuit of size $s \geq n$. Let $G\colon \{0,1\}^\ell \to \{0,1\}^m$ be a pseudorandom generator that ε-fools any circuit $E\colon \{0,1\}^m \to \{0,1\}$ of size $b \cdot s$. Consider any string $z \in \{0,1\}^n$, and let p_z be its probability under D_C. Then*

$$p_z - \varepsilon \leq \Pr_{w \sim \{0,1\}^\ell}[C(G(w)) = z] \leq p_z + \varepsilon.$$

[27]This problem was called "Empty" in [Kor21].

Proof. For any fixed string $z \in \{0,1\}^n$, we consider the non-uniform circuit $E_z \colon \{0,1\}^m \to \{0,1\}$ that on an input x outputs 1 if and only if $C(x) = z$. Note that $\mathbf{Pr}_x[E_z(x) = 1] = p_z$. Since E_z has size at most $O(s+n) = O(s)$, this circuit is ε-fooled by G, assuming that b is large enough. The pseudorandomness property of G against the test computed by E_z yields the desired bound. $\diamond$

Consider a generator G as above. The previous lemma and discussion imply that given a circuit $C \colon \{0,1\}^m \to \{0,1\}^n$ of size polynomial in n and a parameter $\varepsilon \geq 1/\mathsf{poly}(n)$, and assuming access to an oracle that solves Avoid, we can compute in polynomial time (by iterating over all seeds of G) a set $S = \{z^1, \ldots, z^a\}$, where $a = \mathsf{poly}(n)$ and each $z^i \in \{0,1\}^n$, and probability estimates $\widetilde{p_1}, \ldots, \widetilde{p_a}$, such that the following hold:

- For every $i \in [a]$, $\widetilde{p_i} - \varepsilon \leq D_C(z^i) \leq \widetilde{p_i} + \varepsilon$.
- If a string $z \in \{0,1\}^n$ satisfies $D_C(z) > \varepsilon$, then $z \in S$.

In particular, given parameters $\delta > \varepsilon \geq 1/\mathsf{poly}(n)$, by picking appropriate parameters in the discussion above we can output in polynomial time a set $B_{\delta,\varepsilon} \subseteq \{0,1\}^n$ that contains every string z that is δ-heavy in D_C and no string z that is not $(\delta - \varepsilon)$-heavy in D_C. Given $B_{\delta,\varepsilon}$, we can easily solve `Heavy-Avoid` (by selecting any string not in $B_{\delta,\varepsilon}$) and `Heavy-Find` (by selecting a string in $B_{\delta,\varepsilon}$), as in the statement of Theorem A.1. $\square$

B Properties of the PSPACE-Complete Language

In this section, we discuss the proof of the following result.

Theorem 3.1 (A PSPACE-Complete Language with Useful Properties)**.** *There is a language $L^\star \subseteq \{0,1\}^*$ with the following properties:*

1. **(Complexity Upper Bound)** $L^\star \in \mathsf{PSPACE}$.
2. **(Completeness)** $L^\star$ *is* PSPACE*-hard under* $\mathsf{DLOGTIME}$*-uniform projection reductions.*
3. **(Instance Checkability)** *There is a* $\mathsf{DLOGTIME}$*-uniform family of* $\mathsf{BP\text{-}AC}^0[\oplus]$ *oracle circuits* $\{\mathsf{IC}_n\}_{n\geq 1}$ *making projection queries such that, on every input string* $x \in \{0,1\}^n$ *and for every oracle* $\mathcal{O} \subseteq \{0,1\}^*$*, the following holds:*
   - $\mathsf{IC}_n^{\mathcal{O}}(x)$ *only makes queries of length* n *to* $\mathcal{O}$*.*
   - *If* $\mathcal{O}$ *agrees with* $L^\star$ *on inputs of length* n*, then* $\mathbf{Pr}_r[\mathsf{IC}_n^{\mathcal{O}}(x;r) = L^\star(x)] = 1$*.*
   - *For every oracle* $\mathcal{O}$*,* $\mathbf{Pr}_r[\mathsf{IC}_n^{\mathcal{O}}(x;r) \in \{\perp, L^\star(x)\}] \geq 1 - \exp(-n)$*.*

To establish this theorem, we verify that the language called $L^{\mathsf{WH\text{-}TV}}$ described in [Che23, Section 7] satisfies the properties we need.[28] In particular, [Che23] showed that this language is PSPACE-complete and has instance checkers in $\mathsf{AC}^0[\oplus]$. However, [Che23] only considered P-uniformity. For both the instance checker and the PSPACE-hardness reduction, a few minor modifications of the construction are needed to achieve $\mathsf{DLOGTIME}$-uniformity.

The rest of this section will verify the required uniformity conditions by inspecting the proof in [Che23, Section 7]. Note that we will assume familiarity with [Che23, Section 7].[29]

[28]We work with $L^{\mathsf{WH\text{-}TV}}$ instead of the final language L^{PSPACE} because the only difference between $L^{\mathsf{WH\text{-}TV}}$ and L^{PSPACE} is *paddability*, which is not required for our arguments.

[29]The full version of [Che23] can be found at ECCC Report TR22-183.

B.1 Preliminaries

We assume familiarity with notations in [Che23, Section 7] such as pw_ℓ, ℓ_n, sz_n, and $\mathbb{F}_n$. We will use length-sz_n strings and elements in $\mathbb{F}_n$ interchangably (instead of explicitly going through the bijection κ_n). The following tasks can be computed in $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ [HV06]:

- (Iterated addition.) Given $\alpha_1, \ldots, \alpha_t \in \{0,1\}^{\mathsf{sz}_n}$, compute $\sum_{i\in[t]} a_i \in \{0,1\}^{\mathsf{sz}_n}$.
- (Iterated multiplication.) Given $\alpha_1, \ldots, \alpha_t \in \{0,1\}^{\mathsf{sz}_n}$ where $t \le \log n$, compute $\prod_{i\in[t]} a_i \in \{0,1\}^{\mathsf{sz}_n}$.

We will need a $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ circuit for *polynomial interpolation* over $\mathbb{F}_n$, which is the following task. Let $\alpha_1, \alpha_2, \ldots, \alpha_t$ be the lexicographically smallest t elements in $\mathbb{F}_n$. Given $\beta_1, \beta_2, \ldots, \beta_t \in \mathbb{F}_n$ and $z \in \mathbb{F}_n$ as inputs, the goal is to output $p(z)$, where $p : \mathbb{F}_n \to \mathbb{F}_n$ is the unique degree-$(t-1)$ polynomial over $\mathbb{F}_n$ such that $p(\alpha_i) = \beta_i$ for every $i \in [t]$.

It is shown in [Che23, Corollary 7.2] that the polynomial interpolation problem admits a uniform $\mathsf{AC}^0[\oplus]$ circuit when $t \le \log n$. Jumping ahead, the circuit is not $\mathsf{DLOGTIME}$-uniform since (3) requires one to compute the inverse of $\alpha_i - \alpha_j$, and it is unclear how to compute inverses over $\mathbb{F}_n$ in $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$. One way to work around this technical issue is to let the interpolation algorithm output two numbers $u, v \in \mathbb{F}_n$ such that $p(z) = u \cdot v^{-1}$.

Claim B.1. *For any constant $t \ge 1$*[30]*, there is a $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ circuit that given $z, \beta_1, \ldots, \beta_t \in \mathbb{F}_n$ as inputs, outputs two elements $u, v \in \mathbb{F}_n$ such that $p(z) = u \cdot v^{-1}$, where $p : \mathbb{F}_n \to \mathbb{F}_n$ is the unique degree-$(t-1)$ polynomial over $\mathbb{F}_n$ such that $p(\alpha_i) = \beta_i$ for every $i \in [t]$.*

Proof Sketch. The expression for $p(z)$ is

$$p(z) = \sum_{i\in[t]} \beta_i \cdot \prod_{j\in[t]\setminus\{i\}} \frac{z-\alpha_j}{\alpha_i - \alpha_j}. \tag{3}$$

Hence, we have $p(z) = u \cdot v^{-1}$ where

$$\begin{aligned} v &= \prod_{1\le i<j\le t} (\alpha_i - \alpha_j), \text{ and} \\ u &= p(z)v = \sum_{i\in[t]} \beta_i \cdot \prod_{j\in[t]\setminus\{i\}} (z-\alpha_j) \cdot \prod_{\substack{1\le i'<j'\le t \\ i'\ne i \text{ and } j'\ne i}} (\alpha_i - \alpha_j). \end{aligned}$$

(Note that we omitted some $(-1)^i$ terms since our field has characteristic 2.)

The desired $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ circuit follows from [HV06]. □

B.2 The Instance Checker

We assume familiarity with the notations in [Che23, Section 7], such as $S_Q, f_{n,i}, J_{n,j}, Q_{n,j}$. We start by showing that [Che23, Algorithm 7.1] (i.e., the instance checker for the polynomials $\{f_{n,i}\}$ defined in [Che23, Lemma 7.3]) admits a $\mathsf{DLOGTIME}$-uniform family of $\mathsf{BP}\text{-}\mathsf{AC}^0[\oplus]$ oracle circuits. The instance checker receives input parameters $n, i \in \mathbb{N}$ such that $1 \le i \le n$, input $\vec{x} \in \mathbb{F}_n^n$, and oracle access to $n-i+1$ functions $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n : \mathbb{F}_n^n \to \mathbb{F}_n$. It draws $z_i, z_{i+1}, \ldots, z_{n-1} \leftarrow \mathbb{F}_n$ uniformly at random and performs the following steps:

[30] Actually, solving the polynomial interpolation problem for $t = 3$ suffices for our instance checker.

1. First, it computes $\vec{\alpha}_i, \vec{\alpha}_{i+1}, \ldots, \vec{\alpha}_n \in \mathbb{F}_n^n$ as in [Che23, Eq. (8)]. For each $i \le j \le n$ and $\ell \in [n]$, let j_{max} be the maximum $j' < j$ such that $j' \ge i$, $J_{n,j'} = \ell$ and $Q_{n,j'} \neq \mathsf{MUL}$ (or $\perp$ if such j' does not exist). If j_{max} does not exist, then $(\vec{\alpha}_j)_\ell = x_\ell$; otherwise $(\vec{\alpha}_j)_\ell = z_{j_{\mathsf{max}}}$. We will show in Claim B.2 that given j and ℓ, we can compute j_{max} in $O(\log n)$ time; hence, we can compute each $\vec{\alpha}_j$ via a $\mathsf{DLOGTIME}$-uniform projection.

2. Then, it queries the oracles to obtain $r_j = \tilde{f}_j(\vec{\alpha}_j)$ for every $i \le j \le n$.

3. Let $t := c_{\mathsf{deg}} + 1 = 3$ (by [Che23, Lemma 7.7], the polynomials have individual degree at most 2), $w_1, \ldots, w_t$ be the first t non-zero elements of $\mathbb{F}_n$. For each $i \le j < n$ and $\ell \in [t]$, it queries the oracles to obtain $\beta_\ell^j := \tilde{f}_{j+1}((\vec{\alpha}_j)^{J_{n,j} \leftarrow w_\ell})$.

4. For every $i \le j < n$ *in parallel*:

   - If $Q_{n,j} = \mathsf{MUL}$, then it verifies that $r_j = r_{j+1} \cdot \mathsf{Term}_{n,j}(\vec{\alpha}_j)$, where $\mathsf{Term}_{n,j}$ is defined in [Che23, Eq. (4)] and hence computable by $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ circuits.
   - Otherwise, let p be the unique degree-$(t-1)$ polynomial such that $p(w_\ell) = \beta_\ell^j$ for every $\ell \in [t]$, we can use Claim B.1 to obtain the values $p(0)$, $p(1)$, and $p(z_j)$. If $r_j \neq S_{Q_{n,j}}((\vec{\alpha}_j)_{J_{n,j}}, p(0), p(1))$ or $r_{j+1} \neq p(z_j)$, then output $\perp$ and halt.

5. Finally, if $r_n = 1$ then accept and output r_i $(= \tilde{f}(\vec{x}))$; otherwise output $\perp$.

We remark that the values $p(0), p(1), p(z_j)$ in Item 4 are represented as $u \cdot v^{-1}$ for some $u, v \in \mathbb{F}_n$, but it is still possible to check the equalities. For example:

- If $Q = \exists$, then $S_Q(x, y_0, y_1) = y_0 \cdot y_1$, hence $S_Q(x, y_0 z_0^{-1}, y_1 z_1^{-1}) = r$ if and only if $y_0 y_1 = r z_0 z_1$.
- If $Q = \forall$, then $S_Q(x, y_0, y_1) = 1 - (1 - y_0)(1 - y_1)$, hence $S_Q(x, y_0 z_0^{-1}, y_1 z_1^{-1}) = r$ if and only if $(z_0 - y_0)(z_1 - y_1) = z_0 z_1 (1 - r)$.
- If $Q = \mathsf{LIN}$, then $S_Q(x, y_0, y_1) = x y_1 + (1 - x) y_0$, hence $S_Q(x, y_0 z_0^{-1}, y_1 z_1^{-1}) = r$ if and only if $x y_1 z_0 + (1 - x) y_0 z_1 = r z_0 z_1$.

The first three steps above issue queries to the oracles $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n$, and it is easy to see that these queries can be generated by a $\mathsf{DLOGTIME}$-uniform projection over $\vec{x}$ and $\vec{z} = (z_i, \ldots, z_{n-1})$. The last two steps above perform $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ computation over $\vec{z}$ and the answers returned from the oracles. This establishes the complexity of the instance checker.

Finally, we need to compute j_{max} efficiently:

Claim B.2. *There is an algorithm running in $O(\log n)$ time that given $i < j \le n$ and $\ell \in [n]$, finds the maximum $j' < j$ such that $j' \ge i$, $J_{n,j'} = \ell$, and $Q_{n,j'} \neq \mathsf{MUL}$.*

Proof. We recall the definitions of $J_{n,j}$ and $Q_{n,j}$ from [Che23, Proof of Lemma 7.6]. For some integers $m < \lambda < \sqrt{n}$ computable in $O(\log n)$ time, we have:

$$(J_{n,j}, Q_{n,j}) = \begin{cases} (1, \mathsf{LIN}) & \text{if } j > \lambda^2, \\ (j \bmod \lambda, \mathsf{LIN}) & \text{otherwise, if } \lambda \nmid j, \\ (1, \mathsf{MUL}) & \text{otherwise, if } j \ge (m+1)\lambda, \\ (j/\lambda, Q_{j/\lambda} \in \{\exists, \forall\}) & \text{otherwise.} \end{cases}$$

If $\ell \ge \lambda$, then we can safely return $\perp$. If $\ell = 1$ and $j > \lambda^2 + 1$, then we can simply return $j_{\mathsf{max}} = j - 1$. Otherwise, there are only two possible candidates for j_{max}:

- The case that $(J_{n,j}, Q_{n,j}) = (j \bmod \lambda, \mathsf{LIN})$: the largest $j' < \min\{j, \lambda^2\}$ s.t. $j \bmod \lambda = \ell$;
- The case that $(J_{n,j}, Q_{n,j}) = (j/\lambda, Q_{j/\lambda})$: $j' = \ell \cdot \lambda$.

We discard any candidate not in the range $[i, j)$ and return the (larger) remaining candidate j'. If both candidates are not in $[i, j)$ then we return $\bot$. This finishes the algorithm for finding j_{max} in $O(\log n)$ time. $\square$

The instance checker for $L^{\mathsf{WH\text{-}TV}}$ reduces to the instance checker for the polynomials $\{f_{n,i}\}$ in a straightforward way. In fact, let *input* $\in \{0,1\}^m$ be the input of $L^{\mathsf{WH\text{-}TV}}$ and k be the integer defined in [Che23, Algorithm 7.2] (computable in $O(\log m)$ time), then:

- given access to (a purported oracle for) the m-th slice of $L^{\mathsf{WH\text{-}TV}}$, one can access the polynomials $f_{n_k,i_k}, f_{n_k,i_k+1}, \dots, f_{n_k,n_k}$ via $\mathsf{DLOGTIME}$-uniform projections;
- given *input* and the answers of each $f_{n_k,j}(\vec{x})$ $(j \geq i_k)$, where $\vec{x} \in \mathbb{F}_n^n$ corresponds to the length-$(n \cdot \mathsf{sz}_n)$ prefix of *input*, one can compute $L^{\mathsf{WH\text{-}TV}}(\mathit{input})$ using [Che23, Eq. (10) and (11)] via a $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0[\oplus]$ circuit.

Since the instance checker for $\{f_{n,i}\}$ is a $\mathsf{DLOGTIME}$-uniform $\mathsf{BP\text{-}AC}^0[\oplus]$ circuit making projection queries, so is the instance checker for $L^{\mathsf{WH\text{-}TV}}$. Finally, the error probability of the instance checker for $\{f_{n,i}\}$ is at most $\mathsf{poly}(n)/2^n$ by [Che23, Claim 7.12], hence the error probability of the instance checker for $L^{\mathsf{WH\text{-}TV}}$ is also at most $\mathsf{poly}(n)/2^n$ by a union bound.

B.3 PSPACE-Completeness

We also need to show that $L^{\mathsf{WH\text{-}TV}}$ is PSPACE-complete under $\mathsf{DLOGTIME}$-uniform projections. Note that $L^{\mathsf{WH\text{-}TV}}$ is an arithmetization of the problem $\mathsf{TQBF}^{\mathsf{u}}$ defined in [Che23, Section 7.3], thus we first show that $\mathsf{TQBF}^{\mathsf{u}}$ is PSPACE-complete under $\mathsf{DLOGTIME}$-uniform projections, and then reduce $\mathsf{TQBF}^{\mathsf{u}}$ to $L^{\mathsf{WH\text{-}TV}}$ using $\mathsf{DLOGTIME}$-uniform projections. However, the PSPACE-completeness reduction presented in [Che23, Section 7.3] (from the classical TQBF to $\mathsf{TQBF}^{\mathsf{u}}$) is not a projection, so we need to implement the reduction more efficiently here.

Definition of $\mathsf{TQBF}^{\mathsf{u}}$. There are $8 \cdot \binom{n}{3}$ possible width-3 clauses on n variables and we let $\phi_n^{\mathsf{cl\text{-}idx}}$ be a bijection between $[8 \cdot \binom{n}{3}]$ and the set of valid width-3 clauses. A 3-CNF ϕ can thus be described by a string $y \in \{0,1\}^{8 \cdot \binom{n}{3}}$. The $\mathsf{TQBF}^{\mathsf{u}}$ problem takes such a bit-string as an input, constructs the corresponding 3-CNF $\Phi(x_1, x_2, \dots, x_n)$, and outputs

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \; \Phi(x_1, x_2, \dots, x_n), \tag{4}$$

where Q_i equals $\exists$ for odd i and $\forall$ for even i.

PSPACE-completeness of $\mathsf{TQBF}^{\mathsf{u}}$. First, the proof of the PSPACE-completeness of TQBF (that is, computing (4) when Φ is a *general circuit*) actually shows the following stronger result: For every language $L \in \mathsf{PSPACE}$, there is a polynomial-time Turing machine M and a polynomial $\ell(n)$ such that, for every $z \in \{0,1\}^n$, $z \in L$ if and only if

$$Q_1 x_1 Q_2 x_2 \dots Q_{\ell(n)} x_{\ell(n)} \; M(z_{1\sim n}, x_{1\sim \ell(n)}),$$

where, again, Q_i equals $\exists$ for odd i and $\forall$ for even i. (See [ALR99] for an exposition. In particular, [ALR99, Section 6.2] pointed out that the above PSPACE-completeness reduction can be computed by a $\mathsf{DLOGTIME}$-uniform projection.)

Since P is equal to $\mathsf{DLOGTIME}$-uniform $\mathsf{SIZE}[\mathsf{poly}]$ [BI94], there is a family of $\mathsf{poly}(n)$-size circuits $\{C_n \colon \{0,1\}^{n+\ell(n) \to \{0,1\}}\}$ that simulates M and satisfies the following uniformity conditions. Let $s(n) \le \mathsf{poly}(n)$ denote the number of gates in C_n (including input gates), then $1, 2, \dots, s$ is a valid topological order of C_n (the first $n + \ell$ gates are inputs and the s-th gate is the output gate). By adding dummy gates, we may assume that no gate has both children being z_i variables (this will imply that our final is 1-local). Finally, the direct connection language of C_n can be computed in $O(\log n)$ time: there is an algorithm running in deterministic $O(\log n)$ time that given n, indices of gates $g_1, g_2, g_3 \in [|C_n|]$ (where $g_1 > \max\{g_2, g_3\}$), and assignments $b_1, b_2, b_3 \in \{0,1\}$, returns true if and only if the outputs of g_2 and g_3 are fed as inputs of g_1 and $\{g_i = b_i\}_{i \in [3]}$ is consistent with the gate type of g_1 (e.g., if g_1 is an AND gate, then it cannot be the case that $b_1 = 1$ but $b_2 = 0$).

Now we show that this implies a $\mathsf{DLOGTIME}$-uniform projection reduction from L to $\mathsf{TQBF}^{\mathsf{u}}$. We will reduce an instance $z \in \{0,1\}^n$ of L to a $\mathsf{TQBF}^{\mathsf{u}}$ instance Φ_z with $s(n)$ variables. Let D be a clause expressing

$$(g_i \ne b_i) \vee (g_j \ne b_j) \vee (g_k \ne b_k),$$

where g_i, g_j, g_k are variables corresponding to gates in C_n and $b_i, b_j, b_k \in \{0,1\}$. We may assume that $g_i > \max\{g_j, g_k\}$. Then, D appears in Φ if and only if g_j, g_k are fed as inputs of g_i but $\{g_i = b_i\}_{i \in [3]}$ is *inconsistent* with the gate type of g_i. This information can be retrieved by $O(1)$ queries to the direct connection language.

Finally, the $\mathsf{TQBF}^{\mathsf{u}}$ instance we produced is

$$Q_1 x_1 Q_2 x_2 \dots Q_{\ell(n)} x_{\ell(n)} \exists g_{(\ell(n)+n+1)\sim s(n)} \ \Phi(z_{1\sim n}, x_{1\sim \ell(n)}, g_{(\ell(n)+n+1)\sim s(n)}),$$

and we may insert $\forall$ quantifiers among $g_{(\ell(n)+n+1)\sim s(n)}$ to make the quantifiers alternate. It is easy to see that the reduction is computable in $\mathsf{DLOGTIME}$; it is a projection because every clause (i.e., gate in C_n) only touches one z_i variable.

PSPACE-completeness of $L^{\mathsf{WH\text{-}TV}}$. The reduction from $\mathsf{TQBF}^{\mathsf{u}}$ to $L^{\mathsf{WH\text{-}TV}}$ is straightforward. First, by [Che23, Lemma 7.10], the truth-table of $\mathsf{TQBF}^{\mathsf{u}}$ on input length m coincides with the truth-table of $f_{n,1} = g_{1,1}^{(n)}$ over the Boolean cube, for some $n = \mathsf{poly}(m)$. Hence, when $L = \mathsf{TQBF}^{\mathsf{u}}$, the algorithm A_L^{red} (as defined in Item 4 of [Che23, Lemma 7.3]) is a $\mathsf{DLOGTIME}$-uniform projection. Second, the reduction in [Che23, Lemma 7.19] produces the instance $(\vec{z}, \vec{y}, \vec{u})$ where $\vec{y}$ and $\vec{u}$ are constant vectors whose each bit can be computed trivially, and $\vec{z} = A_L^{\mathsf{red}}(x)$ and x is the input of L. Hence, when $L = \mathsf{TQBF}^{\mathsf{u}}$, this reduction is also a $\mathsf{DLOGTIME}$-uniform projection over x.

Combining all of the above, we can see that the overall reduction from any language $L \in \mathsf{PSPACE}$ to $L^{\mathsf{WH\text{-}TV}}$ is a $\mathsf{DLOGTIME}$-uniform projection.