Electronic Colloquium on Computational Complexity, Report No. 95 (2021)

# LEARN-Uniform Circuit Lower Bounds and Provability in Bounded Arithmetic

Marco Carmosino[*] Valentine Kabanets[†] Antonina Kolokolova[‡] Igor C. Oliveira[§]

July 7, 2021

## Abstract

We investigate randomized LEARN-uniformity, which captures the power of *randomness* and *equivalence queries* (EQ) in the construction of Boolean circuits for an explicit problem. This is an intermediate notion between P-*uniformity* and *non-uniformity* motivated by connections to learning, complexity, and logic. Building on a number of techniques, we establish the first *unconditional* lower bounds against LEARN-uniform circuits:

- For all $c \geq 1$, there is $L \in \mathsf{P}$ that is not computable by circuits of size $n \cdot (\log n)^c$ generated in deterministic polynomial time with $o(\log n / \log \log n)$ equivalence queries to $L$. In other words, small circuits for $L$ cannot be efficiently learned using a bounded number of EQs.
- For each $k \geq 1$, there is $L \in \mathsf{NP}$ such that circuits for $L$ of size $O(n^k)$ cannot be learned in deterministic polynomial time with access to $n^{o(1)}$ EQs.
- For each $k \geq 1$, there is a problem in promise-ZPP that is not in FZPP-uniform $\mathsf{SIZE}[n^k]$.
- Conditional and unconditional lower bounds against LEARN-uniform circuits in the general setting that combines randomized uniformity and access to EQs.

In all these lower bounds, the learning algorithm is allowed to run in *arbitrary* polynomial time, while the hard problem is computed in some *fixed* polynomial time.

We employ these results to investigate the (un)provability of *non-uniform circuit upper bounds* (e.g., Is NP contained in $\mathsf{SIZE}[n^3]$?) in *theories of bounded arithmetic*. Some questions of this form have been addressed in recent papers of Krajíček-Oliveira (2017), Müller-Bydzovsky (2020), and Bydzovsky-Krajíček-Oliveira (2020) via a mixture of techniques from proof theory, complexity theory, and model theory. In contrast, by extracting computational information from proofs via a direct translation to LEARN-uniformity, we establish robust unprovability theorems that unify, simplify, and extend nearly all previous results. In addition, our lower bounds against *randomized* LEARN-uniformity yield unprovability results for theories augmented with the *dual weak pigeonhole principle*, such as $\mathsf{APC}^1$ (Jeřábek, 2007), which is known to formalize a large fragment of modern complexity theory.

Finally, we make precise potential limitations of theories of bounded arithmetic such as PV (Cook, 1975) and Jeřábek's theory $\mathsf{APC}^1$, by showing *unconditionally* that these theories cannot prove statements like "$\mathsf{NP} \not\subseteq \mathsf{BPP} \wedge \mathsf{NP} \subset \text{io-}\mathsf{P}/\mathsf{poly}$", i.e., that NP is uniformly "hard" but non-uniformly "easy" on infinitely many input lengths. In other words, if we live in such a complexity world, then this cannot be established *feasibly*.

[*]Boston University, USA. E-mail: `marco@ntime.org`
[†]Simon Fraser University, Canada. E-mail: `kabanets@cs.sfu.ca`
[‡]Memorial University of Newfoundland, Canada. E-mail: `kol@mun.ca`
[§]University of Warwick, UK. E-mail: `igor.oliveira@warwick.ac.uk`

ISSN 1433-8092

## Contents

# 1 Introduction

The fundamental question *in* complexity theory is what can (and cannot) be *feasibly computed*. A fundamental meta-question *about* complexity theory is what lower/upper bounds can (and cannot) be *feasibly proved*. The two questions and their intimate relationship are the focus of this paper. We show new *computational complexity lower bounds*, and use them to derive new *unprovability results*. We achieve this by linking these two topics through the investigation of the *efficient learnability* of small Boolean circuits for explicit computational problems. We discuss the setting and explain our results next.

## 1.1 Overview: Computation, Proofs, and Learnability

Despite substantial efforts, we appear to be still far from settling basic questions about the complexity of computations, such as the power of efficient algorithms ($\mathsf{P}$ vs. $\mathsf{NP}$), the advantage of randomized computations ($\mathsf{P}$ vs. $\mathsf{BPP}$), and the limitations of non-uniform Boolean circuits ($\mathsf{NP}$ vs. $\mathsf{P/poly}$). We have, however, discovered many *connections* between these problems, and such connections have led to significant progress in theory of computation, mathematics, and beyond (cf. [Wig19]). In order to explain how our work relates to these fundamental problems, and the main connection highlighted by our results, we consider two research themes.

**Proofs and Computations.** A more refined way to investigate $\mathsf{P}$ vs. $\mathsf{NP}$ and circuit lower bounds, and indeed the very nature of *complexity*, is to measure the feasibility of a computation not just by the *running time/circuit size*, but also by how complicated it is to establish its *correctness/existence* via a mathematical proof (see, e.g., the textbooks [Kra95; CN10]).

In this research program, we fix an appropriate formal theory $T$ known to formalize core concepts from algorithms and complexity, such as Cook's theory $\mathsf{PV}$ [Coo75] for polynomial-time reasoning or Buss's theory $\mathsf{S}^1_2$ [Bus86], and consider the *provability* of a complexity upper bound in $T$ (cf. [CK07]). As a concrete example, if $T$ *does not prove* that $\mathsf{SAT} \in \mathsf{SIZE}[n^3]$, then either there is no family of circuits of size $O(n^3)$ that solve $\mathsf{SAT}$, or any proof of the existence of such circuits requires concepts and arguments that go beyond the reasoning capabilities of $T$. Consequently, as we change the base theory $T$, we are able to simultaneously investigate, in a principled way, the power and limits of *proofs* and *computations*.[1]

Note that establishing the unprovability of upper bounds is a *necessary* step towards showing an unconditional complexity lower bound. This takes us to our next theme.

**Circuit Complexity, Uniformity, and Randomness.** Given that proving circuit lower bounds is a notoriously difficult task, several works focused instead on lower bounds against *strongly* uniform families of Boolean circuits, such as DLOGTIME-uniform constructions (cf. [Vol99]). Such results become more difficult to obtain when one considers *weaker* notions of uniformity. Indeed, establishing lower bounds for $\mathsf{SAT}$ against $\mathsf{P}$-*uniform* Boolean circuits of polynomial size is equivalent to separating $\mathsf{P}$ and $\mathsf{NP}$, while proving exponential lower bounds against *non-uniform* circuits provides an approach to show that $\mathsf{P} = \mathsf{BPP}$ [IW97].

The investigation of intermediate notions between $\mathsf{P}$-uniform and non-uniform circuits is also well motivated (see, e.g., [SW14]). For instance, in order to separate $\mathsf{P}$ and $\mathsf{BPP}$, it is sufficient to study *randomized uniformity*. Relevant to the discussion below, we note that non-uniformity is

[1] A proof that $\mathsf{PV}$ cannot show that $\mathsf{SAT} \in \mathsf{P}$ would significantly advance our understanding of efficient computations and proofs. Such a result would be implied by a super-polynomial proof size lower bound against the Extended Frege proof system, a central open question in proof complexity (cf. [Kra19]).

essentially equivalent to zero-error polynomial-time uniformity with access to an $\mathsf{NP}$ oracle, since small circuits for a hard problem like $\mathsf{SAT}$, when they exist, can be produced in $\mathsf{FZPP}^{\mathsf{NP}}$ [Bsh+96]. Thus, from the perspective of proving lower bounds, the extension of $\mathsf{P}$-uniformity with randomness and arbitrary $\mathsf{NP}$ queries takes us to non-uniform constructions. In particular, this suggests the possibility of a rich landscape of uniformity notions between $\mathsf{P}$-uniformity and $\mathsf{FZPP}^{\mathsf{NP}}$-uniformity. As explained next, this turns out to be indeed the case.

A recurring theme in this work is that a mathematical proof of the existence of an object often provides more information about the object than just existence. This leads to a far-reaching connection between *provability* on the one hand, and *circuit uniformity* on the other hand: if a theory $T$ proves that a problem admits small Boolean circuits, then such circuits admit a uniform construction that, while weaker than $\mathsf{P}$-uniformity, is not as general as $\mathsf{FZPP}^{\mathsf{NP}}$-uniformity. As a consequence, establishing certain *uniform circuit lower bounds* implies that the theory $T$ does not prove a corresponding *non-uniform circuit upper bound*.

Interestingly, the right notion of uniformity in this context is related to a natural class of *learning algorithms* with access to *equivalence queries* (EQ).[2] This motivates the investigation of LEARN-uniform Boolean circuits, which are families of circuits for a language $L$ that can be efficiently constructed with access to the EQ oracle that answers the following question when queried with a candidate circuit $C$: Does $C$ compute $L$? If so, the oracle answers "yes"; otherwise, the oracle answer is a *counterexample*: an (arbitrary) input $x$ such that $L(x) \neq C(x)$.[3] Provability in more powerful theories naturally leads to *randomized* learning algorithms for constructing Boolean circuits for $L$, which are significantly more difficult to analyze.

Motivated by this connection and its consequences, we initiate a systematic investigation of lower bounds against randomized uniformity and LEARN-uniformity. In addition to making explicit the relation between provability of non-uniform circuit upper bounds and learnability, our main contributions are:

(*i*) Building on advances from the study of uniform circuits [SW14], (un)provability of upper bounds [CK07; KO17; BKO20], and randomness in computation [LOS21], we establish unconditional circuit lower bounds against weaker uniformity notions.

(*ii*) By extracting computational information from proofs via a direct translation to different forms of LEARN-uniformity, we establish several robust unprovability theorems that unify, simplify, and extend nearly all previous results [KO17; BM20; BKO20].

(*iii*) Our lower bounds against the expressive model of *randomized* LEARN-uniformity yield unprovability results for theories augmented with the *dual weak pigeonhole principle*, such as $\mathsf{APC}^1$ [Jeř07a], which can formalize a large part of modern complexity theory [MP20].

(*iv*) Finally, we show certain separations between uniform and non-uniform computations (e.g., the possibility that both $\mathsf{NP} \not\subseteq \mathsf{BPP}$ and $\mathsf{NP} \subseteq \text{io-}\mathsf{SIZE}[\mathsf{poly}]$), if true, cannot be proved *feasibly*.

We now explain our results in more detail and contrast them with previous work.

[2]A more general notion of Student-Teacher games that is also motivated by provability has been studied in several works (cf. [Kra95]). In contrast, our setup is *specific* to the provability of circuit upper bounds.

[3]Learnability with EQs is a well-known research area in computational learning. Here, however, we are mainly concerned with proving the learnability *lower bounds* for a *fixed* and *known* language.

## 1.2 Results

### 1.2.1 Unconditional LEARN-Uniform Circuit Lower Bounds

For concreteness, our results refer to standard Boolean circuits consisting of AND, OR, and NOT gates of fan-in at most two. For a circuit size bound $s(n)$ and a query bound $r(n)$, we say that a language $L$ is in $\mathsf{LEARN}^{\mathsf{EQ}[r]}$-uniform $\mathsf{SIZE}[s]$ if there is a deterministic polynomial-time algorithm $A$ that, when given $1^n$ as input and oracle access to an *equivalence query* (EQ) oracle for $L$, makes at most $r(n)$ queries to the oracle on circuits of size $\leq s(n)$ defined over $n$ input variables, and outputs a circuit $C$ of size $\leq s(n)$ that computes $L_n$ (i.e., $L$ restricted to $\{0,1\}^n$). A query to the oracle EQ on a circuit $D$ returns "correct" if $D = L_n$; otherwise it returns a counterexample $x \in \{0,1\}^n$ such that $D(x) \neq L_n(x)$. Note that the learner must succeed after making at most $r(n)$ queries, no matter which particular counterexamples are provided by the EQ oracle.

In some results, we will also consider the standard search problem Search-SAT: Given a Boolean formula $\varphi$ of bitlength $n$, decide if $\varphi$ is satisfiable, and if it is, then output a satisfying assignment of $\varphi$. Note that this is equivalent to the promise problem of producing a satisfying assignment on any input formula that is satisfiable. This is a fundamental computational problem both in complexity theory and in the context of SAT solvers. In order to discuss the construction of Boolean circuits that solve Search-SAT, we adapt our LEARN-uniform framework as follows: the EQ oracle, when queried on a (multi-output) circuit $D$, outputs "correct" if $D$ solves Search-SAT on all satisfiable formulas of length $n$, or a pair $(\varphi, z)$ with $\varphi(z) = 1$ (i.e., a satisfiable formula with a proof of satisfiability) such that $D$ fails to find a satisfying assignment on $\varphi$ (i.e., $\varphi(D(\varphi)) = 0$). We call an oracle of this form a Search-SAT-EQ oracle.

We establish the following lower bounds against circuits that can be deterministically constructed using equivalence queries.

**Theorem 1.1** (Lower bounds against deterministic LEARN uniformity)**.**

1. *For all $k \geq 1$, there is a language $L \in \mathsf{P}$ such that $L \notin \mathsf{LEARN}^{\mathsf{EQ}[O(1)]}$-uniform $\mathsf{SIZE}[n^k]$.*
2. *For all $C \geq 1$ and $r(n) = o(\log n/\log\log n)$, $\mathsf{P} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[r(n)]}$-uniform $\mathsf{SIZE}[n \cdot (\log n)^C]$.*
3. *For all $k \geq 1$, $\mathsf{NP} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[n^{o(1)}]}$-uniform $\mathsf{SIZE}[n^k]$.*
4. *For all $k \geq 1$, $\mathsf{NP} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[n^{O(1)}]}$-uniform $\mathsf{SIZE}[n^k]$ or*

$$\mathsf{Search\text{-}SAT} \notin \mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[n^{O(1)}]}\text{-uniform } \mathsf{SIZE}[n^k].$$

**Discussion.** Items 1 and 2 show that there are relatively easy problems such that small circuits for them cannot be learned deterministically in polynomial time using a bounded number of equivalence queries. As we move from P to NP, we are able to handle more equivalence queries in the lower bound (Item 3 of Theorem 1.1). Finally, Item 4 of Theorem 1.1 rules out polynomial-time constructions with an arbitrary number of equivalence queries for some computational task that can be efficiently solved with nondeterminism.

We stress that, in each lower bound stated in Theorem 1.1, the hard problem can be solved in some *fixed* polynomial-time bound, while our notions of uniformity allow computations of *arbitrary* polynomial time (which moreover are granted access to an EQ oracle). For this reason, simple diagonalization arguments are not sufficient to establish the lower bounds stated in Theorem 1.1 and in the rest of this paper.

In contrast to Items 1 and 2 of Theorem 1.1, [SW14] established that $\mathsf{P} \not\subseteq \mathsf{P}$-uniform $\mathsf{SIZE}[n^k]$. Their approach forms the base case of our lower bounds, which in addition need to handle up to

$o(\log n / \log \log n)$ EQs. As we explain in Appendix A, even a single EQ can provide significant power to uniform constructions. Note that one can eliminate each EQ with $n$ bits of advice, but it is unclear how to analyze the $\mathsf{P}/O(n)$-uniform constructions resulting from the elimination of $O(1)$ EQs. The fact that the lower bound from [SW14] does not easily extend to EQs is precisely what creates technical difficulties in the unprovability result from [KO17].

In contrast to Items 3 and 4 of Theorem 1.1, [SW14] proved that $\mathsf{NP} \not\subseteq \mathsf{P}^{\mathsf{NP}}_{||}$-uniform $\mathsf{SIZE}[n^k]$. Our results are incomparable in this case, as we need to handle up to $n^{O(1)}$ *adaptive* EQs, and each EQ can produce about $n$ bits of information, as opposed to the yes/no answer provided by an $\mathsf{NP}$ oracle. Building on ideas from [CK07], we present in Appendix B a strengthening of the aforementioned result of [SW14], which is also the starting point for the proof of Items 3 and 4.

**Randomized uniformity.** Before we consider *randomized* LEARN-uniform circuits, we turn our attention to the simpler setting of randomized uniformity. Perhaps surprisingly, it seems that a proper complexity-theoretic investigation of this notion has not been carried out until our work. For this reason, we discuss randomized uniformity in more detail.

First, observe that there are two ways of defining $\mathsf{P}$-uniformity. One approach is to insist that the direct-connection language $L_{\mathsf{dc}}$ of the circuit family $\{D_n\}_{n\geq 1}$ (roughly, the description of the internal structure of a circuit) is decidable in polynomial time. Another approach is to allow a polynomial-time computable function $F$ such that $F(1^n)$ outputs a (complete description of) circuit that computes $L_n$. It is not hard to see that, for this *deterministic* case, these two definitions are *equivalent*. However, when we consider efficient *probabilistic* computations, it is unclear if the analogous definitions coincide. On the one hand, if $L_{\mathsf{dc}} \in \mathsf{BPP}$ (i.e., $\mathsf{BPP}$-uniformity), then there is a probabilistic polynomial-time algorithm $A$ such that $A(1^n)$ outputs a complete description of the circuit $D_n$ with high probability. (This can be seen as a *pseudo-deterministic* construction of circuits for $L$, using the notion of [GG11].) On the other hand, however, the existence of a probabilistic polynomial-time algorithm $B$ such that $B(1^n)$ outputs with high probability *some* circuit that computes $L_n$ (i.e., $\mathsf{FBPP}$-uniformity) does not provide a *fixed* sequence of circuits for $L$, as $B$ may output *different* circuits for $L_n$ for different random coin tosses. While $\mathsf{BPP}$-uniformity and its zero-error analogue, $\mathsf{ZPP}$-uniformity, might be useful in some applications, the *functional* definitions are more general and considerably harder to analyze.[4]

Formal definitions of the notions discussed above are provided in Section 2.4. We establish the following lower bounds against randomized uniformity.

**Theorem 1.2** (Lower bounds against randomized uniformity). *For all $k \geq 1$, we have the following:*

1. $\mathsf{ZPP} \not\subseteq \mathsf{ZPP}$-uniform $\mathsf{SIZE}[n^k]$.
2. $\mathsf{BPP} \not\subseteq \mathsf{BPP}$-uniform $\mathsf{SIZE}[n^k]$.
3. promise-$\mathsf{ZPP} \not\subseteq \mathsf{FZPP}$-uniform $\mathsf{SIZE}[n^k]$.
4. $\mathsf{MA} \not\subseteq \mathsf{FBPP}$-uniform $\mathsf{SIZE}[n^k]$.

**Discussion.** Items 1 and 2 of Theorem 1.2 and Items 3 and 4 are incomparable: the notion of circuit uniformity is more general in the latter, but the hard problems lie in larger complexity classes. While we are not aware of directly related results in uniform circuit complexity, we recall

[4]Note that $\mathsf{BPP} = \mathsf{FBPP}$-uniform $\mathsf{SIZE}[\mathsf{poly}]$. Also, the standard proof that $\mathsf{BPP} \subset \mathsf{P}/\mathsf{poly}$ produces $\mathsf{FBPP}$-uniform circuits, as different ways of fixing the randomness of an algorithm induce different (deterministic) Boolean circuits.

the following non-uniform lower bound: [San09] proved that $\mathsf{MA}/1 \not\subseteq \mathsf{SIZE}[n^k]$, and it is a well-known open problem to remove the advice bit. Theorem 1.2 Item 4 shows that this is possible at the cost of replacing non-uniformity by $\mathsf{FBPP}$-uniformity.

In terms of technical contributions, Items 1, 2, and 4 of Theorem 1.2 follow by easy adaptations of existing techniques. We have included them here as they provide context for our results and open problems, and in particular complement our most interesting contribution in Theorem 1.2, which is the lower bound against $\mathsf{FZPP}$-uniformity in Item 3. The proof of this result makes use of a very recent pseudodeterministic PRG against polynomial-time computations [LOS21], which is combined in a non-trivial way with other techniques. We say more about this in Section 1.3.

**Randomized LEARN-uniformity.** Finally, we consider randomized LEARN-uniformity, which combines $\mathsf{FBPP}$-uniformity and the power of equivalence queries, and can be viewed as a kind of "interactive" uniformity. By asking an extra EQ, we can always assume that the learner is *zero-error*: it either outputs a correct circuit of size $s$, or a fail symbol "$\perp$". We allow the learner to use fresh randomness after receiving the answer to each EQ, and as before, we require the learner to succeed with high probability no matter the counterexamples provided by the EQ oracle. If a size $s$ circuit for language $L$ can be learned this way with at most $r$ EQs, we say that $L \in \mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{EQ}[r]}\text{-uniform } \mathsf{SIZE}[s]$. Analyzing randomized LEARN-uniform constructions is significantly more delicate than proving lower bounds against randomized uniformity or deterministic LEARN-uniformity, as correctness of the learner on a given execution depends both on its random choices and the provided counterexamples; see Section 2.4 for a formal treatment.

Before we state our lower bounds against randomized LEARN-uniform circuits, recall that $\mathsf{ZPP} \subseteq \mathsf{NP} \subseteq \mathsf{MA} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ (cf. [GZ11]), and that it is a frontier challenge to establish circuit lower bounds for $\mathsf{ZPP}^{\mathsf{NP}}_{||} = \mathsf{ZPP}^{\mathsf{NP}[O(\log n)]}$ (see [DPV18] for a recent reference). It is also unknown if *almost-everywhere* circuit lower bounds hold for $\mathsf{MA}/O(1)$ (i.e., lower bounds against $\text{io-}\mathsf{SIZE}[n^k]$). A result of this form would have interesting applications (see, e.g., [MW20]).

Following the terminology introduced in [TV07], we use $\mathsf{MA}//a$ to denote $\mathsf{MA}$ with $a(n)$ advice bits per input length $n$ that *can depend on the random string of the verifier*. We let $\text{io-}\mathsf{SIZE}^{\mathcal{O}}[s]$ denote the class of languages that can be computed on infinitely many input lengths by a circuit of size $s(n)$ with oracle access to the language $\mathcal{O}$.

**Theorem 1.3** (Lower bounds against randomized LEARN uniformity)**.**
*If* $\mathsf{Search\text{-}SAT} \in \mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[O(1)]}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}]$*, then for every* $k \geq 1$*,*

$$\mathsf{MA}//O(1) \not\subseteq \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[n^k] \quad \textit{and} \quad \mathsf{ZPP}^{\mathsf{NP}[O(1)]} \not\subseteq \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[n^k].$$

**Discussion.** In particular, we get an unconditional lower bound against randomized LEARN-uniform circuits constructed with a constant number of EQs for either $\mathsf{Search\text{-}SAT}$ or a language in $\mathsf{ZPP}^{\mathsf{NP}[O(1)]}$. This should be contrasted with a result of [San09] showing that $\mathsf{ZPP}^{\mathsf{NP}[O(1)]}/1 \not\subseteq \mathsf{SIZE}[n^k]$.[5] The latter does not impose a uniformity condition on the circuits, but computing the hard language requires advice.

Next, we use these lower bounds to show unprovability results in bounded arithmetic.

### 1.2.2 Unprovability of Non-Uniform Circuit Upper Bounds in Bounded Arithmetic

Bounded arithmetic theories are fragments of Peano Arithmetic with close ties to computational complexity and proof complexity. They present a formal notion of reasoning of a predefined

[5]The standard proof that $\mathsf{MA} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ [GZ11] actually shows that $\mathsf{MA} \subseteq \mathsf{ZPP}^{\mathsf{NP}[2]}$.

complexity, with important examples including Cook's theory $\mathsf{PV}$ [Coo75] of polynomial-time reasoning, Jeřábek's theory $\mathsf{APC}^1$ [Jeř04; Jeř05; Jeř07a; Jer09] for probabilistic polynomial time,[6] and Buss's theory $\mathsf{S}^1_2$ [Bus86]; in this work, we follow the equivalent *two-sorted* formulation[7] of [CN10], where $\mathsf{PV}$ becomes $\mathsf{VPV}^1$, $\mathsf{APC}^1$ becomes $\mathsf{VAPC}^1$, and $\mathsf{S}^1_2$ becomes $\mathsf{V}^1$. Informally, a theory *captures reasoning in a complexity class* $\mathsf{C}$ when the totality of every functions in $\mathsf{C}$ is provable within the theory, and, moreover, a proof of a (specific kind of an) existential statement yields an algorithm in the same class $\mathsf{C}$ to compute a witness for that existential quantifier. Such witness-finding algorithms for various theories of bounded arithmetic follow from the corresponding *witnessing theorems*, which exemplify the *constructive nature* of (existence) proofs in such theories. The witnessing theorems play the main role in all our unprovability results (see Section 2.5 for more details).

There is a veritable zoo of theories of various power, and numerous works exploring their connections to complexity theory and proof complexity; see [CN10; Bus97; Kra95; Kra19] for an introduction. For example, theory $\mathsf{VLV}$ (*logarithmic-space reasoning*) proves the existence of expander graphs [Bus+20]; $\mathsf{VPV}^1$ (*polynomial-time reasoning*) formalizes a proof of the PCP theorem [Pic15b]; and $\mathsf{VAPC}^1$ (*probabilistic polynomial-time reasoning*) establishes the correctness of randomized algorithms for perfect matching [LC11] and formalizes advanced circuit lower bounds [MP20].[8] For this reason, showing the *consistency of a complexity lower bound* with such theories (equivalently, the *unprovability of a complexity upper bound*) is a significant (and necessary) step towards understanding computational intractability.

Building on the seminal work of Cook and Krajíček [CK07], [KO17; BM20; BKO20] combined techniques from logic and complexity theory to establish the unprovability of various complexity upper bounds in different theories of bounded arithmetic, exploiting appropriate witnessing theorems for these theories. However, these witnessing theorems alone are *not* sufficient to derive unprovability results, due to the lack of lower bounds against the corresponding notions of circuit uniformity. To overcome this issue, these papers developed ad-hoc arguments that required, among other ideas, the formalization of complexity results and dealing with the specifics of the theory.

In contrast, as a consequence of the stronger circuit lower bounds stated in Section 1.2.1, we establish new results in bounded arithmetic and recover nearly all unprovability results from [KO17; BM20; BKO20]. Since the formalization of circuit complexity in bounded arithmetic has been discussed in detail in [Pic15a; KO17; BKO20; MP20] and would require a proper introduction to bounded arithmetic, we provide here only an informal overview of our results, and refer to the body of the paper for further discussion.

**Theorem 1.4** (Unprovability results for theories $\mathsf{VLV}$, $\mathsf{VPV}^1$, $\mathsf{V}^1$, and $\mathsf{VPV}^2$, informal)**.** *For all $k \geq 1$, we have the following:*

1. $\mathsf{VLV} \nvdash \mathsf{L} \subseteq \mathsf{BP\text{-}SIZE}[n^k]$ ($\mathsf{VLV}$ *does not prove logspace has branching program size $O(n^k)$).*
2. $\mathsf{VPV}^1 \nvdash \mathsf{P} \subseteq \mathsf{SIZE}[n^k]$.
3. $\mathsf{V}^1 \nvdash \mathsf{NP} \subseteq \mathsf{io\text{-}SIZE}[\mathsf{poly}] \cap \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[n^k]$. *In particular,* $\mathsf{V}^1 \nvdash \mathsf{NP} \subseteq \mathsf{SIZE}[n^k]$.
4. $\mathsf{VPV}^2 \nvdash \mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{SIZE}^{\mathsf{SAT}}[n^k]$ (*where* $\mathsf{VPV}^2$ *captures* $\mathsf{P}^{\mathsf{NP}}$ *reasoning*).

[6]The $\mathsf{APC}^i$ terminology comes from [BKT14] and has been used by other references, such as [MP20].

[7]The two-sorted framework allows one also to discuss theories capturing small complexity classes within $\mathsf{P}$, e.g., theory $\mathsf{VLV}$ capturing logarithmic space $\mathsf{L}$.

[8]See also [Oja04; Lê14] for formalizations in bounded arithmetic of several additional results from algorithms, probabilistic and extremal combinatorics, and complexity theory.

**Discussion.** Theorem 1.4 (Items 2–4) recovers nearly all results from [KO17; BM20; BKO20] on the unprovability of circuit upper bounds, with the exception of some extensions to infinitely often inclusions, which we do not systematically pursue here. On the other hand, Theorem 1.4 Item 1 does not have a counterpart in these papers. It is obtained by an adaptation of our LEARN-uniform lower bound results to the bounded-space setting (see Appendix D).

Thanks to the direct extraction of a LEARN-uniform sequence of circuits from a proof, the unprovability statements in Theorem 1.4 are all *robust* with respect to details of the formalization. For example, let $L \in \mathsf{NP}$ be the language granted by Theorem 1.4 Item 3 such that $\mathsf{V}^1 \nvdash \varphi_{c,k}$, where $\varphi_{c,k}$ is a $\mathsf{V}^1$-sentence stating that $L \in \mathsf{SIZE}[cn^k]$. While one can use different algorithms (verifiers) to specify $L$ when fixing the sentence $\varphi_{c,k}$, as a consequence of our approach via a reduction to an unconditional lower bound, this is inessential in Theorem 1.4. In previous works, this robustness was not always clear, or at least required a more elaborate argument.

**Probabilistic polynomial-time reasoning.** Our lower bounds against randomized LEARN-uniform circuits can be used to show results for theories extended with various forms of the *dual weak pigeonhole principle* ($\mathsf{dWPHP}$), such as Jeřábek's theories $\mathsf{VAPC}^1$ ($\mathsf{VPV}^1$ + the $\mathsf{dWPHP}$ axiom scheme) and $\mathsf{VAPC}^2$ (the analogue theory extending $\mathsf{VPV}^2$). Informally, for a function symbol $F$ of a theory $T$, the $\mathsf{dWPHP}(F)^n_m$ axiom (think of $m = n + \log n$ for instance) states that if $F\colon \{0,1\}^n \to \{0,1\}^m$ then there exists a "hole" $y \in \{0,1\}^m$ such that no "pigeon" $x \in \{0,1\}^n$ satisfies $F(x) = y$. From a *computational perspective*, this axiom takes us from the polynomial-time reasoning domain ($\mathsf{VPV}^1$) to the *probabilistic* polynomial-time reasoning domain ($\mathsf{VAPC}^1$). This is because it might not be feasible to produce a string $y$ with this property *deterministically*, but a *random* $y$ is likely to satisfy this condition. Probabilistic arguments are widespread in theoretical computer science, and it is quite remarkable that the $\mathsf{dWPHP}$ axioms alone can provide enough power for $\mathsf{VAPC}^1$ to bootstrap a convenient fragment of probability theory [Jeř04; Jeř05; Jeř07a] and to establish a vast number of results in algorithms and complexity theory (see, e.g., [LC11; BKT14; MP20] and references therein).

**Theorem 1.5** (Unprovability results for theories extended with $\mathsf{dWPHP}$, informal)**.** *For all $k \geq 1$, the following results hold, under appropriate formalizations.*

1. $\mathsf{VAPC}^1$ *does not prove that* $\mathsf{ZPP}^{\mathsf{NP}[O(1)]} \subseteq \mathsf{io\text{-}SIZE}[n^k]$.[9]
2. $\mathsf{VAPC}^1$ *does not prove that* $\mathsf{MA}//O(1) \subseteq \mathsf{io\text{-}SIZE}[n^k]$.[10]
3. *Either* $\mathsf{V}^1 + \mathsf{dWPHP} \nvdash \mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}]$, *or* $\mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \nsubseteq \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[n^k]$.[11]
4. *Either* $\mathsf{VAPC}^2 \nvdash \mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}]$, *or* $\mathsf{ZPP}^{\mathsf{NP}} \nsubseteq \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[n^k]$. *In particular,* $\mathsf{VAPC}^2 \nvdash \mathsf{ZPP}^{\mathsf{NP}} \subseteq \mathsf{io\text{-}SIZE}[n^k]$.

**Discussion.** Given our limited understanding of probabilistic computations and the expressive reasoning power of theories with $\mathsf{dWPHP}$, it is challenging to establish unprovability results for

[9]We prove the stronger result that either $\mathsf{VAPC}^1 \nvdash \mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}]$, or $\mathsf{ZPP}^{\mathsf{NP}[O(1)]} \nsubseteq \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[n^k]$. However, it is not clear how to express the latter in $\mathsf{VAPC}^1$, as this theory might not be expressive enough to define $\mathsf{ZPP}^{\mathsf{NP}}$.

[10]In order to discuss randomness-dependent advice and to talk about a language $L \in \mathsf{MA}//O(1)$ in $\mathsf{VAPC}^1$, we need to consider a relativized version of $\mathsf{VAPC}^1$ which includes a new relation symbol for the advice function.

[11]We say that a language $L \in \mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}}$ if there is a zero-error polynomial time algorithm that computes $L$ when given access to an (arbitrary) $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle. As opposed to LEARN-uniform constructions, this algorithm, when computing on input length $n$, can make oracle queries using circuits on $m$ input bits and of size $\mathsf{poly}(m)$, where $m$ is arbitrary.

$\mathsf{VAPC}^1$ and its extensions. The proof of Theorem 1.5 Item 1 requires several ideas and is one of the most technical arguments of the paper (see Section 1.3).

In comparison with Theorem 1.4, the upper bounds that we show to be unprovable in Theorem 1.5 are for larger uniform classes, and are close to existing unconditional lower bounds. For instance, Theorem 1.5 Item 4 should be contrasted with the separation $\mathsf{ZPP}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}[n^k]$ [KW98], while Item 2 should be contrasted with the lower bounds of [San09; MW20] for $\mathsf{MA}$ with advice (note that neither of these results rules out an *infinitely often* upper bound). Perhaps surprisingly, in the case of Item 3, we show that $\mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \subseteq \mathsf{io\text{-}SIZE}[\mathsf{poly}]$, which is believed to be false for $\mathsf{ZPP}^{\mathsf{NP}}$.[12] Improving the results in Theorem 1.5 is an important direction for future work (see Section 1.4).

**Unprovability of simultaneous uniform lower bounds and non-uniform upper bounds.** Finally, we revisit the relation between provability of circuit upper bounds and the extent to which theories of bounded arithmetic can distinguish *uniform* and *non-uniform* computations.

**Theorem 1.6** (Limits of provability: uniform separations vs. non-uniform inclusions, informal)**.**

1. $\mathsf{VPV}^1 \nvdash (\mathsf{NP} \not\subseteq \mathsf{P}) \land (\mathsf{NP} \subseteq \mathsf{io\text{-}SIZE}[\mathsf{poly}])$.
2. $\mathsf{VAPC}^1 \nvdash (\mathsf{NP} \not\subseteq \mathsf{BPP}) \land (\mathsf{NP} \subseteq \mathsf{io\text{-}SIZE}[\mathsf{poly}])$.

**Discussion.** Theorem 1.6 shows in a precise sense that if uniform computations are "weak" and non-uniform computations are "strong", then this cannot be proved *feasibly*. In order to establish this result, we essentially show that the polynomial-time reasoning of $\mathsf{VPV}^1$ is not "fine-grained" enough to distinguish between polynomial-time algorithms and polynomial-size circuits, resulting in the clash between the assumptions of non-existence of polytime algorithms and existence of polysize circuits that decide $\mathsf{SAT}$ on infinitely many input lengths. The proof utilizes the constructive nature of these theories, expressed though appropriate witnessing theorems (cf. Section 2.5), where the witnessing theorem appropriate for the *circuit upper* bound (KPT Witnessing) is played against the witnessing theorem appropriate for the *uniform lower* bound (Buss's Witnessing).

## 1.3 Techniques

In this section, we explain in more detail some of the techniques used to prove our circuit lower bounds and unprovability theorems. First, we sketch the ideas behind the proof of Theorem 1.1 Item 1, which shows a LEARN-uniform circuit lower bound for a problem in $\mathsf{P}$. After that, we explain the witnessing theorem for $\mathsf{VPV}^1$, its connection to LEARN-uniform circuits, and how the aforementioned circuit lower bound implies an unprovability result for $\mathsf{VPV}^1$ in a direct way (Theorem 1.4 Item 2). We then discuss some challenges and techniques associated with the investigation of theories with $\mathsf{dWPHP}$ (Theorem 1.5) and randomized uniformity (Theorems 1.2 and 1.3). We refer to the body of the paper for the techniques and proofs that we do not cover here.

**A lower bound against deterministic LEARN-uniform circuits.** As in [KO17], the starting point of our proof is a beautiful result established by [SW14] that $\mathsf{P} \not\subseteq \mathsf{P}$-uniform $\mathsf{SIZE}[n^k]$, which in our notation can be equivalently stated as $\mathsf{P} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[0]}$-uniform $\mathsf{SIZE}[n^k]$. As explained below, a slightly stronger version of their result will serve as the base case of our inductive argument, which proceeds via a delicate *query elimination* strategy.

[12] We prove the stronger result that $\mathsf{BPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \subseteq \mathsf{io\text{-}SIZE}[\mathsf{poly}]$, which provides a natural example of the power of *non-uniformity* and *infinitely often* constructions. See Section 4.4 and Appendix C for more details.

First, we note that the argument in [SW14] also establishes the following result: either $(i)$ $\mathsf{P} \not\subseteq \mathsf{SIZE}[n^k]$ (i.e., we have hardness against non-uniform circuits), or $(ii)$ $H \notin \textsf{io-P}/n^{\delta}\textsf{-uniform SIZE}[n^k]$ (i.e., we have almost-everywhere hardness against uniform circuits constructed with $n^{\delta}$ bits of advice), where $\delta = 1/2k$ and $H \in \mathsf{P}$ is an explicit problem obtained from a time-hierarchy theorem.

We can assume that $\mathsf{P} \subseteq \mathsf{SIZE}[n^k]$, since otherwise we are done. Under this assumption, we show the following "Query Elimination Lemma" (Lemma 3.5): If $H \in \mathsf{LEARN}^{\mathsf{EQ}[r]}/a(n)\textsf{-uniform SIZE}[n^k]$ for some advice function $1 \leq a(n) \leq n^{\delta}$, then $H \in \mathsf{LEARN}^{\mathsf{EQ}[r-1]}/a'(n)\textsf{-uniform SIZE}[n^k]$, where $a'(n) \leq (\log n) \cdot a(n)^k$. In other words, we can *eliminate an equivalence query* at the cost of an increase in the *advice complexity* of the learning algorithm. Note that constantly many applications of this lemma together with the base case imply Theorem 1.1 Item 1.

In order to prove the Query Elimination Lemma, we consider the first EQ made by the learner, and argue that it can be correctly answered in polynomial time with a *small* increase in advice complexity. Let $C$ be the corresponding circuit of size $\leq n^k$ used as input to the EQ. Since, by assumption, $(i)$ is false, $(ii)$ must hold. Therefore, using that $a(n) \leq n^{\delta}$, $C$ cannot compute $H$ on inputs of length $n$. Thus, in order to answer this query and establish the lemma, it is sufficient to produce in polynomial time and with a bounded increase in advice complexity some *counterexample*, i.e., an input $x$ such that $C(x) \neq H(x)$.

The language $H$ employed by [SW14] is defined via a hierarchy theorem against deterministic time with sub-linear advice. As a key idea, we explore in our argument a "*compressible-counterexample*" extension of this hierarchy result (Lemma 2.5). In other words, we argue that not only an algorithm of bounded running time and advice complexity fails to compute $H$, but it must also be incorrect on some $n$-bit string that admits a *short and efficient encoding.* (This is somewhat similar to a recent strategy implemented in [CLW20], where a complexity lower bound established via a hierarchy theorem is explicitly witnessed.) While it seems that we have made progress, there is another difficulty: in the proof of the time-hierarchy theorem, to enforce that $H \in \mathsf{P}$, we can only diagonalize against algorithms running in a *fixed* polynomial-time bound $n^c$. However, in LEARN-uniformity, the learner can run in any polynomial time bound. Thus, this is not sufficient to guarantee the existence of a counterexample of the desired form.

It turns out that a similar issue is also present (and addressed) by the lower bound argument from [SW14]. By combining their techniques and the idea presented above in a non-black-box way, it is possible to complete the proof of the Query Elimination Lemma.

By a careful implementation of this approach, we can also go beyond $r = O(1)$ EQs in the context of almost-linear size circuits (Theorem 1.1 Item 2). This is not possible using the less explicit approach of [KO17]. (The latter employs an iterative argument that composes $r$ different polynomial-time algorithms extracted from successive applications of Herbrand's Theorem. This is no longer a polynomial-time computation if $r = \omega(1)$.) Note, however, that the Query Elimination Lemma mentioned above cannot be applied more than $\log n / \log\log n$ times, as the advice complexity becomes too large and we can no longer contradict the base case.

**Provability of non-uniform upper bounds and LEARN-uniformity.** $\mathsf{VPV}^1$ is a *universal theory* (its axioms do not contain existential quantifiers) whose function symbols represent *polynomial-time algorithms.* We will rely on the following *Witnessing Theorem* from [KPT91], which states that if $\varphi(X, Y, Z)$ is a quantifier-free formula and $\mathsf{VPV}^1 \vdash \forall X\, \exists Y\, \forall Z\; \varphi(X, Y, Z)$, then there is a constant $k \geq 1$ and $\mathsf{VPV}^1$-terms $T_1, \ldots, T_k$ such that:

$$T \vdash \forall X\; \forall Z_1\; \ldots \forall Z_k\; \left[\varphi(X, T_1(X), Z_1) \vee \varphi(X, T_2(X, Z_1), Z_2) \vee \cdots \vee \varphi(X, T_k(X, Z_1, \ldots, Z_{k-1}), Z_k)\right].$$

In other words, if the original sentence is provable in $\mathsf{VPV}^1$, so is the sentence above. Most importantly for us, from the *soundness* of $\mathsf{VPV}^1$, this statement is *true.* Unlike the previous works [KO17;

BM20; BKO20], this is all we need from logic. We explain next how to think about this sentence as an interactive protocol, and how this perspective naturally yields LEARN-uniform circuits when the initial sentence $\forall X\, \exists Y\, \forall Z\; \varphi(X,Y,Z)$ states a *non-uniform circuit upper bound.*

The KPT Witnessing Theorem has the following intuitive interpretation due to [KPS90]. Consider a search problem $Q(X)$, where on input $X$, we need to find a string $Y$ such that $\forall Z\; \varphi(X,Y,Z)$. Then $Q(X)$ is solvable by the following $k$-round *Student-Teacher protocol*: The student starts by suggesting $T_1(X)$ as a solution $Y$ to the search problem $Q(X)$. Either it is correct, or there is a counterexample $Z_1$ such that $\neg\varphi(X, T_1(X), Z_1)$. Then a teacher provides some such counterexample value $Z_1$, and the protocol proceeds to the next round. Generally, in round $1 \leq i \leq k$, a student computes $T_i(X, Z_1, \ldots, Z_{i-1})$ (based on the counterexamples $Z_1, \ldots, Z_{i-1}$ received in the previous rounds), which is either a correct value for $Y$ and we are done, or there is a counterexample value $Z_i$, provided by a teacher, such that $\neg\varphi(X, T_i(X, Z_1, \ldots, Z_{i-1}), Z_i)$, in which case we continue to the next round $i+1$. The correctness guarantee is that, for every input $X$, the student successfully solves the search problem $Q(X)$ in some round $1 \leq i \leq k$.

In the proof of Theorem 1.1 Item 2, we are concerned with the provability of non-uniform circuit upper bounds for a language $L$, which can be formalized as follows: *for all* $n \geq 1$ there *exists* a circuit $C$ of size $\leq c \cdot n^k$ such that *for all* inputs $z$ of length $n$, $C(z) = L(z)$. For $L \in \mathsf{P}$, this statement can be captured by sentences of the form $\forall X\, \exists Y\, \forall Z\; \varphi(X,Y,Z)$, as discussed above (think of $X$ as $1^n$, $Y$ as the Boolean circuit $C$, and $Z$ as the $n$-bit string $z$). According to this dictionary, from the provability of a circuit upper bound, we obtain a Student-Teacher protocol for the search problem $Q(1^n)$, where the student proposes a candidate circuit $C$, and the teacher provides a counterexample $z$ to $C$ (an input $z$ such that $C(z) \neq L(z)$) if one exists. Moreover, the student always succeeds after at most $k$ queries, no matter the counterexamples provided by the teacher. Finally, on every choice of $n$, the student computes according to a *constant* number of *fixed* $\mathsf{VPV}^1$ terms $T_1, \ldots, T_k$. Since a term of $\mathsf{VPV}^1$ is just a composition of finitely many $\mathsf{VPV}^1$ function symbols (polynomial-time algorithms), it follows that the student computes in polynomial time. In other words, from provability in $\mathsf{VPV}^1$ of a non-uniform circuit upper bound for a language in $\mathsf{P}$, we can extract a $\mathsf{LEARN}^{\mathsf{EQ}[O(1)]}$-uniform $\mathsf{SIZE}[n^k]$ family of circuits for $L$.

Therefore, by starting with a language $L \in \mathsf{P}$ such that circuits of size $O(n^k)$ for $L$ cannot be deterministically learned in polynomial time with constantly many EQs, we get that $\mathsf{VPV}^1$ cannot establish non-uniform circuit upper bounds for $L$.

**Jeřábek's theory $\mathsf{VAPC}^1$ and the $\mathsf{dWPHP}$ principle.** One faces interesting challenges when trying to show the unprovability of circuit upper bounds in the stronger theory $\mathsf{VAPC}^1 = \mathsf{VPV}^1 + \mathsf{dWPHP}(\mathsf{VPV}^1)$, which adds to $\mathsf{VPV}^1$ a collection of axioms $\mathsf{dWPHP}(F)$, one for each polynomial-time algorithm $F$ in the vocabulary of $\mathsf{VPV}^1$. As mentioned in Section 1.2.2, $\mathsf{dWPHP}(F)$ roughly says that for every $n$ and $m(n) > n$, if we consider the restriction $F\colon \{0,1\}^n \to \{0,1\}^m$, then there is $y \in \{0,1\}^m$ such that for no $x \in \{0,1\}^n$ we have $F(x) = y$. Since $\mathsf{dWPHP}(F)$ is not a universal sentence, $\mathsf{VAPC}^1$ is not a universal theory, and the KPT Witnessing Theorem does not directly apply to $\mathsf{VAPC}^1$. Using a well-known technique from logic (a self-contained exposition appears in Section 4.3), it is possible to extend the list of function symbols in the vocabulary of $\mathsf{VAPC}^1$ in order to define a *conservative extension*[13] of $\mathsf{VAPC}^1$ that has a *universal axiomatization.* Roughly speaking, we introduce for each axiom $\mathsf{dWPHP}(F)$ of $\mathsf{VAPC}^1$ a new function symbol $F'$, and postulate that $F'(1^n)$ explicitly computes a string $y$ of the desired form. Having done this, it is possible to apply the KPT Witnessing to the resulting universal theory. Let $T_1, \ldots, T_k$ for some $k = O(1)$ be the terms (in the vocabulary of the extended theory) that appear in the conclusion

[13] That is, a formula without the new function symbol is provable in the extended theory iff it is provable in $\mathsf{VAPC}^1$.

of the KPT Witnessing Theorem when we apply it to a sentence of the appropriate form. Notice that we can no longer claim that each term $T_i$ computes a polynomial-time function, since $T_i$ might refer to one of the new function symbols $F'$. We discuss next how this issue can be mitigated with the use of probabilistic polynomial-time computations.

Provability of a sentence in a sound theory implies that the sentence is true in *any* model of the theory. In particular, if we assign correct values to all inputs of $F'$ relevant in a proof, given the values of the other relevant variables, we preserve the correctness of the corresponding Student-Teacher protocol. A natural strategy is to use *randomness* to define the value of a function $F'$ on relevant inputs of the form $1^n$, since a random string $y$ likely falls outside the range of $F$. A careful implementation of this idea allows us to establish a (semantic) KPT Witnessing Theorem for $\mathsf{VAPC}^1$, Theorem 4.7 below.[14] Oversimplifying a bit, we show that the provability of a non-uniform upper bound in $\mathsf{VAPC}^1$ yields a $\mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{EQ}[O(1)]}$-uniform algorithm that outputs a correct circuit with *non-negligible* probability. Thus lower bounds against *randomized* LEARN-uniform constructions that succeed with $1/n^{\Omega(1)}$ probability for a language in $\mathsf{P}$, or for $\mathsf{Search}\text{-}\mathsf{SAT}$, imply a corresponding unprovability result for $\mathsf{VAPC}^1$.

**Lower bounds against randomized notions of circuit uniformity.** We now discuss the techniques that go into the proof of Theorem 1.2 Item 3 and Theorem 1.3. As alluded to before, an aspect of randomized uniform constructions ($\mathsf{FZPP}$-uniformity) that makes them difficult to analyze is that different circuits can be produced for different random coin tosses of the underlying algorithm. This becomes even more intricate in the presence of EQs ($\mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{EQ}[\cdot]}$-uniformity), where the final circuit output by the learner depends on its internal random choices and on the sequence of counterexamples.

To prove a lower bound against $\mathsf{FZPP}$-uniform $\mathsf{SIZE}[n^k]$ (Theorem 1.2 Item 3), we combine several ideas. The starting point for this proof is that the separation $\mathsf{P} \nsubseteq \mathsf{P}$-uniform $\mathsf{SIZE}[n^k]$ from [SW14] can be adapted to show that $\mathsf{ZPP} \nsubseteq \mathsf{ZPP}$-uniform $\mathsf{SIZE}[n^k]$ (recall that this corresponds to a simpler *pseudodeterministic* form of circuit uniformity, where a *fixed* circuit of each length is produced with high probability). Moreover, we can tolerate in this lower bound up to $n^\varepsilon$ bits of advice, for some constant $\varepsilon(k) > 0$. Given this, as a natural goal, we attempt to show that any $\mathsf{FZPP}$-uniform construction (which might output different circuits) can be converted into a $\mathsf{ZPP}$-uniform construction (which outputs a canonical circuit) that uses up to $n^\varepsilon$ bits of advice. To achieve that, we view the computation of the $\mathsf{FZPP}$ algorithm $A(1^n, w)$ as a function of its random string $w$. We have $\Pr_w[A(1^n, w) \neq \perp] \geq 1/2$, and since $A$ is *zero-error*, if this event happens $A$ must output a correct circuit. Thus to get a lower bound it would be sufficient to have a *polynomial-time* computable PRG $G$ of seend length $\leq n^\varepsilon$ that fools the set $\{w \mid A(1^n, w) \neq \perp\}$: by fixing a seed $z$ such that $A(1^n, G(z)) \neq \perp$, we can convert the $\mathsf{FZPP}$-uniform construction into a $\mathsf{ZPP}$-uniform construction that uses $\leq n^\varepsilon$ bits of advice to encode $z$, contradicting an existing lower bound.

While constructing a PRG with these properties is a major open problem, in a very recent work, [LOS21] established *unconditionally* the existence of a *pseudo-deterministic* PRG $G'$ computable with one bit of advice and with seed length $\leq n^\varepsilon$ that fools fixed polynomial-time computations (such as $A(1^n, \cdot)$). This allows us to make progress on our plan of using a PRG and a short seed to fix a circuit in the output of the $\mathsf{FZPP}$ computation. Nevertheless, three challenges still remain: (1) $G'$ needs 1 bit of advice; (2) $G'$ is computable with two-sided error, while we need a zero-error construction; and (3) $G'$ only works infinitely often, which does not contradict the $\mathsf{ZPP}$-uniform

[14]Note that assigning an independent random string $y_n$ as $F'(1^n)$ for *all* values of $n \in \mathbb{N}$ will not satisfy the $\mathsf{dWPHP}(F)$ axiom, since the probability of succeeding might converge to zero. However, on each input, the student in the Student-Teacher protocol queries $F'$ on a constant number of inputs, and setting the remaining values of $F'$ in a valid way guarantees the correctness of the protocol, since it agrees with *some* model of the theory.

lower bound (since it is *not* an almost-everywhere lower bound). We are able to sidestep (1) by settling for a (weaker) lower bound for a problem in promise-ZPP instead of ZPP. However, (2) and (3) are more serious issues, and in order to overcome them, an argument that is more involved than our current plan seems unavoidable. Briefly, to address (2) and to make *two-sided error* computations *zero-error*, we rely on a *win-win* argument based on the *easy witness method* [Kab00; IKW02]. On the other hand, to address (3), we do not rely on the lower bound against ZPP-uniformity with advice as a black-box. Our argument requires a more delicate case analysis, and we refer to the actual proof for the details.

Due to use of the pseudodeterministic PRG with 1 bit of advice from [LOS21], the approach described above does not yield a lower bound for a *language* $L$ (i.e., a non-promise problem). Furthermore, the pseudodeterministic PRG from [LOS21] is not known to fool computations with advice complexity $(\log n)^{\omega(1)}$, which might be useful when combining the Query Elimination Lemma with an inductive argument. So, to prove lower bounds against $\mathsf{FZPP\text{-}LEARN}^{\mathsf{EQ}[O(1)]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$ (Theorem 1.3), we must proceed in a different way.

A direct analysis of $\mathsf{FZPP\text{-}LEARN}^{\mathsf{EQ}[\cdot]}$-uniform constructions seems challenging: after each query and counterexample, the learner employs fresh random bits and makes a new equivalence query that can depend on previous queries, randomness, and obtained counterexamples. As an important conceptual idea in our proof, we introduce a new model of randomized LEARN-uniformity where the learner must flip all its random coins *in advance* of interacting with the EQ oracle. We call this $\mathsf{FZPP}^{\mathsf{rf}}\text{-}\mathsf{LEARN}^{\mathsf{EQ}}$-uniformity, where rf stands for *randomness first* (see Section 2.4). In this model, the randomness of the learner is public, and for most choices of its random string, it must behave as a deterministic learner and succeed no matter the counterexamples provided by the EQ oracle.

In general, it is not clear if every randomized learner can be converted into a randomness-first learner without blowing up its query complexity and/or running time, since an adversarial EQ oracle might be able to produce counterexamples that are not so helpful once it has knowledge of future decisions of the learner. Crucially, we are able to show that, for Search-SAT, a simulation is possible without increasing the number of queries, while the overhead in the running time can be kept polynomial if the initial number of queries is constant.

The proof is done in two steps. First, we rely on a certain *parallel repetition procedure* that amplifies the success probability of the initial learner without increasing its query complexity. This explores specific aspects of the Search-SAT problem and of its corresponding equivalence oracle Search-SAT-EQ, and we refer to Lemma 3.26 for more details. After that, we prove via a delicate argument that a randomized learner that succeeds with overwhelming probability and that makes only constantly many queries can be converted into a randomness-first learner. See Lemma 3.29 for the details.

Obtaining lower bounds against randomness-first learners is a more accessible task. Intuitively, for most choices of the random string, we can pretend that we have a correct *deterministic* $\mathsf{LEARN}^{\mathsf{EQ}[O(1)]}\text{-}\mathsf{uniform}$ construction. In order to prove Theorem 1.3, we combine the aforementioned simulation result with ideas used in the proof of Theorem 1.1 (Items 3 and 4).

## 1.4 Open Problems

As explained above, there is a rich landscape of circuit uniformity notions between P-uniformity and non-uniformity, and for the purpose of proving circuit lower bounds for explicit problems, non-uniformity is essentially equivalent to $\mathsf{FZPP}^{\mathsf{NP}}$-uniformity (see Proposition 3.31). The fundamental connections between mathematical proofs and computation explored here show that some some of these notions are particularly interesting and appear in a natural context.

Our unconditional circuit lower bounds and unprovability results suggest several challenges and frontiers to advance this research program. First, we mention problems related to randomized uniformity and LEARN uniformity.

1. In connection with Theorem 1.1 (Items 1 and 2), is it possible to establish that $\mathsf{P} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[O(\log n)]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$?
2. Motivated by Theorem 1.1 (Items 3 and 4), prove that $\mathsf{NP} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[\mathsf{poly}]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$.
3. Strengthen Theorem 1.2 (Item 3) by showing that $\mathsf{ZPP} \not\subseteq \mathsf{FZPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$.
4. Motivated by Theorem 1.3, prove that $\mathsf{ZPP}^{\mathsf{NP}[O(1)]} \not\subseteq \mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{EQ}[O(1)]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$.

We finish with a few related questions about the provability of circuit complexity bounds for explicit computational problems.

5. Improve the consequence of Theorem 1.4 (Item 3) by proving that $\mathsf{V}^1 \nvdash \mathsf{P} \subseteq \mathsf{SIZE}[n^k]$. Using the witnessing theorem for $\mathsf{V}^1$, it is enough to show that $\mathsf{P} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[\mathsf{poly}]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$ (note that this is a strengthening of Problems 1 and 2).
6. In connection with Theorem 1.5 (Item 1), prove that $\mathsf{VAPC}^1 \nvdash \mathsf{ZPP} \subseteq \mathsf{SIZE}[n^k]$. A first step might be to obtain solutions to Problems 3 and 4.
7. Finally, can we establish the *independence* of some natural question in circuit complexity from a theory such as $\mathsf{VPV}^1$? In particular, can we narrow the gap between our results and the recent result from [PS21] on the unprovability of strong circuit lower bounds in $\mathsf{VPV}^1$?

**The remainder of the paper.** We give the necessary background in Section 2. Our LEARN-uniform circuit lower bounds are proved in Section 3: Theorem 1.1 Items 1 and 2 in Section 3.1, and Items 3 and 4 in Section 3.2; Theorem 1.2 Items 1 and 3 in Section 3.3, and Items 2 and 4 in Section 3.4; Theorem 1.3 is also proved in Section 3.4. Our unprovability in bounded arithmetic results of Theorems 1.4 and 1.5 are proved in Section 4. Finally, Theorem 1.6 is proved in Section 5.

**Acknowledgements.** We thank Sam Buss, Jan Krajíček, and Ján Pich for related discussions. We are particularly grateful to Emil Jeřábek for providing several important comments on an earlier version of the paper that improved our treatment of bounded arithmetic. Igor C. Oliveira received support from the Royal Society University Research Fellowship URF\R1\191059.

## 2 Preliminaries

### 2.1 Notation

We use the notation $\langle M \rangle$ for a reasonable binary encoding of mathematical object $M$. To denote a machine supplied with a *particular* sequence of advice $\alpha$, we abuse notation to write $M/\alpha$.

The '$\circ$' symbol denotes a pairing operation on binary strings so that, given strings $x$ and $y$, $z = x \circ y$ is efficiently computable such that $x$ and $y$ can be efficiently recovered from $z$; moreover, we require that $|x \circ y| \leq |x| + |y| + O(\log |x|)$.[15]

[15]The following standard encoding works: Given $x$ and $y$, let $\ell = |x|$. Construct $z = x \circ y$ as follows: write $\ell$ in binary, repeating each bit twice; add a "separator" 01 to signal the end of the encoding of $\ell$; and then append the concatenation of $x$ and $y$. Clearly, $|z| = 2\log|x| + 2 + |x| + |y|$, and one can efficiently recover $x$ and $y$ from $z$.

## 2.2 Complexity Theory Basics

We assume the basic notions of complexity theory [AB09]. Sometimes we need to use an $\mathsf{NP}$ oracle which, when the oracle query answer is "yes", will also produce a witness proving that the answer is "yes". This type of $\mathsf{NP}$ oracle access is denoted by $\mathsf{NP}[\mathsf{wit}, k]$, where $k \in \mathbb{N}$ is the number of queries allowed to be made to the $\mathsf{NP}$ oracle. When $k$ is a large enough (say, polynomial in the input size), then a standard search-to-decision reduction for $\mathsf{NP}$ can be used to produce witnesses, thereby obviating the need for $\mathsf{wit}$; however, when $k$ is small, say a constant, the use of $\mathsf{wit}$ seems essential.

We will need the notion of randomized semantic complexity classes with advice *dependent on randomness*, introduced by [TV07] for the case of $\mathsf{BPP}$. Their definition naturally generalizes to the cases of $\mathsf{AM}$ and $\mathsf{MA}$.

**Definition 2.1.** We say that $L \in \mathsf{AM}//a(n)$ if there is a deterministic polynomial $t(n)$-time predicate $A$ and a function $\alpha$ such that, for every $n \geq 1$,

$$\mathbf{Pr}_{r \in \{0,1\}^{t(n)}} \left[ \forall x \in \{0,1\}^n \quad \Big( x \in L \Longleftrightarrow \exists y \in \{0,1\}^{t(n)} \; A(x, y, r, \alpha(r)) \Big) \right] \geq \frac{3}{4},$$

and $|\alpha(r)| = a(n)$ for $|r| = t(n)$.

**Definition 2.2.** We say that $L \in \mathsf{MA}//a(n)$ if there is a polynomial $t(n)$-time algorithm $A$ and a function $\alpha$ such that, for every $n \geq 1$ and every $x \in \{0,1\}^n$,

$$\begin{aligned} x \in L &\implies \exists y \in \{0,1\}^{t(n)} \; \mathbf{Pr}_{r \in \{0,1\}^{t(n)}} [A(x, y, r, \alpha(r))] \geq 2/3, \\ x \notin L &\implies \forall y \in \{0,1\}^{t(n)} \; \mathbf{Pr}_{r \in \{0,1\}^{t(n)}} [A(x, y, r, \alpha(r))] \leq 1/3, \end{aligned}$$

and $|\alpha(r)| = a(n)$ for $|r| = t(n)$.

We will often use the following diagonalization result by Kannan [Kan82].

**Lemma 2.3** ([Kan82]). *For any $L \in \mathsf{PH}$ and any $k \geq 1$, $\mathsf{PH} \not\subset \text{io-}\mathsf{SIZE}^{L}[O(n^k)]$.*

We also need the following well-known Karp-Lipton Theorem.

**Theorem 2.4** ([KL80; KL82]). *If $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$, then $\mathsf{PH} = \Sigma_2^p$.*

## 2.3 Compressible-Counterexample Hierarchy Theorem

The classic Deterministic Time Hierarchy Theorem of Hartmanis and Stearns [HS65] implies an almost-everywhere hierarchy against sublinear advice inside $\mathsf{P}$, i.e.,

$$\mathsf{DTIME}[n^{b+1}] \not\subseteq \text{io-}\mathsf{DTIME}[n^b]/o(n).$$

In fact, it implies more: There is a "hard" language in $\mathsf{DTIME}[n^{b+1}]$ and a simple efficient "refuter" algorithm that is able to point out an error input for any given candidate $n^b$-time TM with advice, and, moreover, this error input is of a very special form that is highly compressible.

**Lemma 2.5** (Almost Everywhere P Hierarchy with compressible counterexamples). *There is a language $H_b \in \mathsf{DTIME}[n^{b+1}]$ satisfying the following:*

- COUNTEREXAMPLES: *Every candidate $n^b$-time TM $M$ with advice $\alpha = \{\alpha_n\}_{n\in\mathbb{N}}$, for $|\alpha_n| \in o(n)$, that tries to compute $H_b$ will make a mistake an $n$-bit input*

$$x_{error} = \langle M \rangle \circ \alpha_n \circ \pi,$$

*where $\pi \in \{0\}^*$ is a padding string whose length is chosen so that $|x_{error}| = n$, for all sufficiently large input lengths $n \geq 1$.*

- COMPRESSIBILITY OF COUNTEREXAMPLES: *The counterexample input $x_{error} \in \{0,1\}^n$ is efficiently compressible to $|\alpha_n| + O(1)$ bits by dropping the padding $\pi$, and can be efficiently reconstructed from $\langle M \rangle \circ \alpha_n$ by adding back the padding $\pi$ of appropriate length.*[16]

*Proof sketch.* Define $H_b$ to be decidable by the following TM $A$: "On input $x \in \{0,1\}^n$, try to interpret $x = \langle M \rangle \circ \alpha_n \circ \pi$ for some TM $M$ and strings $\alpha_n \in \{0,1\}^*$ and $\pi \in \{0\}^*$. If not possible, then reject. Otherwise, simulate TM $M$ with advice $\alpha_n$ on input $x$ for at most $n^b$ time steps, and accept iff $M$ rejects." □

## 2.4 Definitions of Circuit Uniformity

A family $\{C_n\}_{n\geq 1}$ of $n$-input circuits is called uniform if there is a uniform algorithm that, given $n$, computes some fixed binary encoding $\langle C_n \rangle$ of the circuit $C_n$. It is common to use the *direct connection* encoding of circuits: $\langle C_n \rangle_i = 1$ iff $i$ encodes a triple $(g, h, r)$ such that $g$ and $h$ gate names (indices), $r$ is the type of $g$ (AND/OR/NOT/INPUT/OUTPUT, and in case of INPUT, which of the $n$ input bits $g$ is), and $h$ is a gate feeding in to $g$ in case the type $r$ is not INPUT. Note that if $C_n$ has $s$ gates, then the direct-connection encoding of $C_n$ is of length $|\langle C_n \rangle| \leq \exp(3 \log s) \leq s^3$.

**Remark 2.6.** *For most of the paper, we need just two properties from binary encodings $\langle C_n \rangle$ of $n$-input circuits $C_n$ on $s(n)$ gates:*

- $|\langle C_n \rangle| \leq \mathsf{poly}(s(n))$*, and*
- *there is a polytime algorithm for the circuit value problem: given $\langle C_n \rangle$ and $x \in \{0,1\}^n$ as inputs, compute the value $C_n(x)$.*

There are two ways to define efficient uniformity: ask for the string function $F(n) = \langle C_n \rangle$ to be efficiently computable (say in time polynomial in $n$), or ask for the Boolean bit function of the encoding $f(n,i) = \langle C_n \rangle_i$ to be efficiently computable.

The first way is about the search problem: given $n$, find a circuit $C_n$. It can be defined using uniform *function* complexity classes as follows.

**Definition 2.7** (FC-uniformity)**.** For a uniform complexity class FC of string functions, a language $L$ is in

$$\mathsf{FC}\text{-uniform } \mathsf{SIZE}[s(n)]$$

if there exists an FC function $F$ such that $F(1^n)$ outputs a string $\langle C_n \rangle$ encoding some $n$-input circuit $C_n$ on at most $s(n)$ gates such that $C_n$ correctly computes $L \cap \{0,1\}^n$.

Before giving the second way of defining uniformity, we define padded Direct Connection Language (DCL). Fix some canonical (direct connection) encoding $\langle C_n \rangle$ of circuits $C_n$.

[16]This is under the assumption that the length $n$ is known, which will be the case in our applications. In general, however, we can always add $n$ in binary to the compressed image, getting $\langle M \rangle \circ n \circ \alpha_n$, of length $|\alpha_n| + 2 \log n + O(1)$.

**Definition 2.8** (padded DCL)**.** Let $C = \{C_n\}_{n\geq 1}$ be a circuit family, and let $pad(n) \geq 0$. Define the padded Direct Connection Language for $C$ as follows:

$$pad(n)\text{-}\mathsf{DCL}(C) = \{(n, 1^{pad(n)}, i) \mid \langle C_n \rangle_i = 1\},$$

where $n$ is given in binary, and the padding $pad(n)$ in unary.

If $0\text{-}\mathsf{DCL}(C) \in \mathsf{P}$, we say that the circuit family $C$ is polylogtime-uniform. If $n\text{-}\mathsf{DCL}(C) \in \mathsf{P}$, we say that $C$ is $\mathsf{P}$-uniform. More generally, we have the following definition of $\mathsf{C}$-uniformity for any uniform complexity class $\mathsf{C}$.

**Definition 2.9** ($\mathsf{C}$-uniformity)**.** For a uniform complexity class $\mathsf{C}$, a language $L$ is in

$$\mathsf{C}\text{-uniform } \mathsf{SIZE}[s(n)]$$

if there exists a family $C = \{C_n\}_{n\geq 1}$ of $n$-input circuits computing $L \cap \{0,1\}^n$, with each $C_n$ containing at most $s(n)$ gates, such that $n\text{-}\mathsf{DCL}(C) \in \mathsf{C}$.

**Remark 2.10.** *It is easy to see that* $\mathsf{P}$*-uniformity is the same as* $\mathsf{FP}$*-uniformity. However, a difference emerges for function classes where the string computed by a function is not uniquely determined by the input. For example, for the class* $\mathsf{FNP}$ *of* $\mathsf{NP}$ *search problems,* $F(1^n)$ *may output different circuit descriptions* $\langle C_n \rangle$ *of correct small circuits for a given language* $L$ *on different accepting computation branches. Similarly, for randomized search classes like* $\mathsf{FZPP}$ *or* $\mathsf{FBPP}$*, the final output string depends on the actual randomness used by the algorithm. Finally, for* $F(1^n)$ *defined by a learning algorithm with Equivalence Queries (see Definition 2.12 below), the resulting correct output* $\langle C_n \rangle$ *also depends on the particular answers to the equivalence queries.*

**Definition 2.11** (Equivalence Query (EQ))**.** Given a search problem to produce a size at most $s(n)$ circuit $C_n$ in some circuit class $\Lambda$ for some Boolean $n$-variate function $f$, an *equivalence query (EQ) for $f$* is defined as follows: Given a circuit $C'_n \in \Lambda$ of size at most $s(n)$ (a "candidate" circuit for $f$), the output of the EQ is either "yes" if $\forall x \in \{0,1\}^n \; C'_n(x) = f(x)$ or a *counterexample:* an $n$-bit string $x_{error}$ such that $C'_n(x_{error}) \neq f(x_{error})$. The choice of the counterexample is arbitrary.

By allowing a bounded number of proper equivalence queries, we generalize the notion of $\mathsf{FC}$-uniformity as follows.

**Definition 2.12** ($\mathsf{LEARN}$-uniformity)**.** For a uniform function class $\mathsf{FC}$, a language $L$ is in

$$\mathsf{FC}\text{-}\mathsf{LEARN}^{\mathsf{EQ}[r(n)]}\text{-uniform } \mathsf{SIZE}[s(n)]$$

if there exists an EQ-learning algorithm $F \in \mathsf{FC}$ such that, $F(1^n)$ asks at most $r(n)$ equivalence queries for $L \cap \{0,1\}^n$, using candidate circuits in $\mathsf{SIZE}[s(n)]$, and finally outputs a description of an $s(n)$-gate circuit $C_n$ computing $L \cap \{0,1\}^n$. (For the case $\mathsf{FC} = \mathsf{FP}$, we will write $\mathsf{LEARN}$ without the prefix $\mathsf{FP}$.)

**Remark 2.13.** *Note that* $\mathsf{LEARN}^{\mathsf{EQ}[0]}$*-uniformity is exactly the* $\mathsf{FP}$*-uniformity defined above. On the other hand, having one equivalence query is strictly more powerful than having zero equivalence queries! See Proposition A.1 in Section A of the appendix for details.*

We further generalize all the uniformity notions defined above by allowing the uniformity algorithms (in $\mathsf{C}$ or $\mathsf{FC}$) some amounts of non-uniform advice. We use the standard complexity-theoretic notation "$/a(n)$" to indicate that we allow $a(n)$ bits of advice for inputs of length $n$.

**Randomized LEARN Uniformity.** When we allow a *randomized* algorithm to interact with the $\mathsf{EQ}$ oracle, different definitions of randomized $\mathsf{LEARN}$ uniformity are possible. This happens because a query to the $\mathsf{EQ}$ oracle might admit more than one valid answer. Consequently, the success of a learner no longer depends only on the choice of its random string. Note that, since the learning algorithm can check with the $\mathsf{EQ}$ oracle before committing to the final answer, the randomized $\mathsf{LEARN}$ uniformity model can be assumed to be *zero error* by the addition of a final EQ query.

- $\mathsf{FZPP}^{\mathsf{rf}}\text{-}\mathsf{LEARN}^{\mathsf{EQ}}_{\delta}$. A more restrictive definition is to say that one can fix the randomness of the learning algorithm $A$ first, so that, with probability $\geq \delta(n)$ over the choice of randomness $r \in \{0,1\}^{\mathsf{poly}(n)}$, we obtain a deterministic algorithm $A_r$ that correctly solves the learning task, for all possible sequences of valid answers from the $\mathsf{EQ}$ oracle. Note that here the *same* randomness must be used for *every* possible sequence of $\mathsf{EQ}$ answers. We will use $\mathsf{FZPP}^{\mathsf{rf}}\text{-}\mathsf{LEARN}^{\mathsf{EQ}}$ to denote this model, where $\mathsf{rf}$ stands for "Randomness First".

- $\mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{EQ}}_{\delta}$. The most natural definition is to allow the randomized learning algorithm to use fresh randomness after each interaction with the $\mathsf{EQ}$ oracle. We model this computation by an algorithm $A$ that during the $i$-th stage of its computation, for $1 \leq i \leq r(n) + 1$, has access to input $(1^n, w_1, z_1, \ldots, w_{i-1}, z_{i-1}, \boldsymbol{w_i})$, where $w_j, z_j$ encode previous choices of random strings and answers provided by the oracle, respectively, and $\boldsymbol{w_i}$ is a fresh random string. Consequently, the execution of $A(1^n)$ can be viewed as a rooted tree $\mathcal{T}_n$, where each maximal path starting from the root is described by a string $(w_1, z_1, \ldots, w_r, z_r, w_{r+1})$ and fully specifies the circuit produced by $A$.

  The success probability of $A(1^n)$ is defined *recursively* via a function $\gamma_n$. (For simplicity, we assume that in each path exactly $r = r(n)$ queries are made by $A(1^n)$ before it outputs a final circuit.) For a terminal node $v = (w_1, z_1, \ldots, w_r, z_r, w_{r+1})$, we let $\gamma_n(v) = 1$ if the output circuit $C_v$ is correct, and $\gamma_n(v) = 0$ otherwise. In general, there are two cases to consider. For an internal node of the form $v = (w_1, z_1, \ldots, w_i, z_i)$, we let $\gamma_n(v) = \mathbb{E}[\gamma_n(v, \boldsymbol{w_{i+1}})]$. Finally, for an internal node $v$ of the form $v = (w_1, z_1, \ldots, w_i)$, we let $\gamma_n(v) = \min_{z_v} \gamma_n(v, z_v)$, where $z_v$ ranges over the possible answers (counterexamples or "correct") to the $i$-th query $C_v$. We define the success probability of $A(1^n)$ as $\gamma_n(\varepsilon)$, where $\varepsilon$ denotes the top node of $\mathcal{T}_n$, and require that $\gamma_n(\varepsilon) \geq \delta(n)$.

When the success probability $\delta$ is omitted, we tacitly assume $\delta = 3/4$.[17]

Crucially, the definition of $\mathsf{FZPP}\text{-}\mathsf{LEARN}^{\mathsf{EQ}}$ guarantees that if fresh randomness is employed in each stage, no matter the answers returned by the oracle, provided that they are always correct we will obtain a correct circuit with probability at least $\gamma_n(\varepsilon)$. (This can be established by a simple inductive argument on $r$.) In some cases, when lower bounding the success probability of a concrete algorithm $A$, it might be possible to argue that, in any "good" partial path $(w_1, z_1, \ldots, w_{i-1}, z_{i-1})$, with probability at least $1 - \delta$ over $\boldsymbol{w_i}$, a "good" random choice is made. In this case, one can lower bound the overall success probability $\gamma(\varepsilon)$ by $(1-\delta)^{r+1} \geq 1 - (r+1) \cdot \delta$. Note that this is useful if we can guarantee $\delta \leq 1/3(r+1)$. For instance, a general setting where such an analysis is possible is when the algorithm $A$ is obtained by the obvious *randomized simulation* of a *deterministic* algorithm $B$ that has access to equivalence queries and moreover can make queries to an oracle computing a language in $\mathsf{BPP}$.

[17]One way to interpret the two definitions is that in randomness-first learners, the choice of random string is *public* and made in advance (i.e., the learner behaves deterministically after that). On the other hand, in the general learning model, the randomness used by the learner is in a sense *private*, as the learner can use a "fresh" choice of randomness after each oracle answer is provided, and the oracle cannot anticipate its future decisions.

## 2.5 Bounded Arithmetic and Witnessing Theorems

We will use two-sorted (second-order) theories of bounded arithmetic as in [CN10] (using the second-order vocabulary introduced by [Zam96]); for the equivalent first-order theories, see, e.g., [Bus86; Kra95; Kra19]. In such two-sorted theories, one sort is for natural numbers, usually denoted by lower-case letters $i, j, n, m, \ldots$, and the other sort is for finite binary strings (sets of numbers), usually denoted by upper-case letters $X, Y, \ldots$. The only available primitive operations on strings are the string length, $|X|$, and the bit predicate $X(i)$. The numbers serve as indices for the strings, while the strings are the main inputs for Boolean devices (Turing machines or circuits) usually considered in complexity theory; when numbers are used as inputs for Boolean devices, they are assumed to be presented in unary notation. Number functions, denoted by lower-case letters $f, g, \ldots$, output numbers; string functions, denoted by upper-case letters $F, G, \ldots$, output strings; both kinds of functions may have any combination of number and string inputs.

A formula $\varphi$ in a vocabulary $\mathcal{L}$ is a $\Sigma_i^B(\mathcal{L})$-formula if it has at most $i$ alternating blocks of bounded string quantifiers $\exists\forall\exists\ldots$, and all number quantifiers are bounded; when the vocabulary $\mathcal{L}$ is clear from the context, we will drop its explicit mention. For example, a $\Sigma_0^B$-formula has no string quantifiers, but may have bounded number quantifiers. We denote by $\mathcal{T} \vdash \varphi$ the statement that theory $\mathcal{T}$ *proves* formula $\varphi$; and by $\mathbb{N} \vDash \varphi$ the statement that $\varphi$ is true in the standard model $\mathbb{N}$ of natural numbers.

We will primarily need the theories $\mathsf{VPV}^1$ and $\mathsf{V}^1$ (isomorphic to the first-order theories $\mathsf{PV}_1$ and $\mathsf{S}_2^1$, respectively), and their approximate-counting extensions with dual Weak Pigeonhole Principles, $\mathsf{VPV}^1 + \mathsf{dWPHP}$ and $\mathsf{V}^1 + \mathsf{dWPHP}$ (isomorphic to the first-order theories $\mathsf{APC}_1 = \mathsf{PV}_1 + \mathsf{dWPHP}$ and $\mathsf{S}_2^1 + \mathsf{dWPHP}$, respectively). We recall that each theory postulates a slightly different version of the axiom $\mathsf{dWPHP}(F)_m^n$, i.e., the corresponding axiom scheme can differ with respect to the relation between the parameters $n$ (which controls the number of pigeons) and $m(n)$ (which controls the number of holes). We review the necessary axioms in Section 4 when discussing each theory.

For our purposes, the most important features of these systems of bounded arithmetic are the concomitant *witnessing theorems*. Informally, these are results saying that if one can prove the existence of a "good" object within a system like $\mathsf{VPV}^1$ or $\mathsf{V}^1$, then one can also extract an *efficient algorithm* that will construct (witness) such a "good" object. Depending on the complexity of the formula defining the "goodness" condition (and on the system of bounded arithmetic used), one may get a polytime witnessing algorithm (cf. Buss's Witnessing Theorem 2.14 below) or a polytime learning algorithm in the model of exact learning with equivalence queries (cf. KPT Witnessing Theorems 2.16 and 2.17 below).

**Theorem 2.14** (Buss's Witnessing for $\mathsf{V}^1$ [Bus86]). *Suppose that, for a $\Sigma_1^B$-formula $\varphi$,*

$$\mathsf{V}^1 \vdash \forall X\, \exists Y \;\; \varphi(X, Y).$$

*Then there is an* $\mathsf{FP}$ *algorithm $F$ such that $\mathbb{N} \vDash \forall X\; \varphi(X, F(X))$.*

The following analogue of Buss's Witnessing for $\mathsf{VPV}^1$ can be proved using the Herbrand Theorem (see, e.g., [CN10, Corollary VIII.2.5]).

**Theorem 2.15** (Witnessing for $\mathsf{VPV}^1$). *Suppose that, for a $\Sigma_0^B$-formula $\varphi$,*

$$\mathsf{VPV}^1 \vdash \forall X\, \exists Y \;\; \varphi(X, Y).$$

*Then there is an* $\mathsf{FP}$ *algorithm $F$ such that $\mathbb{N} \vDash \forall X\; \varphi(X, F(X))$.*

The following is a version of the classical Herbrand Theorem (see, e.g., [CN10, Theorem VIII.6.1] for the proof).

**Theorem 2.16** (KPT Witnessing [KPT91]). *Let $T$ be an universal theory with vocabulary $\mathcal{L}$. Suppose that, for a $\Sigma_0^B(\mathcal{L})$-formula $\varphi$,*

$$T \vdash \forall X\ \exists Y\ \forall Z\ \ \varphi(X,Y,Z).$$

*Then there exist a constant $k \geq 1$ and a sequence $T_1, \dots, T_k$ of $\mathcal{L}$-string-terms such that*

$$T \vdash \forall X\ \forall \vec{Z}\ \ [\varphi(X, T_1(X), Z_1) \vee \varphi(X, T_2(X, Z_1), Z_2) \vee \cdots \vee \varphi(X, T_k(X, Z_1, \dots, Z_{k-1}), Z_k)]\,.$$

Theorem 2.16 applies to every theory whose axioms are universally quantified formulas. In particular, it applies to the universal theory $\mathsf{VPV}^1$, yielding a polytime learning algorithm (student) with constantly many counterexample queries (asked of the teacher).

Since $\mathsf{V}^1$ is not a universal theory, we cannot apply Theorem 2.16 directly to get KPT Witnessing for it. However, it is possible to derive a KPT-style Witnessing Theorem for $\mathsf{V}^1$, using Theorem 2.14.

**Theorem 2.17** (KPT Witnessing for $\mathsf{V}^1$ [Kra92; Pud92]). *Suppose that, for a $\Sigma_1^B$-formula $\varphi$,*

$$\mathsf{V}^1 \vdash \forall X\ \exists Y\ \forall Z(|Z| \leq |X|)\ \ \varphi(X,Y,Z).$$

*Then there is an* $\mathsf{FP}$ *algorithm $A$ such that*

$$\mathbb{N} \vDash \forall X\ \forall Z(|Z| \leq |X|)\ \ \varphi(X, A(X), Z),$$

*assuming $A$ has access to the counterexample oracle $O(X,Y)$, which returns a string $Z \leq |X|$ such that $\neg\varphi(X,Y,Z)$ if such a counterexample $Z$ exists, or outputs "yes" if $Y$ is good for all $Z$s.*[18]

KPT Witnessing Theorems exist also for $\mathsf{VPV}^1 + \mathsf{dWPHP}$ and $\mathsf{V}^1 + \mathsf{dWPHP}$; see Section 4 for the statements and proofs.

Finally, we will also use Parikh's theorem, which can be viewed as a precursor to the witnessing theorems for bounded arithmetic stated above. This theorem applies to any polynomial-bounded theory (extending $\mathsf{I}\Delta_0$, and axiomatized by bounded formulas, where each function in the vocabulary is polynomially bounded). In particular, Parikh's theorem applies to all theories considered in our paper. When such a theory proves $\forall X\ \exists Y\ \varphi(X,Y)$, we can bound the length of the existentially quantified string variable $Y$ by a polynomial in $X$.

**Theorem 2.18** (Parikh's Theorem [Par71]). *Let $T$ be a polynomial-bounded theory, and $\varphi(\vec{X}, Y)$ a bounded formula with all free variables displayed. If $T \vdash \forall\vec{X}\exists Y\ \varphi(\vec{X}, Y)$, then there is a term $t$ involving only variables in $\vec{X}$ such that $T \vdash \forall\vec{X}\exists Y(|Y| \leq t(\vec{X}))\ \varphi(\vec{X}, Y)$.*

We will be using KPT Witnessing in connection with circuit search problems: given $n \in \mathbb{N}$, find an $n$-input circuit $C$ (of small size) that agrees with a given function $f$ on all inputs $Z \in \{0,1\}^n$. Under an appropriate formalization, if the *existence* of small circuits computing $f$ can be established in $\mathsf{VPV}^1$ (resp. $\mathsf{V}^1$), then we get from the corresponding KPT Witnessing Theorem that a small circuit $C$ for $f$ can be learned in polytime (randomized polytime for the theories with $\mathsf{dWPHP}$) with a constant (resp. polynomial) number of equivalence queries. That is, from a proof of a circuit upper bound within an appropriate theory, we get (randomized) $\mathsf{LEARN}^{\mathsf{EQ}}$-uniformity.

[18]Such an algorithm $A$ is also called a *counterexample computation* [Kra95].

## 3 LEARN-Uniform Circuit Lower Bounds

### 3.1 Deterministic LEARN Uniformity for P

Here we show the following

**Theorem 3.1.** *For any constant $k \geq 1$, we have*

$$\mathsf{P} \not\subset \mathsf{LEARN}^{\mathsf{EQ[const]}}\text{-uniform } \mathsf{SIZE}[O(n^k)].$$

As our techniques relativize, we also get the following extension of this result. Note that an oracle to a language $L$ is simultaneously provided to the polynomial-time algorithm computing the hard problem, the uniform algorithm that attempts to produce a circuit, and the output circuit. Additionally, equivalence queries can be made using circuits that contain $L$-oracle gates.

**Theorem 3.2** (Relativized version of Theorem 3.1)**.** *Let $L$ be an arbitrary language. For any constant $k \geq 1$, we have*

$$\mathsf{P}^L \not\subset \mathsf{FP}^L\text{-}\mathsf{LEARN}^{\mathsf{EQ[const]}}\text{-uniform } \mathsf{SIZE}^L[O(n^k)].$$

#### 3.1.1 Base Case: Zero EQs

As a warm-up, we prove the special case of 0 equivalence queries, which is a restatement of a result from [SW14] since $\mathsf{LEARN}^{\mathsf{EQ[0]}}$-uniformity is the same as $\mathsf{FP}$- and $\mathsf{P}$-uniformity.

**Theorem 3.3** (Implicit in [SW14])**.** *For any constant $k \geq 1$, we have*

$$\mathsf{P} \not\subset \left(\mathsf{LEARN}^{\mathsf{EQ[0]}}/n^{1/(2k)}\right)\text{-uniform } \mathsf{SIZE}[O(n^k)].$$

*Proof.* The proof is by contradiction. Suppose $\mathsf{P} \subset \left(\mathsf{P}/n^{1/(2k)}\right)$-uniform $\mathsf{SIZE}[O(n^k)]$ for some constant $k \geq 1$. We use the uniformity assumption to argue that then there is a constant $k' \geq 1$ such that $\mathsf{P} \subseteq \mathsf{TIME}[n^{k'}]/o(n)$. Finally, we appeal to the classical Time Hierarchy Theorem to get a contradiction. We give more details next.

1. PICK A "HARD" LANGUAGE: Consider $H_b \in \mathsf{P}$ from Lemma 2.5 for $b > k$ to be determined.

2. USE UNIFORMITY: Since $H_b \in \mathsf{P}$, our assumption implies that there is a circuit family $C = \{C_n\}_{n\geq 1}$ of $dn^k$-size circuits $C_n$ computing $H_b$, for some constant $d \geq 0$, such that

$$n\text{-}\mathsf{DCL}(C) \in \mathsf{P}/n^{1/(2k)},$$

   where

$$n\text{-}\mathsf{DCL}(C) = \{(n, 1^n, i) \mid \langle C_n \rangle_i = 1\}.$$

3. PAD DOWN: For $\varepsilon = 1/(2k)$, we can use the same advice-taking polytime algorithm for $n$-$\mathsf{DCL}$ as above to also get that

$$n^\varepsilon\text{-}\mathsf{DCL}(C) \in \mathsf{P}/m,$$

   where

$$n^\varepsilon\text{-}\mathsf{DCL}(C) = \{(n, 1^{n^\varepsilon}, i) \mid \langle C_n \rangle_i = 1\}$$

   and its input length is $m \leq \log n + n^\varepsilon + 3k \log n \leq 2n^\varepsilon$.

4. COMPRESS $C$: Since $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$ implies that $\mathsf{P}/n \subset \mathsf{SIZE}[O(n^k)]$, we get that there is a family $D = \{D_m\}_{m\geq 1}$ of $O(m^k)$-size circuits $D_m$ for the language $n^\varepsilon$-$\mathsf{DCL}(C)$. That is,

$$D_m(n, 1^{n^\varepsilon}, i) = \langle C_n \rangle_i,$$

for all $1 \leq i \leq 3k \log n$. Note that each $D_m$ has at most $O((2n^\varepsilon)^k) \leq O(\sqrt{n})$ gates.

5. USE $D$ AS ADVICE TO SPEED UP $H_b$: Each $D_m$ can be encoded as a binary string of length $d(n) = O(\sqrt{n}(\log n))$. We will use these $d(n)$ bits of advice in the following "evaluator" TM $\mathcal{E}$ that computes $H_b$ on $n$-bit inputs.

   Given as advice a description of $D_m$, TM $\mathcal{E}$ on input $x \in \{0,1\}^n$ does the following:

   (a) Evaluate $D_m(n, 1^{n^\varepsilon}, i)$ over all $1 \leq i \leq 3k \log n$ to recover $\langle C_n \rangle$.

   (b) Evaluate $C_n(x)$ and output the result.

6. DIAGONALIZE: It is clear that, with correct advice, the TM $\mathcal{E}/d(n)$ correctly computes $H_b$ on all $n$-bit strings. The runtime of $\mathcal{E}$ is at most $O(n^{3k} \cdot n)$ to construct $\langle C_n \rangle$, plus $O(n^{2k})$ to evaluate $C_n(x)$.[19] Thus the overall time of $\mathcal{E}/d(n)$ at most $n^{4k}$. This contradicts Lemma 2.5 since $d(n) \in o(n)$.

□

Examining the proof of Theorem 3.3 more closely, we see that we've actually proved the following stronger statement.

**Lemma 3.4.** *For any constants $k \geq 1$ and $b \geq 4k$, at least one of the following must be true:*

1. $\mathsf{P} \not\subset \mathsf{SIZE}[O(n^k)]$,
2. $H_b \notin \left(\mathsf{P}/n^{1/(2k)}\right)$ *-uniform* $\mathsf{SIZE}[O(n^k)]$.

### 3.1.2 Induction: Eliminating One EQ

So far we haven't used the compressibility of counterexamples in the Time Hierarchy Theorem of Lemma 2.5. We'll use this property next when we show how to eliminate equivalence queries one by one.

**Lemma 3.5** (EQ Elimination via Extra Advice)**.** *Suppose, for some constants $k \geq 1$ and $b \geq 4k$,* $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$ *and*

$$H_b \in \left(\mathsf{LEARN}^{\mathsf{EQ}[r]}/a(n)\right)\text{-uniform } \mathsf{SIZE}[O(n^k)],$$

*where $a(n) = n^\delta$ for some $0 < \delta < 1/(2k)$, and $r \geq 1$ is arbitrary. Then*

$$H_b \in \left(\mathsf{LEARN}^{\mathsf{EQ}[r-1]}/a'(n)\right)\text{-uniform } \mathsf{SIZE}[O(n^k)],$$

*where $a'(n) \leq (c \log n) \cdot a(n)^k$, for some universal constant $c > 0$.*

[19]For standard descriptions of $s$-gate circuits, the circuit can be evaluated on any given input in time $O(s^2)$ (see, e.g., [Vol99, Theorem 2.15]).

*Proof.* Let $C = \{C_n\}_{n\geq 1}$ be the circuit family of $O(n^k)$-gate circuits that the learning algorithm makes the first equivalence query on to find out if $C$ correctly computes $H_b$. Since the first EQ is computable in $\mathsf{FP}/a(n)$, we get that $n$-$\mathsf{DCL}(C) \in \mathsf{P}/a(n)$.

We next argue that such a uniform circuit family $C$ cannot compute $H_b$, and moreover, we will find a compressible counterexample to this first equivalence query. We follow the proof structure of Theorem 3.3 above.

1. PAD DOWN: We can use the same advice-taking polytime algorithm for $n$-$\mathsf{DCL}$ to also get that
$$a(n)\text{-}\mathsf{DCL}(C) \in \mathsf{P}/m,$$
where
$$a(n)\text{-}\mathsf{DCL}(C) = \{(n, 1^{a(n)}, i) \mid \langle C_n\rangle_i = 1\}$$
and its input length is $m \leq \log n + a(n) + 3k \log n \leq 2a(n)$.

2. COMPRESS $C$: Since $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$ implies that $\mathsf{P}/n \subset \mathsf{SIZE}[O(n^k)]$, we get that there is a family $D = \{D_m\}_{m\geq 1}$ of $O(m^k)$-size circuits $D_m$ for the language $a(n)$-$\mathsf{DCL}(C)$. That is,
$$D_m(n, 1^{a(n)}, i) = \langle C_n\rangle_i,$$
for all $1 \leq i \leq O(k \log n)$. Each $D_m$ has at most $O((2a(n))^k) \leq O(n^{\delta k}) \leq O(\sqrt{n})$ gates, and so can be encoded as a binary string $\beta_n$ of length $d(n) = O(n^{\delta k}(\log n)) \in o(n)$.

3. EVALUATOR TM $\mathcal{E}$: Given as advice a description $\beta_n$ of $D_m$, TM $\mathcal{E}$ on input $x \in \{0,1\}^n$ does the following:

   (a) Evaluate $D_m(n, 1^{a(n)}, i)$ over all $1 \leq i \leq O(k \log n)$ to recover $\langle C_n\rangle$.

   (b) Evaluate $C_n(x)$ and output the result.

4. DIAGONALIZE: Since the runtime of $\mathcal{E}$ at most $n^{4k} \leq n^b$, it cannot compute $H_b$ by Lemma 2.5. Moreover, $\mathcal{E}$ with advice $\beta_n$ disagrees with $H_b$ on the input
$$x_{error} = \langle \mathcal{E}\rangle \circ \beta_n \circ \pi,$$
for some $\pi \in \{0\}^*$ of appropriate length so that $|x_{error}| = n$.

5. ELIMINATE THE EQUIVALENCE QUERY THROUGH ADVICE: Since, for the advice $\beta_n$, we have by construction that $(\mathcal{E}/\beta_n)(x) = C_n(x)$ for all $x \in \{0,1\}^n$, it follows that
$$C_n(x_{error}) \neq H_b(x_{error}).$$
We add to the advice of our learning algorithm the succinct encoding $\langle \mathcal{E}\rangle \circ \beta_n$ of $x_{error}$ to be used as the answer to the first equivalence query, and eliminate that query. The new learning algorithm now makes $(r-1)$ equivalence queries, and has advice of length $a(n) + O(a(n)^k(\log a(n))) \leq O(a(n)^k(\log n))$, as claimed.

Note that after we eliminate a query, the advice complexity increases, but the same learning algorithm is maintained. □

### 3.1.3 Finishing the Proof

Using Lemma 3.5 for equivalence query elimination, we can finish the proof of Theorem 3.1. We establish a slightly stronger result.

**Theorem 3.6.** *For any constants $k \geq 1$ and $b \geq 4k$, at least one of the following must be true:*

1. $\mathsf{P} \not\subset \mathsf{SIZE}[O(n^k)]$
2. $H_b \notin \mathsf{LEARN}^{\mathsf{EQ[const]}}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)]$.

*Proof.* Towards a contradiction, suppose that, for some constants $k \geq 1$, $b \geq 4k$, and $r \geq 0$, both $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$ and $H_b \in \mathsf{LEARN}^{\mathsf{EQ}[r]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)]$. Set $a_0(n) = n^\gamma$ for some $0 < \gamma \leq 1/(2k)$ to be determined. Imagine giving a (useless) advice string $1^{a_0(n)}$ to our assumed learning algorithm for $H_b$. Now apply Lemma 3.5 for $r$ rounds, getting a learning algorithm with no EQs, and advice size $a_r(n)$ satisfying the following recurrence: for every $1 \leq i \leq r$,

$$a_i(n) \leq (c \log n) \cdot a_{i-1}(n)^k,$$

assuming that $a_i(n) \leq n^{1/(2k)}$ for each $0 \leq i \leq r$. Solving the recurrence, we get that

$$a_r(n) \leq (c \log n)^{k^{r-1}+k^{r-2}+\cdots+k+1} \cdot a_0(n)^{k^r} \leq (c \log n)^{rk^{r-1}} \cdot a_0(n)^{k^r}.$$

Setting $\gamma = 1/(3k^{r+1})$ ensures that $a_r(n) < n^{1/(2k)}$, and we get a contradiction by Lemma 3.4. □

*Proof Sketch of Theorem 3.2.* This follows from the observation that all ingredients employed in the proof of Theorem 3.1 *relativize*. Crucially, note that the compressible counterexample hierarchy theorem (Lemma 2.5) admits a simple proof by simulation and diagonalization, and as such, providing oracle access to a fixed but arbitrary language $L$ does not affect the argument. The proof of Theorem 3.1 goes through in the presence of the oracle language $L$ because the involved algorithms and circuits have access to $L$, thereby the necessary simulations can be efficiently implemented via a straightforward adaptation of the original argument. Since this is rather standard, we omit the details. □

### 3.1.4 Beyond a Constant Number of EQs

While we seem unable to go beyond a constant number of EQs for $\mathsf{SIZE}[O(n^k)]$ for every $k \geq 1$, we can for the special case of $k = 1$, i.e., for $\mathsf{SIZE}[O(n)]$ and even for $\mathsf{SIZE}[O(n \cdot \mathsf{poly}(\log n))]$, as we show next.

**Theorem 3.7** (Super-constant number of EQs)**.** *For any constants $b \geq 8$ and $A \geq 1$, at least one of the following must be true:*

1. $\mathsf{P} \not\subset \mathsf{SIZE}[O(n \cdot (\log n)^A)]$
2. $H_b \notin \mathsf{LEARN}^{\mathsf{EQ}[r(n)]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n \cdot (\log n)^A)]$, *for any $r(n) \leq (\log n)/(10(A+1) \log \log n)$.*

*Proof.* Similarly to the proof of Theorem 3.6, we start with ("useless") advice of length $a_0(n) = n^{1/10}$, and apply Lemma 3.5 for $r(n)$ rounds. (Note that invoking this lemma a super-constant number of times is possible because we maintain the *same* learning algorithm after each query is eliminated.) For each $1 \leq i \leq r$, the advice size satisfies the recurrence

$$a_i(n) \leq (c \log n)^{A+1} \cdot a_{i-1}(n),$$

which solves to $a_r(n) \leq (c \log n)^{r(n)\cdot(A+1)} \cdot n^{1/10} \leq n^{(1/10)+((1.1)/10)} < n^{1/4}$, yielding a contradiction by Lemma 3.4 as $H_b \notin (\mathsf{P}/n^{1/4})\text{-}\mathsf{uniform}\ \mathsf{SIZE}(O(n^2))$. □

## 3.2 Deterministic LEARN Uniformity for NP

The proof of the following theorem uses some ideas of [CK07]; in particular, it generalizes the proof of Lemma B.2 (implicit in [CK07]) given in Appendix B.

**Theorem 3.8.** *If, for some function $1 \leq r(n) \leq \mathsf{poly}(n)$,*

$$\mathsf{SAT} \in \mathsf{LEARN}^{\mathsf{EQ}[r(n)]}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}],$$

*then $\mathsf{PH} \subset \mathsf{NP}/r(\mathsf{poly}(n))$. Hence, for every $k \geq 1$,*

$$\mathsf{NP} \not\subset \mathsf{LEARN}^{\mathsf{EQ}[n^{o(1)}]}\text{-uniform } \mathsf{SIZE}[O(n^k)].$$

*Proof.* For some $r = r(n)$, suppose we have a $\mathsf{LEARN}^{\mathsf{EQ}[r(n)]}$ algorithm $A$ that, on input $1^n$, constructs a $\mathsf{poly}(n)$-size circuit $C_n$ that correctly decides $\mathsf{SAT}$ on all $n$-bit input formulas. We assume that the learning algorithm, before outputting a correct circuit $C_n$, makes one last EQ for $C_n$ (getting the answer "yes" form the EQ oracle).

**Claim 3.9.** *Suppose $C$ is a candidate $\mathsf{SAT}$ circuit for formulas of size $n$ such that*

1. *$C$ is correct on all satisfiable input formulas, but*
2. *there exists some unsatisfiable formula $\psi$ such that $C(\psi) = 1$.*

*Then*

*either there exists a variable-free formula $\psi$ that evaluates to $0$ (i.e., is unsatisfiable), but $C(\psi) = 1$,*

*or there exists an unsatisfiable formula $\psi$ with some variable $x$ such that $C(\psi) = 1$, but $C(\psi[x = 0]) = C(\psi[x = 1]) = 0$.*

*Proof of Claims 3.9.* Suppose the conclusion is false. In other words, Items 1 and 2 above hold, but we have

($a$) For every variable-free formula $\psi$ that evaluates to 0 (i.e., $\psi$ is unsatisfiable), $C(\psi) = 0$.

($b$) For every unsatisfiable formula $\psi$ and for every free variable $x$ of $\psi$, if $C(\psi) = 1$ then either $C(\psi[x = 0]) = 1$ or $C(\psi[x = 1]) = 1$.

Now consider the unsatisfiable formula $\psi$ with $C(\psi) = 1$ provided by Item 2. Consider a free variable $x$ of $\psi$. By Item (b) above, there is a way to fix $x$ in $\psi$ to obtain a formula $\psi'$ such that $C(\psi') = 1$. Since $\psi$ is unsatisfiable, so is $\psi'$. Repeating this process until no free variables are left, we arrive at an unsatisfiable variable-free formula $\varphi$ (i.e., $\varphi$ evaluates to 0) such that $C(\varphi) = 1$. But this contradicts (a), which concludes the proof. $\square$

Defining the advice. As a thought experiment, imagine running $A(1^n)$ using the following strategy when answering EQs that $A$ makes with candidate $\mathsf{SAT}$ circuits $C^i$, $1 \leq i \leq r(n)$, assuming $C^i$ still doesn't solve $\mathsf{SAT}$ (otherwise, the EQ must be answered with a "yes"):

- if $C^i$ makes mistakes on some satisfiable formulas, then use one of these satisfiable formulas as an answer to the $i$th EQ;

- if $C^i$ is correct on all satisfiable formulas, but makes mistakes on some unsatisfiable formulas, then answer with an unsatisfiable formula $\psi$ of the form guaranteed to exist by Claim 3.9 above.

Consider the following $\mathsf{NP}$ algorithm $M$: "On input $1^n$ and a string $\alpha \in \{0,1\}^\ell Y^{r(n)-\ell}$ (i.e., a binary string, possibly followed with some string of $Y$s) for some $1 \leq \ell \leq r(n)$,

- for each $1 \leq i \leq \ell$ such that $\alpha_i = 1$, guess a formula $\varphi_i$ of size $n$, together with a satisfying assignment $w_i$ (such that $\varphi_i(w_i) = 1$),
- for each $1 \leq i \leq \ell$ such that $\alpha_i = 0$, guess a formula $\psi_i$ of size $n$ and, if $\psi_i$ has some variables, then also guess a variable name $x_i$ from among the variables of $\psi_i$,
- guess the transcript of the partial computation of $A$ on input $1^n$ that makes exactly $\ell + 1$ EQs, using the previously guessed formulas to answer each $i$th EQ, for $1 \leq i \leq \ell$: answer with $\varphi_i$, if $\alpha_i = 1$, or answer with $\psi_i$, if $\alpha_i = 0$,
- accept if
  - the partial transcript is valid (assuming the guessed EQ answers are correct),
  - letting $C^i$ be the candidate circuit produced by the learning algorithm for the $i$th EQ, for $1 \leq i \leq \ell$, check that, for all $1 \leq i \leq \ell$,
    - $C^i(\varphi_i) = 0$, if $\alpha_i = 1$,
    - ($\psi_i$ is a variable-free formula that evaluates to 0 and $C^i(\psi_i) = 1$) or ($C^i(\psi_i) = 1$ and $C^i(\psi_i[x_i = 0]) = C^i(\psi_i[x_i = 1]) = 0$), if $\alpha_i = 0$."

Note that for each $n$, there is at least one string $\alpha$ such that $M(1^n, \alpha)$ accepts (as we can use a run of $A(1^n)$ as in the thought experiment above, and define a string $\alpha$ based on that run).

Let $\alpha_n^*$ be the *lexicographically largest* string of length $r(n)$ such that $M(1^n, \alpha_n^*)$ accepts, where we define the order on the alphabet symbols $\{0, 1, Y\}$ as $1 > 0 > Y$. Let $\ell_n^*$ be the length of the binary $\{0,1\}^*$ prefix of $\alpha_n^*$ (i.e., $\alpha_n^*$ consists of a binary prefix of length $\ell_n^*$, followed by $Y$'s).

By Claim 3.9 and the discussion above, there is at least one accepting branch of $M$ on input $(1^n, \alpha_n^*)$. The crucial observation is the following.

**Claim 3.10.** *Every accepting branch (witness) $W$ of the NTM $M$ on input $(1^n, \alpha_n^*)$ contains a correct transcript of the computation of $A(1^n)$ with $\ell_n^*$ EQs, including a* correct $\mathsf{SAT}$ circuit $C^{\ell_n^*+1}$.

*Proof of Claim 3.10.* Indeed, consider such a transcript $W$. All EQs for the $i$s where $(\alpha_n^*)_i = 1$ are answered correctly (with satisfiable formulas on which $C^i$ says 0), by the definition of $M$.

For any EQ $i$ where $(\alpha_n^*)_i = 0$, if the corresponding circuit $C^i$ is correct on all satisfiable input formulas, it must be the case that $\psi_i$ is unsatisfiable. Indeed, since $W$ is an accepting branch of $M$ and $(\alpha_n^*)_i = 0$, we have that ($\psi_i$ is a variable-free formula that evaluates to 0 and $C^i(\psi_i) = 1$) or ($C^i(\psi_i) = 1$ and $C^i(\psi_i[x_i = 0]) = C^i(\psi_i[x_i = 1]) = 0$). If the former case happens, $\psi_i$ is obviously unsatisfiable. Otherwise, since by assumption $C^i$ is always correct on satisfiable formulas, if $\psi_i$ is satisfiable we must have that either $C^i(\psi_i[x_i = 0]) = 1$ or $C^i(\psi_i[x_i = 1]) = 1$. By assumption this does not hold, so we get that $\psi_i$ is unsatisfiable in this case as well. Hence answering the $i$th EQ with $\psi_i$, where $C^i(\psi_i) = 1$, is correct.

On the other hand, suppose there is some $i \leq \ell$ where $(\alpha_n^*)_i = 0$ such that $C^i$ makes mistakes on some satisfiable input formulas. Let $i^*$ be the first such index $i$. Then $\alpha_n^*$ is *not* the lexicographically largest string accepted by $M$. Indeed, there is some extension of the prefix $(\alpha_n^*)_{[1..(i^*-1)]}1$ that is also accepted by $M$.

We conclude that all the EQs are answered correctly (with satisfiable formulas as counterexamples whenever possible). Suppose the circuit $C^{\ell_n^*+1}$ produced after these $\ell_n^*$ EQs is still not a correct SAT circuit. If it makes mistakes on some satisfiable formulas, then some extension of the prefix $(\alpha_n^*)_{[1..\ell_n^*]}1$ is also accepted by $M$. If $C^{\ell_n^*+1}$ is correct on all satisfiable formulas, but not on all unsatisfiable ones, then some extension of the prefix $(\alpha_n^*)_{[1..\ell_n^*]}0$ is also accepted by $M$. In both cases, we get that $\alpha_n^*$ is *not* the lexicographically largest string accepted by $M$. A contradiction. □

Collapsing PH. We will assume the binary prefixes of $\alpha_n^*$ of lengths $\ell_n^*$ are given to us as advice. For any language $L \in \mathsf{coNP}$, defined by a formula $\eta(x) = \forall y \quad R(x, y)$, where $y$ has the length polynomial in the length of $x \in \{0,1\}^n$, and the predicate $R$ is in P. For each string $x$, define a propositional formula $\varphi_x(y')$ such that $\forall y' \quad \varphi_x(y') \iff \forall y \quad R(x, y)$. Such a formula can be constructed in $\mathsf{poly}(|x|)$-time using the Cook-Levin Theorem that SAT is NP-complete. Let $m(|x|) \in \mathsf{poly}(|x|)$ be the size of $\neg\varphi_x$, the negation of $\varphi_x$.

The following $\mathsf{NP}/r(m(n))$ algorithm $E$ will decide $L$: "On input $x \in \{0,1\}^n$, given the first $\ell_{m(n)}^*$ bits of $\alpha_{m(n)}^*$ as advice $\alpha$, of length at most $r(m(n))$,

- guess a string $W \in \{0,1\}^{O(t)}$, where $t = t(m(n))$ is the runtime of $M(1^{m(n)}, \alpha_{m(n)}^*)$,
- accept if
  - $W$ is an accepting branch of the computation of $M(1^{m(n)}, \alpha_{m(n)}^*)$,
  - the circuit $C_m$ contained within $W$, for the $(\ell_n^* + 1)$st EQ, is such that $C_m(\neg\varphi_x) = 0$."

For correctness, observe that $E$ accepts $x$ iff a circuit $C_m$ contained within an accepting computation of $M(1^m, \alpha_m^*)$ (implying that $C_m$ must be a correct SAT circuit by Claim 3.10) says that $\neg\varphi_x$ is unsatisfiable, which is equivalent to saying that $\forall y \quad R(x, y)$. Thus $E$ accepts $x$ iff $\eta(x)$, as required.

Hence we get that $\Sigma_2^p \subset \mathsf{NP}/r(\mathsf{poly}(n))$, implying the collapse $\mathsf{PH} = \Sigma_2^p \subset \mathsf{NP}/r(\mathsf{poly}(n))$.

Finally, for $r(n) \leq n^{o(1)}$, either $\mathsf{SAT} \notin \mathsf{LEARN}^{\mathsf{EQ}[n^{o(1)}]}$-uniform $\mathsf{SIZE}[\mathsf{poly}]$, or $\mathsf{PH} \subset \mathsf{NP}/n$. The latter implies, by Lemma 2.3, the existence of the required hard languages in $\mathsf{NP}/n$ and, hence, also in NP (by making the advice part of the input of the new NP machine). □

We can handle more EQs in case of Search-SAT circuit learning algorithms, rather than SAT circuit learning algorithms. A Search-SAT circuit is a circuit that, given an input formula $\varphi$, either outputs a satisfying assignment for $\varphi$, or says "NO". A correct Search-SAT circuit is the one that outputs a satisfying assignment on every satisfiable formula.

**Remark 3.11** (Search-SAT-EQ: The EQ oracle for Search SAT)**.** *In the following theorem, the* EQ *oracle for* Search-SAT *is the following: given a candidate* Search-SAT *circuit $C$, the* EQ *is answered "yes", if $C$ is correct (i.e., finds a satisfying assignment) on all satisfiable formulas; the* EQ *query is answered with a formula $\varphi$ and a satisfying assignment $w$ for $\varphi$ such that $C(\varphi)$ fails to find a satisfying assignment, otherwise. That is, the* Search-SAT-EQ *oracle provides not only a satisfiable formula $\varphi$ where $C$ makes a mistake, but also a proof that this formula is satisfiable (via a satisfying assignment $w$ for $\varphi$). (This requirement on the* EQ *oracle for* Search-SAT *comes from an actual learning algorithm one gets via KPT Witnessing (cf. Theorem 4.5).) For simplicity, when discussing* Search-SAT *we might simply say* EQ *oracle to refer to a* Search-SAT-EQ *oracle. This will always be clear from the context.*

**Theorem 3.12.** *If, for some function $1 \leq r(n) \leq \mathsf{poly}(n)$,*

$$\mathsf{Search\text{-}SAT} \in \mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[r(n)]}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}],$$

*then, for every $k \geq 1$,*

$$\mathsf{NP} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)].$$

*In particular, for all $k \geq 1$,* $\mathsf{Search\text{-}SAT} \notin \mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[n^{O(1)}]}\text{-uniform } \mathsf{SIZE}[n^k]$ *or* $\mathsf{NP} \not\subseteq \mathsf{LEARN}^{\mathsf{EQ}[n^{O(1)}]}\text{-uniform } \mathsf{SIZE}[n^k]$.

*Proof.* Note that, by definition, every candidate Search-SAT circuit produced by the learning algorithm for EQs is such that it is either correct on all inputs, or that there is a satisfiable formula on which the circuit fails to find a satisfying assignment. That is, the only possible counterexamples returned by EQs are satisfiable formulas. Hence, we can use the same argument as in Theorem 3.8 when defining the advice string $\alpha_n^*$, but now it suffices to use the alphabet $\{1, Y\}$, rather than $\{0, 1, Y\}$. To describe such an advice string $\alpha_n^* \in \{1\}^*\{Y\}^*$, it suffices to just specify the length of the $\{1\}^*$ prefix of $\alpha_n^*$, using at most $O(\log r(n))$ bits. For completeness, we give the full proof next.

STEP 1: DEFINING THE ADVICE. Let $0 \leq \ell(n) \leq r(n)$ be the *largest* number such that there is a run of the learning algorithm on input $1^n$ with $\ell(n)$ EQs satisfying the following: given counterexamples to the previous EQs, for each circuit $C^i$ produced by the learning algorithm for the $i$th EQ, $0 \leq i \leq \ell(n)$, there is a counterexample that is a satisfiable formula $\varphi_i$ (with a satisfying assignment $w_i$) such that $C^i(\varphi_i) = 0$, and the circuit $C^{\ell(n)+1}$ produced after these $\ell(n)$ EQs must be a *correct* Search SAT circuit.

STEP 2: COLLAPSING PH TO $\mathsf{NP}/O(\log n)$. Given the value $\ell(n)$ as advice, one can decide UNSAT in NP as follows:

"On input formula $\psi$ of size $n$, given $\ell = \ell(n)$ as advice,

1. nondeterministically guess size $n$ formulas $\varphi_1, \ldots, \varphi_\ell$ as well as $n$-bit assignments $w_1, \ldots, w_\ell$,
2. run the learning algorithm on $1^n$, answering each $i$th EQ for the produced circuit $C^i$ with formula $\varphi_i$ and assignment $w_i$, for $1 \leq i \leq \ell$, (rejecting if the learner makes fewer than $\ell$ EQs),
3. accept if
   - for all $1 \leq i \leq \ell$, $\varphi_i(w_i) = 1$,
   - for all $1 \leq i \leq \ell$, $C^i(\varphi_i) = 0$, and
   - for the circuit $C^{\ell+1}$ produced after $\ell$ EQs, $C^{\ell+1}(\psi) = 0$."

For correctness, observe that since $\ell$ is the *maximum* number of EQs that can be answered with satisfiable counterexamples before a Search-SAT circuit $C^{\ell+1}$ is produced by the learning algorithm that is correct on all SAT instances of size $n$. Hence, the circuit $C^{\ell+1}$ must be a correct circuit for SAT (on every non-deterministic path where the learner asks $\ell$ queries, and there is at least one such path). Thus, if $\psi$ is unsatisfiable, then there is a way to force $\ell$ EQs with some satisfiable counterexamples $\varphi_1, \ldots, \varphi_\ell$, getting a correct SAT circuit $C^{\ell+1}$ which will reject $\psi$. On the other hand, to incorrectly accept a satisfiable $\psi$, the nondeterministic algorithm above would need to force the learning algorithm to use $\ell$ EQs, but then the next circuit $C^{\ell+1}$ is guaranteed to be a correct SAT circuit which would always accept $\psi$.

Note that the advice $0 \leq \ell(n) \leq r(n)$ has size at most $a(n) = \lceil \log(r(n)+1) \rceil$ bits. Since UNSAT is coNP-complete under polytime reductions, we get that

$$\mathsf{coNP} \subset \mathsf{NP}/a(\mathsf{poly}(n)).$$

As a consequence, we also have $\Sigma_2^p \subseteq \mathsf{NP}/a(\mathsf{poly}(n))$. By Theorem 2.4, we get $\mathsf{PH} \subset \mathsf{NP}/a(\mathsf{poly}(n))$. By Lemma 2.3, for every $k \geq 1$, there is a language $L_k \in \mathsf{PH} \subset \mathsf{NP}/O(\log n)$ such that $L_k \notin$

io-$\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$. By a standard argument (making the advice a part of the input), we get a language $L'_k \in \mathsf{NP}$ such that $L'_k \notin$ io-$\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$. $\square$

## 3.3 Randomized LEARN Uniformity for ZPP

**Definition 3.13.** A language $L$ is in $\mathsf{ZPP}$-uniform $\mathsf{SIZE}[s(n)]$ if there exists a family $C = \{C_n\}_{n\geq 1}$ of $s(n)$-size circuits $C_n$ for $L$ such that the $n$-$\mathsf{DCL}(C) \in \mathsf{ZPP}$.

**Theorem 3.14.** *For any $k \geq 1$,*

$$\mathsf{ZPP} \not\subset \mathsf{ZPP}\text{-uniform } \mathsf{SIZE}[O(n^k)].$$

*Proof.* This follows by a straightforward adaptation of the proof of Theorem 3.21. $\square$

We introduce next a potentially larger family of languages.

**Definition 3.15** ($\mathsf{FZPP}$ Uniformity)**.** We say that a language $L$ is in

$$\mathsf{FZPP}\text{-uniform } \mathsf{SIZE}[s(n)]$$

if there is a probabilistic polynomial time algorithm $A$ such that for every $n$,

$$\Pr_A[A(1^n) \neq \perp] \geq 1/2,$$

and whenever $A(1^n) \neq \perp$, $A(1^n)$ outputs some $s(n)$-size circuit for $L \cap \{0,1\}^n$. Note that a different circuit might be produced for each non-"$\perp$" computation path.

Observe that changing the success probability $1/2$ in the definition above to any value $\delta \geq 1/\mathsf{poly}(n)$ does not affect the class of languages that admit $\mathsf{FZPP}$-uniform circuits.

We will need a result about (pseudodeterministic) pseudorandom generators (PRGs). We say that a probabilistic algorithm $A$ computes a function $f\colon \{0,1\}^* \to \{0,1\}^*$ in *pseudodeterministic polynomial time* if on every input $x$, $A$ runs in time $\mathsf{poly}(|x|)$ and outputs $f(x)$ with probability at least $2/3$. Similarly, we say that $f$ is computable in pseudodeterministic polynomial time with 1 bit of advice if $A$ requires 1 bit of advice per input length.

**Theorem 3.16** (Pseudodeterministic polynomial-time i.o.PRG with 1 bit of advice [LOS21])**.** *For each $\varepsilon > 0$ and $c, d \geq 1$, there is an infinitely often pseudorandom generator $G = \{G_n\}_{n\geq 1}$ mapping $n^\varepsilon$ bits to $n$ bits that is secure against $\mathsf{DTIME}(n^c)$ with error $1/n^d$ and computable in pseudodeterministic polynomial time with 1 bit of advice. More generally, $G$ is infinitely often secure against any ensemble $\mathfrak{D} = \{\mathcal{D}_n\}_{n\geq 1}$ of distributions $\mathcal{D}_n$ supported over circuits of size $\leq n^c$ and samplable in time $n^c$, in the sense that for infinitely many $n$, with probability at most $1/n^d$ over $C \sim \mathcal{D}_n$ we have that $C$ $1/n^d$-distinguishes $\mathcal{U}_n$ and $G_n(\mathcal{U}_{n^\varepsilon})$.*

We note that the argument from [LOS21] can be easily adapted to produce a generator that maps $n^\varepsilon$ bits to $n^c$ bits (instead of just $n$ bits). In our proof, we only need the following weaker form of the result.

**Corollary 3.17.** *Let $c \geq 1$, and let $\{C_n\}_{n\geq 1}$ be a $\mathsf{P}$-uniform sequence of circuits $C_n\colon \{0,1\}^{n^c} \to \{0,1\}$. For each $\varepsilon > 0$, there is a pseudorandom generator $G = \{G_n\}_{n\geq 1}$ mapping $n^\varepsilon$ bits to $n^c$ bits that is infinitely often secure against $\{C_n\}_{n\geq 1}$ and computable in pseudodeterministic polynomial time with 1 bit of advice. In other words, for infinitely many values of $n$, we have*

$$\left| \Pr_{x\in\{0,1\}^{n^c}}[C_n(x) = 1] - \Pr_{z\in\{0,1\}^{n^\varepsilon}}[C_n(G_n(z)) = 1] \right| \leq 1/n.$$

We will instantiate this generator against a $\mathsf{DTIME}[n^c]$-uniform sequence $\{C_n\}_{n\geq 1}$ of circuits $C_n\colon \{0,1\}^{\leq n^c} \to \{0,1\}$, where each $C_n$ has size at most $n^c$. Roughly speaking, this sequence of (deterministic) circuits is obtained from the computation of a randomized algorithm $B(1^n)$ on its random string $w \in \{0,1\}^{\leq n^c}$, which can be equivalently viewed as a deterministic computation on the input string $w$. After the constant $c$ is fixed, for every $\varepsilon > 0$ we can get a PRG $G_n$ mapping $n^\varepsilon$ bits to $n^c$ bits that fools infinitely many circuits $C_n$. Moreover, $G_n$ can be computed in probabilistic polynomial-time with 1 bit of advice.

**Theorem 3.18** (Weakness of zero-error uniformity)**.** *For every $k \geq 1$ and for $b \geq 10k$, at least one of the following statements must be true:*

1. $\mathsf{ZPP}/1 \not\subset \mathsf{SIZE}[O(n^k)]$*, or*
2. $H_b \notin \mathsf{FZPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)]$.

*In particular, for every $k \geq 1$ we have* $\mathsf{promise}\text{-}\mathsf{ZPP} \not\subset \mathsf{FZPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)]$.

*Proof.* Suppose the opposite. Then, for $b = 10k$, $H_b \in \mathsf{FZPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)]$. Let $A$ be a probabilistic polynomial time algorithm that witnesses this $\mathsf{FZPP}\text{-}\mathsf{uniform}$ circuit upper bound. We model $A(1^n, w)$ as a *deterministic* procedure that runs in time $n^c$ for some $c \geq 1$, where $w \in \{0,1\}^{n^c}$ is interpreted as the "random" string. For a choice of $w$ such that $A(1^n, w) \neq \perp$, we let $\langle C^n_w \rangle = A(1^n, w)$ be the description of a circuit $C^n_w$ of size $O(n^k)$ that is produced by $A(1^n, w)$. Note that each such circuit $C^n_w$ computes $H_b \cap \{0,1\}^n$. Also recall that $\Pr_w[A(1^n, w) \neq \perp] \geq 1/2$.

Set $\varepsilon = 1/(3k)$. We condition on whether there are witnesses $w$ such that $A(1^n, w) \neq \perp$ and $w$, when viewed as a truth table of a Boolean function on $(\log |w|)$ inputs, is computable by a Boolean circuit of size $O(n^\varepsilon)$. In both cases we will reach a contradiction via Lemma 2.5.

- Case 1: Easy witnesses for $A(1^n, \cdot)$ exist infinitely often. Suppose for infinitely many $n$, some $O(n^\varepsilon)$-size circuit $E$ exists whose truth table $w_E$ is such that $A(1^n, w_E) = \langle C^n_{w_E} \rangle \neq \perp$. Use the descriptions of such circuits $E$ as a sequence $\{\alpha_n\}$ of advice strings, where $|\alpha_n| \leq O(n^\varepsilon(\log n))$. Let $C = \{C_n\}$ be the sequence of $O(n^k)$-size circuits determined by these advice strings, i.e., $C_n = C^n_{w_E}$ for the circuit $E$ being describe by the advice string $\alpha_n$. For input lengths $n$, where there is no good advice string $\alpha_n$, we still get some circuit $C_n$, but this $C_n$ may not compute $H_b \cap \{0,1\}^n$.

  We get that the language
  $$n^\varepsilon\text{-}\mathsf{DCL}(C) \in \mathsf{P}/O(m(\log m)),$$
  where $m = \log n + n^\varepsilon + O(k \log n) \leq O(n^\varepsilon)$ is its input size. Indeed, the following advice-taking polytime TM computes this language:

  > "On input $n$, $1^{n^\varepsilon}$, and $1 \leq i \leq |\langle C^n_{w_E} \rangle|$, evaluate the advice circuit $E$ on all its inputs to reconstruct its truth table $w_E \in \{0,1\}^{n^c}$. Run $A(1^n, w_E)$ to compute the string $\langle C^n_{w_E} \rangle$, and output its $i$th bit."

  Since $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$, we get that
  $$n^\varepsilon\text{-}\mathsf{DCL}(C) \in \mathsf{SIZE}[O((n^\varepsilon(\log n))^k)] \subset \mathsf{SIZE}[\sqrt{n}],$$
  for our choice of $\varepsilon = 1/(3k)$. Using as advice these small circuits $D$ encoding $O(n^k)$-size circuits $C^n_{w_E}$, we get that
  $$H_b \in \mathsf{io}\text{-}\mathsf{DTIME}[n^b]/o(n),$$
  contradicting Lemma 2.5.

- Case 2: $A(1^n, \cdot)$ has only hard witnesses almost everywhere. In this case, for every large enough $n$, every string $w \in \{0,1\}^{n^c}$ such that $A(1^n, w) \neq \perp$ is a truth table of exponential circuit complexity $n^{\Omega(1)}$. Moreover, for every $n$, $A(1^n, w) \notin \perp$ for at least $1/2$ fraction of all $w$'s. Hence, for every large enough $n$, we can generate a hard truth table in $\mathsf{FZPP}$, which implies $\mathsf{BPP} = \mathsf{ZPP}$, and moreover, $\mathsf{BPP}/1 = \mathsf{ZPP}/1$, via the standard "hardness-randomness tradeoffs" [NW94; IW97].

  By Theorem 3.16, there is a pseudodeterminsitic PRG $G_n \colon \{0,1\}^{\lfloor n^{\varepsilon} \rfloor} \to \{0,1\}^{n^c}$, computable with 1 bit of advice, that fools $A(1^n, \cdot)$ on infinitely many input lengths $n$. That is, some output string $w$ of $G_n$ satisfies $A(1^n, w) \neq \perp$. We use this generator to define the following language

$$L_A^G = \{(n, a, i) \mid a \in \{0,1\}^{n^{\varepsilon}},\ i \in \{0,1\}^{O(k \log n)},\ A(1^n, G_n(a)) \neq \perp, \text{ and } \left\langle C^n_{G_n(a)} \right\rangle_i = 1\},$$

  which can be viewed as the concatenation of the languages $n^{\varepsilon}\text{-}\mathsf{DCL}(C^n_{G_n(a)})$ over all seeds $a \in \{0,1\}^{n^{\varepsilon}}$ of the PRG $G_n$.

  Given that $G$ is computable with 1 bit of advice in pseudodeterministic polynomial time, and $A(1^n, w)$ is computed by a deterministic polynomial time algorithm, we get that

$$L_A^G \in \mathsf{BPP}/1 = \mathsf{ZPP}/1.$$

  The assumption $\mathsf{ZPP}/1 \subseteq \mathsf{SIZE}[O(n^k)]$ implies that $L_A^G$ admits a sequence $\{D_m\}_{m \geq 1}$ of circuits of size $O((m)^k)$, where $m = n^{\varepsilon} + O(k \log n) \leq O(n^{\varepsilon})$ is the input length of $L_A^G$, referring to an $n$-input circuit $C^n_{G_n(a)}$ for some $a \in \{0,1\}^{n^{\varepsilon}}$.

  Since for infinitely many values of $n$ the generator $G_n$ "hits" a good choice of the random string $w$ where $A(1^n, w) \neq \perp$, there are infinitely many input lengths $n$ for which there is a string $a_n \in \{0,1\}^{n^{\varepsilon}}$ (a "good" seed to $G_n$) such that $D_m(n, a_n, \cdot)$ encodes a circuit $C_n$ of size $O(n^k)$ that computes $H_b \cap \{0,1\}^n$. By our choice of $\varepsilon = 1/3k$, the size of $D_m$ is at most $O(n^{1/3})$, which implies that $D_m(1^m, a_m, \cdot)$ is also a Boolean circuit of size at most $O(n^{1/3})$.

  Finally, the existence of infinitely many circuits of this form implies that

$$H_b \in \text{io-}\mathsf{DTIME}[n^b]/o(n),$$

  contradicting Lemma 2.5.

This proves that at least one of the lower bounds in items (1) and (2) of the theorem must hold. We can merge them into a single statement of the "In particular" part of the theorem because of the standard connection between advice and promise classes: $\text{promise-}\mathsf{ZPP} \subset \mathsf{SIZE}[O(n^k)]$ implies that $\mathsf{ZPP}/O(n) \subset \mathsf{SIZE}[O(n^k)]$ (see, e.g., [San09] for an argument). $\square$

**Remark 3.19.** *It is known that, for every $k \geq 1$, $\text{promise-}\mathsf{ZPP}^{\mathsf{MCSP}} \not\subset \mathsf{SIZE}[O(n^k)]$ [IKV18]. Theorem 3.18 removes the need for the $\mathsf{MCSP}$ oracle, albeit at the expense of requiring randomized uniformity.*

## 3.4 Randomized LEARN Uniformity for MA and ZPP with a Restricted Oracle

**Definition 3.20.** A language $L$ is in $\mathsf{BPP}$-uniform $\mathsf{SIZE}[s(n)]$ if there exists a family $C = \{C_n\}_{n \geq 1}$ of $s(n)$-size circuits $C_n$ for $L$ such that the $n\text{-}\mathsf{DCL}(C) \in \mathsf{BPP}$.

**Theorem 3.21.** *For any $k \geq 1$, $\mathsf{BPP} \not\subset \mathsf{BPP}\text{-uniform } \mathsf{SIZE}[O(n^k)]$.*

*Proof.* The argument is very similar to that in Theorem 3.3. Assume that both $\mathsf{BPP} \subset \mathsf{SIZE}[O(n^k)]$ and, for some $b \geq 4k$,

$$H_b \in \mathsf{BPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)].$$

The latter implies that there is a family $C$ of $O(n^k)$-size circuits $C_n$ for $H_b$ such that

$$n^{1/(2k)}\text{-}\mathsf{DCL}(C) \in \mathsf{BPP}.$$

Hence, each $C_n$ is computable by a circuit $D_m$ on at most $\sqrt{n}$ gates. Note that the evaluator TM $\mathcal{E}(x)$ is *deterministic*: on advice describing $D_m$, $\mathcal{E}$ constructs $C_n$ and then evaluates $C_n$ on a given $x \in \{0,1\}^n$. We get that $H_b \in \mathsf{TIME}[n^b]/o(n)$, contradicting Lemma 2.5. $\square$

Extending to the two-sided error randomized search uniformity, we get the following.

**Definition 3.22** ($\mathsf{FBPP}$ Uniformity)**.** We say that a language $L$ is in

$$\mathsf{FBPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[s(n)]$$

if there is a probabilistic polynomial time algorithm $A$ such that for every $n$,

$$\Pr_A[A(1^n) \text{ outputs some } s(n)\text{-size circuit } C_n \text{ for } L \cap \{0,1\}^n] \geq 2/3.$$

The success probability $2/3$ in the definition of $\mathsf{FBPP}$ uniformity can be amplified via standard techniques: independently generate circuits $C_1, \ldots, C_\ell$ and output $\mathsf{MAJ}(C_1, \ldots, C_\ell)$. Moreover, by another standard derandomization argument, a $\mathsf{FBPP}$-$\mathsf{uniform}$ sequence of *randomized* circuits can be converted into a $\mathsf{FBPP}$-$\mathsf{uniform}$ sequence of *deterministic* circuits. In contrast, we are not aware of an analogous result in the case of $\mathsf{BPP}$ uniformity, since fixing the randomness of a randomized circuit in different ways induce distinct direct connection languages, and in $\mathsf{BPP}$ uniformity a *fixed* sequence of deterministic circuits must be specified.

**Remark 3.23** (A connection between $\mathsf{FBPP}$ uniformity and $\mathsf{FP}\text{-}\mathsf{LEARN}^{\mathsf{EQ[poly]}}$ uniformity.)**.** *For certain problems, a lower bound against* $\mathsf{FBPP}$ *uniformity yields a lower bound for the same problem against* $\mathsf{FP}\text{-}\mathsf{LEARN}^{\mathsf{EQ[poly]}}$ *uniformity. Indeed, let* $L \in \mathsf{P}$ *be* self-correctable*, in the sense that a circuit* $C$ *of size* $s$ *for* $L_n$ *that is correct on a* $(1 - 1/n^c)$*-fraction of inputs can be converted in polynomial time with high probability into a circuit* $C'$ *of size* $s' \geq s$ *that computes* $L_n$*. Then, if* $L \in \mathsf{FP}\text{-}\mathsf{LEARN}^{\mathsf{EQ[poly]}}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[s]$ *we have* $L \in \mathsf{FBPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[s']$*. This can be done by exploiting the following idea from learning theory: in order to try to answer an equivalence query on a circuit of size* $\leq s$ *without access to an oracle, a randomized polynomial-time algorithm can sample a random input and check, using that* $L \in \mathsf{P}$*, if a mistake is found. If so, the equivalence query is answered correctly and the simulation proceeds. Otherwise, if enough random inputs do not exhibit a mistake, with high probability the queried circuit is sufficiently close to* $L$*. In this case, the self-corrector for* $L$ *can be invoked to produce a circuit of size* $s'$ *for* $L$*. We note that the assumption that* $L \in \mathsf{P}$ *is only used to answer a membership query to* $L$*. This assumption can be dropped when* $L$ *is downward-self-reducible, using ideas from* [IW01]*.*

It is not clear if an analogue of Theorem 3.18 for $\mathsf{FBPP}$ holds, i.e., whether $\mathsf{promise}\text{-}\mathsf{BPP} \not\subset \mathsf{FBPP}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[O(n^k)]$ for every $k \geq 1$. However, we can show an $\mathsf{FBPP}$-uniform circuit lower bound for the class $\mathsf{MA}$. The following two theorems should be contrasted with the known unconditional circuit lower bound for the promise-version of $\mathsf{MA}$: for every constant $k \geq 1$, $\mathsf{MA}/1 \not\subset \mathsf{SIZE}[O(n^k)]$ [San09].

**Theorem 3.24.** *If* $\mathsf{SAT} \in \mathsf{FBPP}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}]$*, then, for every* $k \geq 1$*,*

$$\mathsf{BPP} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)].$$

*In particular, for every* $k \geq 1$*,* $\mathsf{MA} \not\subset \mathsf{FBPP}\text{-uniform } \mathsf{SIZE}[O(n^k)]$.

*Proof.* The assumption implies that $\mathsf{SAT} \in \mathsf{BPP}$, and hence, we conclude that $\mathsf{PH} = \mathsf{BPP}$. The conclusion follows by Lemma 2.3. To argue the "In particular" part of the theorem, we just use that $\mathsf{SAT} \in \mathsf{NP} \subseteq \mathsf{MA}$ and that $\mathsf{BPP} \subseteq \mathsf{MA}$. □

Next we consider $\mathsf{FZPP}^{\mathsf{rf}}\text{-}\mathsf{LEARN}^{\mathsf{EQ}}$-uniformity.

**Theorem 3.25.** *If, for some* $\mathsf{poly}(n)$*-time function* $1 \leq r(n) \leq \mathsf{poly}(n)$*,*

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP}^{\mathsf{rf}}\text{-}\mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[r(n)]}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}],$$

*then* $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}[O(a(\mathsf{poly}(n)))]} \subset \mathsf{MA}//O(a(\mathsf{poly}(n)))$*, and, for every* $k \geq 1$*,*

1. $\mathsf{MA}//O(a(\mathsf{poly}(n))) \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$*, and*
2. $\mathsf{ZPP}^{\mathsf{NP}[O(a(\mathsf{poly}(n)))]} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$*,*

*where* $a(n) = \lceil \log(r(n) + 1) \rceil$*.*

*Proof.* We follow the proof of Theorem 3.12, but adapting it to the case of randomized learning algorithms. We first prove Items 1 and 2.

Proof of Item 1. Imagine picking a random string $\alpha \in \{0,1\}^{\mathsf{poly}(n)}$ used by the learning algorithm. Once this random string is fixed, with probability at least $3/4$, the algorithm becomes a deterministic learning algorithm, correct on all possible sequences of $\mathsf{EQ}$ answers (by the definition of the *randomness-first* randomized LEARN uniformity). Call such a random string $\alpha$ *good.*

For a good random string $\alpha$, let $\ell = \ell(n, \alpha) \leq r(n)$ be the maximum number of EQs that one can force the resulting deterministic learning algorithm to make before it is guaranteed to output a correct $\mathsf{SAT}$ circuit $C^{\ell+1}$. Note that this value $\ell$ depends both on the input length $n$ *and the randomness* $\alpha$ *used by the learning algorithm.*

To decide if a given formula $\psi$ of size $n$ is unsatisfiable, we can pick a random string $\alpha$ of polynomial length, get $\ell = \ell(n, \alpha)$ as advice dependent on the randomness $\alpha$, and run the $\mathsf{NP}$ algorithm from Step 2 of the proof of Theorem 3.12 above. With probability at least $3/4$, we get a good random string $\alpha$ such that, for the advice $\ell = \ell(n, \alpha) \leq r(n)$, the resulting nondeterministic algorithm correctly decides $\mathsf{UNSAT}$ for all formulas $\psi$ of size $n$. That is, we get that $\mathsf{UNSAT} \in \mathsf{AM}//a(n)$. Hence, $\mathsf{coNP} \subseteq \mathsf{AM}//a(\mathsf{poly}(n))$, and so $\mathsf{PH} \subset \mathsf{AM}//a(\mathsf{poly}(n))$, implying the existence of required hard languages in $\mathsf{AM}//a(\mathsf{poly}(n))$ by Lemma 2.3.

Finally, since $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$ implies that $\mathsf{AM} = \mathsf{MA}$ [Arv+95], and this equality extends to the case of randomness-dependent advice versions of these two classes[20], we get that $\mathsf{PH} \subset \mathsf{MA}//a(\mathsf{poly}(n))$, implying Item 1.

Proof of Item 2. As above, we first pick randomness $\alpha$ for our learning algorithm for $\mathsf{SAT}$ search circuits, but instead of asking for the advice $\ell = \ell(n, \alpha)$, we compute this value $\ell(n, \alpha)$ ourselves, using a $\mathsf{P}^{\mathsf{NP}[a(n)]}$ algorithm.

[20]To go from $\mathsf{AM}$ to $\mathsf{MA}$, the idea of [Arv+95] is to guess a $\mathsf{Search\text{-}SAT}$ polysize circuit $C$ first and then, for given randomness $r$ chosen by Arthur, use $C$ to witness Merlin's existential quantifier in the original $\mathsf{AM}$ protocol. If an $\mathsf{AM}$ protocol on $n$-bit inputs uses advice string $\beta_n(r)$ dependent on Arthur's randomness $r$, we do the same construction as above, also using the advice $\beta_n(r)$ for Arthur's randomness $r$ in the new $\mathsf{MA}$ protocol.

We look for the largest such $i$ so that there is a run of the learning algorithm (using the randomness $\alpha$) with $i$ circuits produced by the learning algorithm all having some satisfiable formulas as counterexamples. For a particular $i$, the existence of such a run is an $\mathsf{NP}$ question. We can look for the maximum $i \le r(n)+1$ by binary search, in $\mathsf{P}^{\mathsf{NP}}$, where we only need to ask $\lceil \log(r(n)+1) \rceil$ $\mathsf{NP}$ queries. If the maximum found $i$ is equal to $r(n)+1$, we know that the randomness $\alpha$ is "bad": the learning algorithm on this $\alpha$ is not guaranteed to find a correct $\mathsf{SAT}$ search circuit after $r(n)$ EQs. In this case, we output $\perp$. Otherwise, we've certified that randomness $\alpha$ is good: an algorithm is guaranteed to produce a correct $\mathsf{SAT}$ circuit no later than after $i$ EQs.

Thus, we can produce a correct $\mathsf{SAT}$ search circuit in $\mathsf{FZPP}^{\mathsf{NP}[\mathsf{wit},a(n)+1]}$. This implies that $\mathsf{PH} = \Sigma_2^p = \mathsf{ZPP}^{\mathsf{NP}[a(\mathsf{poly}(n))+2]}$. Indeed, once we generate a good randomness $\alpha$ (which can be checked using just an $\mathsf{NP}$ oracle, i.e., we don't need access to $\mathsf{NP}[\mathsf{wit}]$), we can view our randomized algorithm as a deterministic computation (using this fixed randomness). So we get that a correct $\mathsf{Search\text{-}SAT}$ circuit can be constructed uniformly by an $\mathsf{FP}^{\mathsf{NP}[\mathsf{wit},a(n)+1]}$ algorithm. This allows one then to collapse $\Sigma_2^p$ to $\mathsf{P}^{\mathsf{NP}[a(\mathsf{poly}(n))+2]}$ using an argument of [CK07]; see the proof of Lemma B.2 (Item 1) in Appendix B for details. The required hard languages are then obtained via Lemma 2.3.

Note that the arguments above imply that $\mathsf{PH} \subset \mathsf{MA}//a(\mathsf{poly}(n))$ and $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}[a(\mathsf{poly}(n))+2]}$, as promised. ☐

Next we show that sometimes it is possible to convert a randomized learning algorithm with EQs to the "randomness first" kind, so that Theorem 3.25 can be applied. More precisely, we establish this for $\mathsf{Search\text{-}SAT}$ and for learners that make constantly many $\mathsf{Search\text{-}SAT\text{-}EQ}$s. To achieve that, we proceed in two steps.

First, we prove that the success probability of the learner can be amplified via a certain parallel repetition procedure. This increases the running time but maintains the original number of queries.

**Lemma 3.26** (Boosting the success probability of a $\mathsf{FZPP\text{-}LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ[const]}}$ algorithm)**.** *For any constant $\ell \ge 1$, if*

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP\text{-}LEARN}_{\gamma}^{\mathsf{Search\text{-}SAT\text{-}EQ}[\ell]}\text{-uniform } \mathsf{SIZE}[a \cdot n^b],$$

*where $\gamma(n) \ge 1/\mathsf{poly}(n)$ and $a, b$ are constants, then for every constant $v \ge 1$,*

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP\text{-}LEARN}_{\beta}^{\mathsf{Search\text{-}SAT\text{-}EQ}[\ell]}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}],$$

*where $\beta(n) \ge 1 - 2^{-n^v}$.*

*Proof.* Let $A$ be a LEARN-uniform construction of $\mathsf{Search\text{-}SAT}$ circuits of size at most $a \cdot n^b$ which makes at most some constant $\ell$ EQs, and succeeds with probability $\gamma(n) \ge 1/\mathsf{poly}(n)$. For simplicity of notation, we assume that $v = 1$. (The parameters can be easily adapted to handle any constant $v$.) We view the run of $A(1^n)$ as consisting of at most $\ell$ rounds. In each round $i$, the algorithm picks some randomness $r$, constructs a candidate $\mathsf{Search\text{-}SAT}$ circuit $C$ (based on $r$, the randomness $r_1, \ldots, r_{i-1}$ from the previous rounds, and the previous counterexamples $z_1, \ldots, z_{i-1}$ obtained from the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle), and asks the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle about $C$.

We call round $i$ (and the randomness $r$ picked in the round) *successful*

- either if $C$ is a correct $\mathsf{Search\text{-}SAT}$ circuit,
- or if, for *every* counterexample $z$ from the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle on $C$, the algorithm $A$ succeeds within at most $\ell - i$ $\mathsf{EQ}$s (rounds) with probability at least $\gamma/2^i$, when using the partial computation transcript $r_1, z_1, \ldots, r_{i-1}, z_{i-1}, r, z$.

**Claim 3.27.** *If $i = 1$ or round $i - 1$ is successful, then round $i$ is successful with probability at least $\gamma/2^i$, for $1 \le i \le \ell$.*

*Proof of Claim 3.27.* The proof is by induction on $i$. It might be helpful to recall the definition of success probability of a $\mathsf{FZPP\text{-}LEARN}^{\mathsf{EQ}}$ algorithm, which is defined in Section 2.4. For $i = 1$, the success probability at least $\gamma$ of the learning algorithm can be written as an expected success probability over the random string $r$ chosen in round 1, conditioned on the random choice and the worst possible counterexample provided by the EQ oracle. This expected success probability being at least $\gamma$ by assumption implies that the conditional expected success probability must be at least $\gamma/2$ for at least $\gamma/2$ fraction of random strings $r$.

For $1 < i \le \ell$, the success probability of the learning algorithm starting at round $i$ is at least $\gamma' = \gamma/2^{i-1}$ by the inductive hypothesis. As in the base case, we write this success probability as an expectation of conditional success probabilities based on the random string chosen in round $i$ and the worst possible counterexample for the corresponding EQ, to conclude that, for at least $\gamma'/2 = \gamma/2^i$ fraction of random strings, the success probability assigned to the top node of the corresponding sub-tree is at least $\gamma'/2 = \gamma/2^i$. □

By Claim 3.27, $A$ succeeds in each round with probability at least $\gamma/2^\ell$ by getting either a correct circuit $C$ or a new learning algorithm (with one less EQ) that succeeds with probability at least $\gamma/2^\ell$. Next we show how to boost this success probability to be exponentially close to 1, without increasing the number $\ell$ of EQs.

Error probability reduction via parallel repetition. For $m = 2^{\ell+1}(\ell+1)n/\gamma$, pick $\ell \cdot m$ random strings $R = \{r_i^j\}_{1\le i\le \ell, 1\le j\le m}$ uniformly at random. We will think of the blocks $R_i = \{r_i^1, \dots, r_i^m\}$ of these strings. Once we randomly pick and fix the set $R$ of random strings, the following learning algorithm $B_R$ is a deterministic algorithm with the Search-SAT EQ oracle access:

"On input $1^n$,

for each round $i = 1 \dots \ell$

- using the previous EQ counterexamples $z_1, \dots, z_{i-1}$, let $C_i^1, \dots, C_i^{m^i}$ be the candidate Search-SAT circuits[21] produced by

$$A\left(1^n, r_1^{j_1}, z_1, \dots, r_{i-1}^{j_{i-1}}, z_{i-1}, r_i^{j_i}\right)$$

  on all $i$-tuples of random strings from $R_1 \times \dots \times R_i$, where each $1 \le j_i \le m$,
- define the candidate Search-SAT circuit

$$C_i^* = \bigvee_{j=1}^{m^i} C_i^j$$

  (i.e., $C_i^*$ outputs a satisfying assignment on a given input formula iff at least one of the circuits $C_i^j$ does),
- ask the EQ for $C_i^*$; if the answer is "yes", then output $C_i^*$ and halt; otherwise, get a counterexample $z_i$ for $C_i^*$ (a satisfiable formula $\psi$, with a satisfying assignment, such that $C_i^*(\psi)$ fails to find a satisfying assignment for $\psi$)[22] and go to the next round."

[21]Recall that each such candidate Search-SAT circuit never makes a mistake on any unsatisfiable formula.

[22]Note that this $z_i$ will be a counterexample for *all* circuits $C_i^j$ simultaneously.

The runtime of the described algorithm $B_R(1^n)$ is at most $O(m^\ell)$ times the runtime of $A(1^n)$, which is still $\mathsf{poly}(n)$. Each candidate $\mathsf{Search\text{-}SAT}$ circuit $C_i^*$ is also of size $\mathsf{poly}(n)$. Next we argue the correctness of $B_R$.

**Claim 3.28.** *The algorithm $B_R$ on $1^n$ succeeds with probability at least $1 - 2^{-n}$.*

*Proof of Claim 3.28.* Each counterexample $z$ from the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle consists of a formula $\psi$ of size $n$ and a satisfying assignment to $\psi$ of at most $n$ bits; we can upperbound the size of $z$ by $2n$ bits. Note that the total number of possible (legal) sequences of $\ell$ counterexamples is thus at most $(2^{2n})^\ell = 2^{2\ell n}$.

Consider a particular sequence $z_1, \dots, z_\ell$ produced by the $\mathsf{EQ}$ oracle during a run of $B_R(1^n)$. In round 1, the probability that *none* of the randomly chosen strings $r \in R_1$ is good for $A(1^n)$ is at most

$$(1 - \gamma/2)^m \leq e^{-m\gamma/2} \leq 2^{-2(\ell+1)n}.$$

Thus, with high probability, $R_1$ contains a good string $r_1^* \in R_1$ (fix one arbitrarily).

Let $C_1$ be the circuit produced by $A(1^n)$, using randomness $r_1^*$. If $C_1$ is a correct circuit, then so is $C_1^*$ produced by $B_R(1^n)$, using $R_1$. Otherwise, the counterexample $z_1$ from the $\mathsf{EQ}$ oracle for $C_1^*$ is a valid counterexample for $C_1$ as well. Hence, $A(1^n)$, starting with the partial transcript $r_1^*, z_1$, will succeed within $\ell - 1$ rounds, with probability at least $\gamma/2$.

For round 2, the probability that *none* of the randomly chosen strings $r \in R_2$ is good for $A(1^n, r_1^*, z_1)$ is again at most $2^{-2(\ell+1)n}$. So, with high probability, $R_2$ contains a good string $r_2^* \in R_2$ (fix one arbitrarily). Let $C_2$ be the circuit produced by $A(1^n)$, using $r_1^*, z_1, r_2^*$. Again, if $C_2$ is correct, then so is $C_2^*$ produced by $B_R(1^n, z_1)$. Otherwise, $z_2$ is a valid counterexample for $C_2$, and so $A(1^n, r_1^*, z_1, r_2^*, z_2)$ must succeed within $\ell - 2$ rounds, with probability at least $\gamma/4$. And so on.

It follows that, for some $1 \leq t \leq \ell$, the algorithm $A(1^n, r_1^*, z_1, \dots, r_t^*)$ outputs a correct $\mathsf{Search\text{-}SAT}$ circuit $C$ in round $t$, and hence $B_R(1^n, z_1, \dots, z_{t-1})$ also outputs a correct circuit in the same round $t$.

The probability over $R$ that $B_R(1^n)$ fails to output a correct circuit within $\ell$ rounds is at most the sum of the probabilities that a set $R_i$ fails to contain a good random string $r_i^*$ for $A(1^n)$, over all $1 \leq i \leq \ell$. The latter is at most $\ell \cdot 2^{-2(\ell+1)n}$. By the union bound, the probability that $R$ is bad for at least one sequence of counterexamples out of at most $2^{2\ell n}$ possible sequences is at most

$$2^{2\ell n} \cdot \ell \cdot 2^{-2(\ell+1)n} \leq \ell \cdot 2^{-2n} \leq 2^{-n},$$

for large enough $n$. The claim follows. □

By Claim 3.28, the result also follows. We note that the resulting LEARN-uniform construction provides circuits of polynomially larger size and requires an EQ oracle that is able to answer queries that describe circuits of polynomially larger size. Its correctness follows from the correctness of the original algorithm $A$ when interacting with a more limited EQ oracle. □

Next, we argue that a learner with overwhelming success probability can be converted into a randomness-first learner if its query complexity is small.

**Lemma 3.29** (Converting a randomized learner to a randomness-first learner)**.** *For every constant $\ell \geq 1$ there is a constant $\ell' \geq 1$ such that if $\beta(n) \geq 1 - 2^{-\ell' n}$ and*

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP\text{-}LEARN}_\beta^{\mathsf{Search\text{-}SAT\text{-}EQ}[\ell]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[\mathsf{poly}],$$

*then*

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP}^{\mathsf{rf}}\text{-}\mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[\ell]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[\mathsf{poly}].$$

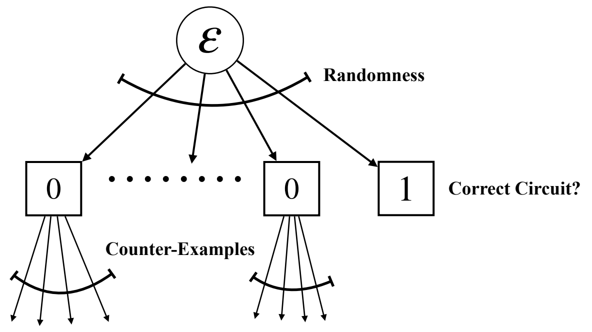

Figure 1: Randomized learner and its game tree when $\ell = 1$.

*Proof.* Let $A$ be a $\mathsf{FZPP\text{-}LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[\ell]}$ algorithm that constructs $\mathsf{poly}(n)$ size circuits that solve $\mathsf{Search\text{-}SAT}$, and assume that it succeeds with success probability $\beta(n) \geq 1 - 2^{-\ell' n}$, where $\ell'$ is a sufficiently large constant independent of $n$. For convenience, let $\beta(n) = 1 - \delta$, where $\delta \leq 2^{-\ell' n}$.

Recall that the computation of $A(1^n)$ can be viewed as a rooted tree $\mathcal{T}$ whose paths can be indexed by strings $(w_1, z_1, \ldots, w_\ell, z_\ell)$, where $w_i \in \{0,1\}^{\mathsf{poly}(n)}$ represents a choice of the $i$-th random string, and $z_i$ encodes the oracle answer to the circuit queried by $A(1^n, w_1, z_1, \ldots, w_{i-1})$.

The proof is by induction on $\ell$. When $\ell = 1$, $A(1^n)$ makes a single query based on its choice of $w_1$. Let $C_{w_1}$ be the corresponding circuit. If $C_{w_1}$ is correct, $z_1$ represents "correct". Otherwise, $z_1$ encodes "wrong" and a counter-example to the correctness of $C_{w_1}$. Clearly, in this case $A$ is itself a randomness-first learner with error probability at most $\delta \leq 1/4$. Consequently, the result holds for $\ell = 1$. (See Figure 1 for a diagram that represents $\mathcal{T}$ in the trivial case $\ell = 1$.)

Next, we analyze the case $\ell = 2$. We would like to prove that, with probability at least $3/4$ over the choice of strings $w_1$ and $w_2$, no matter the counter-example $z$ provided by the oracle on query $A(1^n, w_1)$ we get that $A(1^n, w_1, z, w_2)$ outputs a correct circuit. Let $\gamma$ be the function that describes the success probability of $A(1^n)$. By assumption, $\gamma(\varepsilon) = 1 - \delta$, where $\varepsilon$ denotes the root of tree $\mathcal{T}$. To estimate the desired probability, we first observe that, with probability at least $1 - \sqrt{\delta}$ over the choice of $w_1$, we have $\gamma(w_1) \geq 1 - \sqrt{\delta}$. This follows from a standard Markov argument and $\gamma(\varepsilon) = \mathbb{E}_{\boldsymbol{w_1}}[\gamma(\boldsymbol{w_1}))]$. If this is the case, we say that the sub-tree $\mathcal{T}_{w_1}$ is good.

Fix any good sub-tree $\mathcal{T}_{w_1}$. Since $\gamma(w_1) = \min_z \gamma(w_1, z)$, for any oracle answer $z_1$ to $A(1^n, w_1)$, we get $\gamma(w_1, z_1) \geq \gamma(w_1) \geq 1 - \sqrt{\delta}$. Note that there are at most $2^n$ oracle answers distinct from "correct" (i.e., counter-examples). For a random choice of $w_2 \in \{0,1\}^{\mathsf{poly}(n)}$ (fixed together with $w_1$ and before we know $z_1$, but assuming a good sub-tree $\mathcal{T}_{w_1}$), by a union bound over the oracle answers to the first query, we reach a sub-tree $\mathcal{T}_{w_1, z_1, w_2}$ corresponding to a correct output circuit $A(1^n, w_1, z_1, w_2)$ for $\mathsf{Search\text{-}SAT}$ except with probability $2^n \cdot \sqrt{\delta}$. In other words, even if the oracle knows $w_2$ when it is about to answer the query $A(1^n, w_1)$, with high probability over our choice of $w_2$, no matter its answer $z_1$, $A(1^n, w_1, z_1, w_2)$ outputs a correct circuit. (See Figure 2 for an informal explanation for this upper bound.)

Overall, over the initial choices of $w_1$ and $w_2$, we obtain a correct deterministic learner for $\mathsf{Search\text{-}SAT}$ except with probability $\sqrt{\delta} + 2^n \cdot \sqrt{\delta}$, where the first term accounts for a bad choice of $w_1$, and the second term for a bad choice of $w_2$ assuming a good choice of $w_1$. Consequently, $A$ is a randomness-first learner with success probability at least $1 - \sqrt{\delta} - 2^n \cdot \sqrt{\delta} \geq 3/4$, if $\ell' > 2$.

In the general case, we proceed by induction. In order to explain how this is done in a more concrete way, we show how to reduce the $\ell = 3$ case to $\ell = 2$. First, we summarize what we have shown in the $\ell = 2$ case as follows.

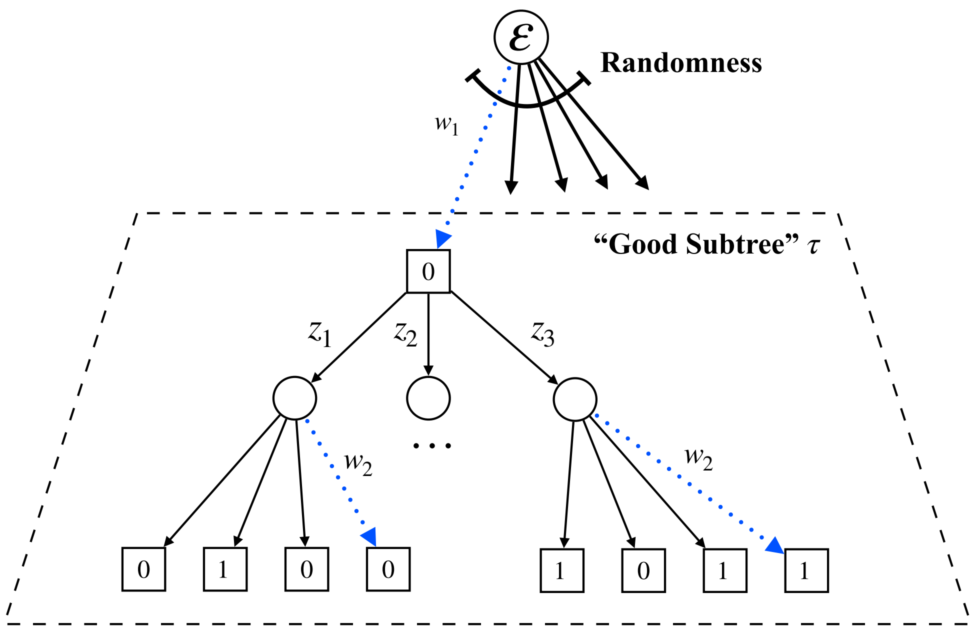

Figure 2: Randomized learner and its game tree when $\ell = 2$. In this example, if the oracle knows $w_1$ and $w_2$ before answering the equivalence query, then by replying with counter-example $z_1$ the learner outputs an incorrect circuit corresponding to path $w_1, z_1, w_2$. Note that each answer $z_i$ *eliminates* a fraction of $w_2$'s.

**Claim 3.30.** *If $A'$ is a randomized learner that makes $\ell = 2$ queries and succeeds with probability $1 - \delta'$, then except with probability at most $\delta'^{1/2} + 2^n \cdot \delta'^{1/2}$, fixing its randomness provides a deterministic learner that makes $\ell = 2$ queries and succeeds regardless of the oracle answers.*

Let $A$ be a randomized learner that makes $\ell = 3$ queries, and assume that $A$ succeeds with probability at least $1 - \delta$. In other words, $\gamma(\varepsilon) \geq 1 - \delta$, where $\varepsilon$ denotes the root of $\mathcal{T}$, and $\mathcal{T}$ encodes the interaction of $A(1^n)$ with the Search-SAT-EQ oracle. Note that the first random string $w_1$ selected by $A(1^n)$ satisfies the following property: with probability at least $1 - \delta^{1/2}$, no matter the counter-example $z_1$ provided by the oracle to the first query, the learner descends to a node $(w_1, z_1)$ of $\mathcal{T}$ such that $\gamma(w_1, z_1) \geq 1 - \delta^{1/2}$. (This is similar to the $\ell = 2$ case, and relies only on a standard Markov argument and the definition of the success probability function $\gamma$.) Now what we know is that, for a good choice of $w_1$ that leads to such a sub-tree, if the oracle decides to answer with a fixed counter-example $z_1 \in \{0,1\}^n$, then we are in a setting that is similar to Claim 3.30 for $\delta' = \delta^{1/2}$. In other words, for the learning game that starts at $\mathcal{T}_{(w_1,z_1)}$, we know that except with probability $\delta'^{1/2} + 2^n \cdot \delta'^{1/2}$, fixing the random strings $w_2$ and $w_3$ before interacting with the oracle produces a correct deterministic learner. Overall, we get to select $w_1$ before the oracle, so except with probability $\delta^{1/2}$, we are at a good sub-tree $\mathcal{T}_{w_1}$. However, we cannot control the answer $z_1$ of the oracle, who knows $w_2$ and $w_3$. Thus, to complete the analysis of the $\ell = 3$ case, we union bound over all possible choices of $z_1$, i.e., we eliminate bad choices of $w_2$ and $w_3$ for each possible $z_1$, assuming a good $w_1$. The probability of not obtaining a correct deterministic learner after randomly fixing $w_1$, $w_2$, and $w_3$ is then at most

$$\delta^{1/2} + 2^n \cdot (\delta'^{1/2} + 2^n \cdot \delta'^{1/2}) = \delta^{1/2} + \delta^{1/4}(2^n + 2^{2n}).$$

This failure probability is at most 1/4 if we start with a randomized learner that succeeds with probability $\beta(n) \geq 1 - \delta(n)$, where $\delta(n) = 2^{-\ell' n}$ for a large enough constant $\ell'$.

The general case follows in the natural way using an induction hypothesis that is similar to Claim 3.30. We omit the details since a precise bound is not needed in other parts of the paper. $\square$

The following result shows that, for the purpose of proving circuit lower bounds for some explicit problem, non-uniformity is essentially equivalent to zero-error uniformity with access to an $\mathsf{NP}$ oracle.

**Proposition 3.31** (Power of zero-error uniformity with an $\mathsf{NP}$ oracle)**.**

1. *For every $k \geq 1$, $\mathsf{NP} \subset \mathsf{SIZE}[O(n^k)]$ if and only if $\mathsf{NP} \subset \mathsf{ZPP}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}[O(n^k)]$.*
2. *For every $k \geq 1$, if $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$, then $\mathsf{P} \subset \mathsf{FZPP}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}[O(n^{k+1})]$. In particular, $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$ for some $k \geq 1$ iff $\mathsf{P} \subset \mathsf{FZPP}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}[O(n^{k'})]$ for some $k' \geq 1$.*

*Proof.* For Item 1, it suffices to show the implication from a nonuniform circuit upper bound to the $\mathsf{ZPP}^{\mathsf{NP}}$-uniform circuit upper bound. For any $k \geq 1$, if $\mathsf{NP} \subset \mathsf{SIZE}[O(n^k)]$, then $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$ by [Bsh+96]. For every $L \in \mathsf{NP}$, one can specify the family $C$ of lexicographically smallest circuits $C_n$ of size $O(n^k)$ deciding $L$, so that $n$-$\mathsf{DCL}(C) \in \mathsf{PH}$. It follows that $n$-$\mathsf{DCL}(C) \in \mathsf{ZPP}^{\mathsf{NP}}$.

For Item 2, the learning algorithm of [Bsh+96] is a $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm that can exactly learn any size $s(n) \geq n$ circuit on $n$ inputs, given oracle access to equivalence queries for $n$-input circuits of size $O(n \cdot s(n))$; so the algorithm outputs an equivalent circuit of size $O(n \cdot s(n))$. Note that for any language $L \in \mathsf{P}$, deciding for a given $n$-input circuit $C$ of size $s'(n)$ if there is an input $x$ where $C(x) \neq L(x)$ (and if so, finding such an input $x$) can be done in $\mathsf{P}^{\mathsf{NP}}$ in a fairly straightforward way. That is, one can simulate the $\mathsf{EQ}$ oracle for a language $L \in \mathsf{P}$ with an $\mathsf{NP}$ oracle. It follows that a circuit for $L \in \mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$ of size $O(n^{k+1})$ can be found by an $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm. $\square$

## 4 Unprovability of Circuit Upper Bounds in Bounded Arithmetic

Here we prove that various systems of Bounded Arithmetic cannot prove certain circuit upper bounds. For any complexity class $\mathsf{C}$, and a constant $k \geq 1$, the *circuit upper bound* statement

$$\mathsf{C} \subset \mathsf{SIZE}[O(n^k)]$$

means the following: for every Boolean function family $f = \{f_n\}_{n \geq 1} \in \mathsf{C}$, there is a constant $c \geq 0$, such that, for all input length $n \geq 1$, $f_n \in \mathsf{SIZE}[cn^k]$. We say that *a theory $\mathcal{T}$ proves that* $\mathsf{C} \subset \mathsf{SIZE}[O(n^k)]$ if, for every $f \in \mathsf{C}$ and for some constant $c \geq 0$,

$$\mathcal{T} \vdash \quad \forall n \;\; \exists C \; (\mathsf{size}(C) \leq cn^k) \;\; \forall X \; (|X| = n) \quad [C(X) = f(X)],$$

where the function $\mathsf{size}(C)$ (counting the number of gates in a circuit described by a binary string $C$), the circuit evaluation $C(X)$, and the function evaluation $f(X)$ must all be expressible in the language of $\mathcal{T}$.

**Remark 4.1.** *We recall the following facts about some of the theories investigated in this section.*

- *For $\mathsf{VPV}^1$, which contains all polytime algorithms as symbols $f$ of the language of $\mathsf{VPV}^1$, $\mathsf{size}(C)$ and circuit evaluation $C(X)$ are all expressible in a straightforward way.*
- *Similarly for $\mathsf{VPV}^2$, which contains function symbols $f$ for all $\mathsf{P}^{\mathsf{NP}}$ algorithms, the circuit evaluation problem for circuits $C$ with $\mathsf{NP}$ oracle gates is expressible as a $\mathsf{P}^{\mathsf{NP}}$ function.*

- *For* $\mathsf{V}^1$, *the polytime computable circuit evaluation problem can be expressed by a* $\Sigma_1^B$ *formula in the language of* $\mathsf{V}^1$ *[Bus86].*

*Crucially, one can verify in each case that the corresponding circuit upper bound sentences are compatible with the witnessing theorems that are needed in our proofs.*

Note that, in order to formally state a circuit upper bound sentence for a function or a language, we fix an *algorithm* that computes $f(X)$ to refer to $f$ in the sentence. Consequently, different algorithms that specify the same function or language $f \in \mathsf{C}$ result in different upper bound sentences. This can be an issue, since the theory $\mathcal{T}$ might not be able to establish that distinct algorithms compute the same function. However, as explained in Section 1.2.2, the unprovability results established in this work are all *robust* to this. In other words, when we show that a theory $\mathcal{T}$ cannot prove circuit upper bounds for a language $L \in \mathsf{C}$, this is independent of the algorithm used to represent $L$ in the upper bound sentence.

We refer the interested reader to [Pic15a; KO17; BKO20; MP20] for additional discussion on the formalization of circuit complexity in bounded arithmetic.

**Remark 4.2** (On the relation between the KPT Theorem and LEARN-uniformity)**.** *As explained in Section 1.3, in the context of circuit upper bounds, we can interpret the disjunction provided by the KPT Theorem (over the standard model) as a LEARN-uniform construction. Here we clarify a subtle point that is not essential but that might occur to some readers: there is a trivial reason for a formula* $\varphi(X, T_i(X, Z_1, \ldots, Z_{i-1}), Z_i)$ *in the KPT disjunction obtained from a circuit upper bound sentence to be false on a given (partial) input* $X, Z_1, \ldots, Z_{i-1}$*, regardless of the value of* $Z_i$*: the circuit proposed by the term* $T_i$ *(and that serves as a query to the EQ oracle) is simply not of the correct form (e.g., it is larger than the size bound or not defined over the right number of input variables). Since these checks can be done in polynomial time and any value of the variable* $Z_i$ *serves as a counter-example when this happens, one can simply proceed with the simulation and pretend that such trivial-to-answer queries are never passed on to the EQ oracle.*

## 4.1 $\mathsf{VPV}^1$ and $\mathsf{VPV}^2$

**Theorem 4.3** ([KO17])**.** *For any constant* $k \geq 1$, $\mathsf{VPV}^1$ *does* not *prove that* $\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$.

*Proof.* Suppose it does. Then by the KPT Witnessing for $\mathsf{VPV}$, we get that

$$\mathsf{P} \subseteq \mathsf{LEARN}^{\mathsf{EQ[const]}}\text{-uniform } \mathsf{SIZE}[O(n^k)],$$

contradicting Theorem 3.1. □

**Theorem 4.4.** *For any constant* $k \geq 1$, $\mathsf{VPV}^2$ *does* not *prove that* $\mathsf{P}^{\mathsf{NP}} \subset \mathsf{SIZE}^{\mathsf{NP}}[O(n^k)]$.

*Proof.* Suppose it does. Then by the KPT Witnessing for the universal theory $\mathsf{VPV}^2$, we get that

$$\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{FP}^{\mathsf{NP}}\text{-}\mathsf{LEARN}^{\mathsf{EQ[const]}}\text{-uniform } \mathsf{SIZE}^{\mathsf{NP}}[O(n^k)],$$

contradicting Theorem 3.2 (the relativized version of Theorem 3.1 where the learning algorithm is an $\mathsf{FP}^{\mathsf{NP}}$ algorithm). □

Since $\mathsf{VPV}^2$ is a conservative extension of $\mathsf{TV}^1$ (which is the second-order version of the first-order theory $\mathsf{T}_2^1$) [CN10, Theorem VIII.7.11], we also get that $\mathsf{TV}^1$ does not prove $\mathsf{P}^{\mathsf{NP}} \subset \mathsf{SIZE}^{\mathsf{NP}}[O(n^k)]$, strengthening a result of [BKO20].

## 4.2 $\mathsf{V}^1$

The statement $\mathsf{V}^1 \vdash \mathsf{NP} \subset \mathsf{SIZE}[O(n^k)]$ means that, for every $\Sigma_1^B$ formula $\varphi(X,Y)$ (defining an $\mathsf{NP}$ language $\{X \mid \exists Y\ \varphi(X,Y)\}$) and for some constant $c \geq 0$, $\mathsf{V}^1$ proves the following:

$$\forall n\ \exists C\ (\mathsf{size}(C) \leq cn^k)\ \forall X\ (|X| = n) \quad [C(X) = 1 \iff \exists Y\ \varphi(X,Y)]. \tag{1}$$

The "infinitely often" statement $\mathsf{V}^1 \vdash \mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}]$ means that

$$\forall n\ \exists m \geq n\ \exists C\ (\mathsf{size}(C) \leq cm^d)\ \forall \varphi\ (|\varphi| = m) \quad [C(\varphi) = 1 \iff \exists Y\ \varphi(Y)].$$

We strengthen a result from [BKO20] as follows.

**Theorem 4.5.** *For any constant $k \geq 1$, $\mathsf{V}^1$ does* not *prove that $\mathsf{NP} \subset \mathsf{io\text{-}SIZE}[O(n^k)]$. In fact, at least one of the following must be true:*

1. *$\mathsf{V}^1$ does* not *prove that $\mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}(n)]$.*
2. *For any constant $k \geq 1$, $\mathsf{NP} \not\subset \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[O(n^k)]$.*

*In particular, for any constant $k \geq 1$, $\mathsf{V}^1$ does* not *prove that*

$$\mathsf{NP} \subset \mathsf{io\text{-}SIZE}[\mathsf{poly}(n)] \cap \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[O(n^k)].$$

*Proof.* If $\mathsf{V}^1$ doesn't prove that $\mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}]$, we are done. Otherwise, we argue as follows.
Reducing the "infinitely often" to the "almost everywhere" case. Suppose that $\mathsf{V}^1$ proves that $\mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}]$. As in [BKO20], we use Parikh's Theorem (Theorem 2.18) applied to $\mathsf{V}^1$ to argue that the existentially quantified number parameter $m$ is bounded by some term $t(n) \leq \mathsf{poly}(n)$. Since $\mathsf{SAT}$ is paddable language (a formula $\varphi$ of size $n$ can be always padded up to a satisfiability-equivalent formula $\varphi'$ of size $m$ for any $m \geq n$) and $m \leq \mathsf{poly}(n)$, we conclude that $\mathsf{V}^1$ proves that $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$, and proceed to the "almost everywhere" case next.
The "almost everywhere" case. If $\mathsf{V}^1$ proves that $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$, then it also proves that $\mathsf{Search\text{-}SAT} \in \mathsf{SIZE}[\mathsf{poly}]$ (as noted by [CK07]). The latter can be formalized as saying that

$$\forall n\ \exists C\ (\mathsf{size}(C) \leq cn^d)\ \forall (\varphi, w) \leq n \quad [\varphi(w)\ \Rightarrow\ \varphi(C(\varphi))],$$

where $C$ is an encoding of a $\mathsf{Search\text{-}SAT}$ circuit of $\mathsf{poly}(n)$ size, $\varphi$ is a formula, and $w$ is an assignment for $\varphi$. So the above formula says that $C$ finds a satisfying assignment for every input satisfiable formula $\varphi$. By the KPT-witnessing for $\mathsf{V}^1$ (Theorem 2.17), we get that $\mathsf{Search\text{-}SAT} \in \mathsf{LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[\mathsf{poly}]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[\mathsf{poly}]$; recall that here the $\mathsf{EQ}$ oracle on an incorrect $\mathsf{Search\text{-}SAT}$ candidate circuit $C$ must answer with a pair $(\varphi, w)$ where $w$ is a satisfying assignment for $\varphi$, and $C(\varphi)$ fails to find a satisfying assignment on input $\varphi$. By Theorem 3.12, the proof is complete. □

## 4.3 $\mathsf{VAPC}^1$: $\mathsf{VPV}^1 + \mathsf{dWPHP}$

We denote by $\mathsf{VAPC}^1$ the second-order analogue of the first-order theory $\mathsf{APC}^1$ introduced by [Jeř05]. That is, $\mathsf{VAPC}^1 = \mathsf{VPV}^1 + \mathsf{dWPHP}(\mathsf{VPV}^1)$, where $\mathsf{dWPHP}(\mathsf{VPV}^1)$ is a collection of axioms $\mathsf{dWPHP}(F)$, one for each $\mathsf{VPV}^1$ function symbol $F$ stating that $F$ is *not* a surjection if the size of the codomain of $F$ is slightly bigger than the size of its domain. Moreover, we must allow $F$ to have parameters, denoted by $P$ in the formal statement below. Formally, we have the following:

$$\forall n\ \forall P\ \exists Y\ \ (|Y| = n+1, Y < 2^n + 2^n/n)\ \ \forall X\ \ (|X| = n) \quad [F(P,X) \neq Y].$$

Here we interpret $Y$ as a binary integer less than $2^{n+1}$, and "$<$" is the standard ordering on integers.

**Remark 4.6.** *Note the choice of the parameters for the dual Weak Pigeonhole Principle for* $\mathsf{VAPC}^1$*: it says that, for every choice of parameter $P$, every polytime computable map $F(P,\cdot)$ from $N = 2^n$ to $N + N/n$ elements cannot be a surjection. By random guessing, we can get an element outside the range of $F(P,\cdot)$ with probability at least $1/(n+1)$. In contrast, for a stronger system* $\mathsf{V}^1$*, one can use a version of the dual Weak Pigeonhole Principle for maps from $N$ to $N^2$ elements, achieving a much higher success probability of at least $1 - 1/N$ for randomly guessing an element outside the range of a given map. Over* $\mathsf{V}^1$*, the two versions of the* $\mathsf{dWPHP}$ *axioms are equivalent, which is not known over* $\mathsf{VPV}^1$ *(and is false for the relativized versions of* $\mathsf{dWPHP}$*) [Jeř07b].*

**Theorem 4.7** (KPT Witnessing for $\mathsf{VAPC}^1$). *Suppose that, for a $\Sigma_0^B(\mathsf{VPV}^1)$-formula $\varphi$,*

$$\mathsf{VAPC}^1 \vdash \forall n\, \exists C\, \forall Z \quad \varphi(n, C, Z).$$

*Then there are a constant number $\ell$ of $\mathsf{poly}(n)$-time computable functions*

$$A_1(n, R_1),\ A_2(n, R_1, Z_1, R_2),\ \ldots\ ,\ A_\ell(n, R_1, Z_1, \ldots, R_{\ell-1}, Z_{\ell-1}, R_\ell)$$

*and a constant $c \geq 1$ such that, for every $n \geq 1$, the following holds.*

1. *With probability at least $1/n^c$ over uniform randomness $R_1$, for $C_1 = A_1(n, R_1)$, either $\mathbb{N} \vDash \forall Z_1\, \varphi(n, C_1, Z_1)$, or for any $Z_1$ such that $\mathbb{N} \vDash \neg\varphi(n, C_1, Z_1)$, the following holds.*
2. *With probability at least $1/n^c$ over $R_2$, for $C_2 = A_2(n, R_1, Z_1, R_2)$, either $\mathbb{N} \vDash \forall Z_2\, \varphi(n, C_2, Z_2)$, or for any $Z_2$ such that $\mathbb{N} \vDash \neg\varphi(n, C_2, Z_2)$, the following holds.*

$$\vdots$$

$\ell$. *With probability at least $1/n^c$ over $R_\ell$, for $C_\ell = A_\ell(n, R_1, Z_1, \ldots, R_{\ell-1}, Z_{\ell-1}, R_\ell)$, we have $\mathbb{N} \vDash \forall Z_\ell\, \varphi(n, C_\ell, Z_\ell)$.*

*Proof.* In order to derive a KPT Witnessing Theorem for $\mathsf{VAPC}^1$, we first turn $\mathsf{VAPC}^1$ into a universal theory, by making each instance of a $\mathsf{dWPHP}$ axiom a universal formula. We follow an idea of Krajíček [Kra19, Lemma 12.6.2] to apply Skolemization in order to get rid of the existential variable $Y$. That is, for each axiom $\mathsf{dWPHP}(F)$, introduce a new function symbol $F'$ and the following (universal) axiom $\mathsf{dWPHP}(F, F')$:

$$\forall n\ \ \forall P\ \ \forall X\ \ (|X| = n) \quad \big[|F'(n, P)| = n + 1\ \wedge\ F'(n, P) < 2^n + 2^n/n\ \wedge\ F(P, X) \neq F'(n, P)\big].$$

Intuitively, $F'$ is intended to be a multi-valued function that, on inputs $n$ and $P$, should output some (arbitrary) string *not* in the range of $F(P,\cdot)$.

It is clear that each axiom $\mathsf{dWPHP}(F)$ follows from $\mathsf{dWPHP}(F, F')$. So if $\mathsf{VAPC}^1$ proves

$$\forall n\ \ \exists C\ \ \forall Z\ \ \varphi(n, C, Z),$$

then so does the universal theory

$$\mathcal{T} = \mathsf{VPV}^1 \cup \{\mathsf{dWPHP}(F, F') \mid F \text{ is a } \mathsf{VPV}^1 \text{ function symbol}\}.$$

By the KPT Witnessing Theorem 2.16, there is a constant $\ell \geq 1$ and a finite sequence $T_1, \ldots, T_\ell$ of string-terms in the vocabulary of $\mathsf{VPV}^1$ and the set $\{F' \mid F \text{ is a } \mathsf{VPV}^1 \text{ function symbol}\}$ such that $\mathcal{T} \vdash \Phi$, where

$$\Phi\ =\ \forall n\, \forall \vec{Z}\ \ [\varphi(n, T_1(n), Z_1) \vee \varphi(n, T_2(n, Z_1), Z_2) \vee \cdots \vee \varphi(n, T_\ell(n, Z_1, \ldots, Z_{\ell-1}), Z_\ell)]\,.$$

Consequently, if $M$ is a model of $\mathcal{T}$, we have $M \vDash \Phi$. In particular, this is the case for every model $M$ that extends the standard model $\mathbb{N}$ using a valid interpretation over $\mathbb{N}$ for the new function symbols $F'$ (i.e., $M \vDash \mathsf{dWPHP}(F^M, F'^M)$).

Thanks to the property described above, we conclude the argument by a probabilistic analysis of randomly constructed interpretations for the new functions symbols extending the standard model $\mathbb{N}$. The terms $T_i$, $1 \leq i \leq \ell$, may mention some constant number of new function symbols $F'_1, \dots, F'_c$. The idea is to replace each of the occurrences of these functions $F'_j$ with a random choice of a string in the codomain of the corresponding $\mathsf{VPV}^1$ function $F(P, \cdot)$, for some parameter value $P$. Ignoring $F'_j$s, each individual term $T_i$ computes a polytime function, and so can ask for $\mathsf{dWPHP}(F, F')$ for parameterized functions $F(P, \cdot)$ on some $\mathsf{poly}(n)$-bit inputs. Hence, a uniformly randomly chosen string in the codomain of each such $F(P, \cdot)$ will fall outside the range of $F(P, \cdot)$ with probability at least $1/\mathsf{poly}(n)$ (by the parameters of the $\mathsf{dWPHP}$ axioms for $\mathsf{VAPC}^1$). Since each term $T_i$ mentions at most a constant number of $F'_j$s and on any input $n$ there are at most constantly many values of the parameter $P$ that are relevant (the parameter(s) can depend on $n$ and on the constantly many counter-examples), the probability that all $F'_j$s in it are computed correctly by this randomized procedure is at least $1/\mathsf{poly}(n)$. Since the next terms may use the values of $F'_j$s we chose randomly in $T_i$, we need to pass our random strings to all subsequent terms $T_j$, for $i < j \leq \ell$. $\square$

**Theorem 4.8.** *At least one of the following must be true:*

1. $\mathsf{VAPC}^1$ *does* not *prove that* $\mathsf{SAT} \in \text{io-}\mathsf{SIZE}[\mathsf{poly}(n)]$.

2. *There is a constant* $a \geq 1$ *such that* $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}[a]} \subset \mathsf{MA}//a$*, and, for every* $k \geq 1$, $\mathsf{MA}//a \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$ *and* $\mathsf{ZPP}^{\mathsf{NP}[a]} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$.

*Proof.* Suppose the first case does not hold. As in the proof of Theorem 4.5, we use Parikh's Theorem (Theorem 2.18) to conclude that $\mathsf{VAPC}^1$ proves that $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}(n)]$. In turn, this implies that $\mathsf{VAPC}^1$ also proves that $\text{Search-}\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}(n)]$ (as the implication $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}] \Rightarrow \text{Search-}\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$ is provable already in $\mathsf{VPV}^1$ [CK07]).

By the KPT-witnessing for $\mathsf{VAPC}^1$ (Theorem 4.7), we get that a Search-SAT circuit of polynomial size can be learned with probability $\gamma(n) \geq 1/\mathsf{poly}(n)$, using a constant number of EQs, i.e.,

$$\text{Search-SAT} \in \text{FZPP-LEARN}_{\gamma}^{\text{Search-SAT-EQ[const]}}\text{-uniform } \mathsf{SIZE}[c \cdot n^d],$$

where $c, d$ are constants. As explained in Remark 4.2, we can assume without loss of generality that the EQ oracle is never queried on a circuit of size larger than $c \cdot n^d$.

By Lemmas 3.26 and 3.29, the hypothesis of Theorem 3.25 is satisfied, implying Item 2. $\square$

Next we would like to unify Items 1 and 2 of Theorem 4.8 above to conclude that $\mathsf{VAPC}^1$ cannot prove some circuit upper bound. To that end, we would like to formalize within $\mathsf{VAPC}^1$ Item 2 of Theorem 4.8. Item 2 talks about hard languages in complexity classes $\mathsf{ZPP}^{\mathsf{NP}[\mathsf{const}]}$ and $\mathsf{MA}//\mathsf{const}$, none of which is directly expressible within $\mathsf{VAPC}^1$. However, it is shown in [Jeř07a] that $\mathsf{MA}$ can be formalized within $\mathsf{VAPC}^1$, using the ability to count approximately (which comes from the $\mathsf{dWPHP}$ axioms). To formalize $\mathsf{MA}//\mathsf{const}$ within $\mathsf{VAPC}^1$, we add a new, uninterpreted function symbol $\alpha$, so that $\alpha(n, r)$ is intended to represent the bits of the constant-length advice string on input size $n$, dependent on randomness $r$. This yields a relativized version $\mathsf{VAPC}^1(\alpha)$ of the theory $\mathsf{VAPC}^1$, where we can formalize circuit upper bounds for $\mathsf{MA}//a$.

**Corollary 4.9.** *For any constant* $k \geq 1$, $\mathsf{VAPC}^1(\alpha)$ *does* not *prove that* $\mathsf{MA}//\mathsf{const} \subset \text{io-}\mathsf{SIZE}[O(n^k)]$.

*Proof.* Suppose the opposite. Since $\mathsf{VAPC}^1(\alpha)$ proves $\mathsf{MA}//\mathsf{const} \subset \text{io-}\mathsf{SIZE}[O(n^k)]$, it also proves that $\mathsf{SAT} \in \text{io-}\mathsf{SIZE}[\mathsf{poly}]$, and hence also $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$ by Parikh's theorem. We get, via KPT Witnessing, a learning algorithm for polysize Search-SAT circuits, but now this algorithm may also use oracle access to the function symbol $\alpha$. However, since the algorithm is supposed to work for any interpretation of $\alpha$ (as $\alpha$ is not mentioned in the statement formalizing circuit upper bounds for SAT), we may interpret $\alpha$ as the constant 0 function, for example, and get a learning algorithm for Search-SAT that uses only the Search-SAT Equivalence Query oracle. Then, as in the proof of Theorem 4.8, we get the existence of a hard language $L^*$ in $\mathsf{MA}//a$, for some constant-size output, randomness-dependent advice function $\alpha^*$. But $\mathsf{VAPC}^1(\alpha)$ proves that all languages in $\mathsf{MA}//a$, including this $L^*$, have small circuit sizes, which contradicts the soundness of $\mathsf{VAPC}^1(\alpha)$. $\square$

## 4.4 $\mathsf{V}^1 + \mathsf{dWPHP}$

We allow all $\mathsf{VPV}^1$ functions symbols to be part of the vocabulary for $\mathsf{V}^1$ (which forms a conservative extension of $\mathsf{V}^1$), and, for every $\mathsf{VPV}^1$ functions symbol $F$, define $\mathsf{dWPHP}(F)$ to be the following axiom:

$$\forall n\ \exists m\ \forall P\ \exists Y\ (|Y| = m)\ \forall X\ (|X| = n)\quad [F(P, X) \neq Y].$$

Here again we allow parameters $P$ in our $\mathsf{VPV}^1$ functions symbols $F$. Over $\mathsf{V}^1$, this version of $\mathsf{dWPHP}$ is *equivalent* to the version used in the definition of $\mathsf{VAPC}^1$ above (see, e.g., [Jeř07b] for the proof).

**Theorem 4.10** (KPT Witnessing for $\mathsf{V}^1 + \mathsf{dWPHP}$)**.** *Suppose that, for a $\Sigma_1^B(\mathsf{VPV}^1)$-formula $\varphi$,*

$$\mathsf{V}^1(\mathsf{VPV}^1) + \mathsf{dWPHP}(\mathsf{VPV}^1)\ \vdash\ \forall n\, \exists C\, \forall Z(|Z| = n)\ \varphi(n, C, Z).$$

*Then there is a positive integer $d$ such that the following holds. There is a polynomial-time randomized algorithm $A^*$ such that, for every $n \geq 1$, $A^*(1^n)$ makes at most $n^d$ oracle queries, and with probability at least $1 - 2^{-n}$, it outputs a string $C'$ such that*

$$\mathbb{N}\ \vDash\ \forall Z(|Z| = n)\ \varphi(n, C', Z),$$

*assuming $A^*$ has access to an arbitrary counterexample oracle $O(n, C)$, which returns a string $Z \in \{0,1\}^n$ such that $\neg\varphi(n, C, Z)$ if such a counterexample $Z$ exists, or outputs "yes" if $C$ is good for all $Z$s.*[23]

*Proof.* We follow [Kra95, Theorem 7.3.7, attributed to A. Wilkie] and apply Herbrandization to the $\mathsf{dWPHP}$ axioms first. Suppose there is a proof of

$$\forall n\, \exists C\, \forall Z(|Z| = n)\quad \varphi(n, C, Z),$$

using the $\mathsf{dWPHP}(\mathsf{VPV}^1)$ axioms for function symbols $F_1, \ldots, F_k$, for some constant $k \geq 1$. For simplicity, we assume $k = 1$, and use the function symbol $F$; the case of $k > 1$ is similar. We get that there is a $\mathsf{V}^1(\mathsf{VPV}^1)$ proof of

$$\forall n\ [(\exists a\, \forall b\, \exists P\, \forall Y(|Y| = b)\, \exists X(|X| = a)\quad F(P, X) = Y)\ \vee\ (\exists C\, \forall Z(|Z| = n)\quad \varphi(n, C, Z))]\,.$$

[23]In more detail, $A^*(1^n)$ has access to random strings $R = R_1, \ldots, R_\ell$, where $\ell(n) = n^d$, and its success probability is computed similarly to the statement of Theorem 4.7. However, in contrast with that statement, here there are $\ell(n) = n^d$ rounds instead of constantly many rounds, and $A^*(1^n)$ progresses to the next round with success probability $\geq 1 - 2^{-n}$ instead of just $\geq n^{-c}$.

Introduce Herbrand functions $h$ and $H$ to get rid of the universal quantifiers over $b$ and $Y$, respectively, with the axioms $h(n,a) = 2n + a$ and $|H(n,a,P)| = 2n + a$. Note that $h$ is a polytime computable number function, and so it is already in $\mathsf{VPV}^1$. We get that $\mathsf{V}^1(\mathsf{VPV}^1, H)$ proves that

$$\forall n \ [(\exists a \, \exists P \, \exists X(|X| = a) \quad F(P,X) = H(n,a,P)) \ \vee \ (\exists C \, \forall Z(|Z| = n) \quad \varphi(n,C,Z))] \, .$$

Next we apply Herbrandization to get rid of the universal quantifier over $Z$, by introducing a Herbrand function $W$, with the axiom $|W(n,C)| = n$. Thereby we get that $\mathsf{V}^1(\mathsf{VPV}^1, H, W)$ proves

$$\forall n \ [(\exists a \, \exists P \, \exists X(|X| = a) \quad F(P,X) = H(n,a,P)) \ \vee \ (\exists C \quad \varphi(n,C,W(n,C)))] \, .$$

By a relativized version of Buss's Witnessing (Theorem 2.14) applied to $\mathsf{V}^1(\mathsf{VPV}^1, H, W)$, we get a polytime oracle algorithm $A^{H,W}$, with oracle access to functions $H$ and $W$, satisfying the following for any $H$ and $W$. For every $n \geq 1$, $A^{H,W}(1^n)$ outputs $a', P', X', C'$, where $a' \leq \mathsf{poly}(n)$, $|X'| = a'$, and $|C'| \leq \mathsf{poly}(n)$, such that the following holds over $\mathbb{N}$:

$$(F(P',X') = H(n,a',P')) \ \vee \ \varphi(n,C',W(n,C')). \tag{2}$$

Moreover, $A^{H,W}(1^n)$ runs in time at most $n^d$ for some fixed $d$, and in particular makes at most $n^d$ queries to $H$ and $W$.

Consider the following multi-valued function $W^*(n,C)$ as an instantiation of the oracle $W$:

$$W^*(n,C) = \begin{cases} \text{any } Z^*(|Z^*| = n) \text{ such that } \neg\varphi(n,C,Z^*) & \text{if it exists} \\ 0^n & \text{otherwise.} \end{cases}$$

That is, $W^*$ is the counterexample oracle (Teacher) in the Student-Teacher protocol to find a good $C$ such that $\forall Z(|Z| = n) \ \ \varphi(n,C,Z)$. For this $W^*$, the truth of the formula $\varphi(n,C',W^*(n,C'))$ yields that

$$\forall Z(|Z| = n) \ \ \varphi(n,C',Z).$$

Hence, the $C'$ output by the algorithm $A^{H,W^*}(1^n)$ is a correct solution to the search problem we want to solve, *assuming that the left disjunct in Eq. (2) is false*, i.e., that $H(n,a',P') \neq F(P',X')$.

We can always make the latter happen by instantiating $H$ with the following (not necessarily efficiently deterministically computable) multi-valued function:

$$\widehat{H}(n,a,P) = \ \text{any } Y \in \{0,1\}^{2n+a} \text{ such that } \forall X(|X| = a) \ F(P,X) \neq Y,$$

where such a string $Y$ always exists by the dual pigeon-hole principle. Then the algorithm $\widehat{A} = A^{\widehat{H}}$ finds a good string $C$, given access to the counterexample oracle $W^*$. The only problem is that $\widehat{A}$ is not necessarily a polytime algorithm, given the complexity of computing the function $\widehat{H}$.

We use randomness to compute $\widehat{H}$ correctly, with high probability. We answer each call to $\widehat{H}(n,a,P)$ with a uniformly random string $Y \in \{0,1\}^{2n+a}$. For the correctness analysis, note that the range of $F(P,\cdot)$ is of size at most $2^a$, and so the probability that a random $Y \in \{0,1\}^{2n+a}$ is in the range of $F(P,\cdot)$ is at most $2^a/2^{2n+a} = 2^{-2n}$.

Denote by $H^*$ this (randomized) implementation of $\widehat{H}$. Observe that, no matter the queries made by $A$, it will succeed in returning good $Y$s for all at most $n^d$ queries to $\widehat{H}$ with probability at least $1 - n^d \cdot 2^{-2n} \geq 1 - 2^{-n}$, for all sufficiently large $n$, as desired.

The oracle algorithm $A^*(1^n)$ from the statement implements the computation described above, using its counterexample oracle $O(n,C)$ to compute a valid $W^*$, and its randomness $R$ to implement $\widehat{H}$ as $H^* = H^*(R)$. The success probability of $A^*$, its number of oracle queries, and the polynomial upper bound on its running time follow from our discussion. □

Recall that we denote by $\mathsf{Search\text{-}SAT\text{-}EQ}$ the $\mathsf{EQ}$ oracle for $\mathsf{Search\text{-}SAT}$: on a query $C$, where $C$ is a candidate $\mathsf{Search\text{-}SAT}$ circuit on $n$-bit inputs, the $\mathsf{EQ}$ oracle either says "yes", if $C$ is a correct circuit, or provides a formula $\psi$ with a satisfying assignment $w$ (so $\psi(w)$ is true) such that $C(\psi)$ fails to find a satisfying assignment for $\psi$. Note that a single oracle call to $\mathsf{Search\text{-}SAT\text{-}EQ}$ may be simulated with $O(n)$ calls to an $\mathsf{NP}$ oracle (or a single call to an $\mathsf{NP[wit]}$ oracle), but it is not clear if one could simulate access to a general $\mathsf{NP}$ oracle by using access to the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle. So we have $\mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$, where the class on the left appears to be weaker than the class on the right (cf. Proposition 4.13 below). We give a more formal definition next.

**Definition 4.11** (Randomized machines with access to a $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle)**.** A language $L$ is in $\mathsf{BPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}}$ if there is a probabilistic oracle algorithm $A$ running in polynomial time with a query-size bound $s(m) \in \mathsf{poly}(m)$ such that $A$ only asks Equivalence Queries about $m$-input circuits of size at most $s(m)$, and for every deterministic oracle $\mathcal{O}$ that solves $\mathsf{Search\text{-}SAT\text{-}EQ}$, on every input $x$ we have $\Pr_r[A^{\mathcal{O}}(x,r) = L(x)] \geq 2/3$.[24] The class $\mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}}$ is defined similarly.

**Theorem 4.12.** *At least one of the following must be true:*

1. $\mathsf{V}^1 + \mathsf{dWPHP}$ *does* not *prove that* $\mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}(n)]$.
2. *For every* $k \geq 1$, $\mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \not\subseteq \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[O(n^k)]$.

*Proof.* Assume Item 1 is false. Then, arguing as in the proof of Theorem 4.8, we get by the KPT-witnessing for $\mathsf{V}^1 + \mathsf{dWPHP}$ (Theorem 4.10) that

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP\text{-}LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ[poly]}}\text{-}\mathsf{uniform\ SIZE[poly]}.$$

Since we can construct circuits that solve SAT in zero-error polynomial time using an oracle to $\mathsf{Search\text{-}SAT\text{-}EQ}$, it follows by a standard argument that $\Sigma_2^p \subseteq \mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}}$. Moreover, under the inclusion $\mathsf{SAT} \in \mathsf{SIZE[poly]}$, we get via a Karp-Lipton collapse that $\mathsf{PH} \subseteq \Sigma_2^p \subseteq \mathsf{ZPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}}$. Appealing to Lemma 2.3, we get that Item 2 holds. $\square$

While it is believed that $\mathsf{ZPP}^{\mathsf{SAT}} \not\subseteq \mathsf{io\text{-}SIZE[poly]}$, we *unconditionally* show that the *opposite is true* for the case of the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle (which provides some evidence that the $\mathsf{Search\text{-}SAT\text{-}EQ}$ oracle may indeed be strictly weaker than the $\mathsf{SAT}$ oracle). The following result should be contrasted with Item 2 of Theorem 4.12 above; we give the proof in Appendix C.

**Proposition 4.13.** $\mathsf{BPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \subset \mathsf{io\text{-}SIZE[poly]}$.

## 4.5 $\mathsf{VAPC}^2$: $\mathsf{VPV}^2 + \mathsf{dWPHP}$

Here $\mathsf{VAPC}^2 = \mathsf{VPV}^2 + \mathsf{dWPHP}(\mathsf{VPV}^2)$, where $\mathsf{VPV}^2$ is a conservative extension of $\mathsf{TV}^1$ (the second-order version of $\mathsf{T}^1_2$), and $\mathsf{dWPHP}(\mathsf{VPV}^2)$ is the dual weak pigeonhole principle for function symbols of $\mathsf{VPV}^2$, which correspond exactly to all $\mathsf{P}^{\mathsf{NP}}$ algorithms.

**Theorem 4.14.** *At least one of the following must be true:*

1. $\mathsf{VAPC}^2$ *does* not *prove that* $\mathsf{SAT} \in \mathsf{io\text{-}SIZE}[\mathsf{poly}(n)]$.
2. *For every* $k \geq 1$, $\mathsf{ZPP}^{\mathsf{NP}} \not\subseteq \mathsf{io\text{-}SIZE}^{\mathsf{SAT}}[O(n^k)]$.

[24]Note that, when defining LEARN-uniform circuits, the equivalence query oracle is queried on circuits with exactly $n$ input bits. On the other hand, in this definition, we allow the oracle to be queried on circuits with a number of input bits that might be different from the input length of the algorithm.

*In particular, for any constant $k \geq 1$, $\mathsf{VAPC}^2$ does* not *prove $\mathsf{ZPP}^{\mathsf{NP}} \subset \text{io-}\mathsf{SIZE}[O(n^k)]$.*

*Proof.* If $\mathsf{VAPC}^2$ does prove that $\mathsf{SAT} \in \text{io-}\mathsf{SIZE}[\mathsf{poly}(n)]$, then by Parikh's Theorem (Theorem 2.18), we get $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$. By [Bsh+96; Kan82], we conclude that $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$. Since $\mathsf{VAPC}^2$ can talk about $\mathsf{ZPP}^{\mathsf{NP}}$ [Jeř07a], the "In particular" part of the theorem follows. $\square$

**Remark 4.15.** *It is known that, for every $k \geq 1$, $\mathsf{ZPP}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[O(n^k)]$ [Bsh+96]. So the statement "$\mathsf{ZPP}^{\mathsf{NP}} \subset \text{io-}\mathsf{SIZE}[O(n^k)]$" that $\mathsf{VAPC}^2$ cannot prove according to Theorem 4.14 is very close to the unconditionally false statement. (Note however the infinitely often inclusion in Appendix C, which we currently cannot show to hold on all input lengths.)*

## 5 Non-Uniform Circuit Upper Bounds vs. Uniform Lower Bounds

### 5.1 $\mathsf{NP} \subset \text{io-}\mathsf{P/poly}$ vs. $\mathsf{NP} \not\subseteq \mathsf{P}$ in $\mathsf{VPV}^1$

It may be possible that $\mathsf{NP} \not\subseteq \mathsf{P}$ (i.e., $\mathsf{SAT}$ is hard uniformly) but still $\mathsf{NP} \subset \text{io-}\mathsf{P/poly}$ (i.e., $\mathsf{SAT}$ is infinitely often easy nonuniformly). Even if this possibility may be true in the real world, $\mathsf{VPV}^1$ won't be able to prove this!

FORMALIZATIONS. First we formalize the statements we wish to talk about within $\mathsf{VPV}^1$.

- $\mathsf{NP} \subset \text{io-}\mathsf{P/poly}$: It is equivalent to $\mathsf{SAT} \in \text{io-}\mathsf{SIZE}[\mathsf{poly}]$, which in turn is equivalent to $\mathsf{Search\text{-}SAT} \in \text{io-}\mathsf{SIZE}[\mathsf{poly}]$: For constants $c, d > 0$, we introduce the sentence $\Phi_{c,d}$ as

$$\forall m \;\; \exists n \geq m \, \exists C \, (\mathsf{size}(C) \leq cn^d) \;\; \forall (\varphi, w) \leq n \quad [\varphi(w) \;\; \Rightarrow \;\; \varphi(C(\varphi))], \tag{3}$$

  where $C$ is an encoding of a candidate $\mathsf{Search\text{-}SAT}$ circuit of $\mathsf{poly}(n)$ size, $\varphi$ is a formula, and $w$ is an assignment for $\varphi$.

- $\mathsf{NP} \not\subseteq \mathsf{P}$: It is equivalent to $\mathsf{SAT} \notin \mathsf{P}$, which in turn is equivalent to $\mathsf{Search\text{-}SAT} \notin \mathsf{FP}$. The latter means that, for every $\mathsf{VPV}^1$ function $G$ attempting to solve $\mathsf{Search\text{-}SAT}$, the following sentence $\Psi_G$ is true in the standard model:

$$\forall n \;\; \exists m \geq n \, \exists (\varphi, w) \leq m \quad [\varphi(w) \;\wedge\; \neg \varphi(G(\varphi))], \tag{4}$$

  i.e., there are infinitely many input lengths $m$, where there is at least one satisfiable formula $\varphi$ of size $m$ with a satisfying assignment $w$ such that $G$ fails to find a satisfying assignment for this $\varphi$.

The theories $\mathsf{VPV}^1$ and $\mathsf{V}^1$ are known to have the same sets of $\Sigma_1$ theorems (statements like the one in Eq. (4)). So $\mathsf{NP} \neq \mathsf{P}$ is provable in $\mathsf{VPV}^1$ iff it is provable in $\mathsf{V}^1$. On the other hand, for $\Sigma_2$ theorems (statement like Eq. (3)), $\mathsf{VPV}^1$ and $\mathsf{V}^1$ are not believed to be equivalent (see, e.g., [CN10, Chap. VIII]). We will prove the following.

**Theorem 5.1** (KPT vs. Buss)**.** *At least one of the following must be true:*

1. *$\mathsf{VPV}^1$ does not prove that $\mathsf{NP} \subset \text{io-}\mathsf{P/poly}$. Formally, there are no constants $c, d$ such that $\mathsf{VPV}^1 \vdash \Phi_{c,d}$.*

2. *$\mathsf{VPV}^1$ does not prove that $\mathsf{NP} \not\subseteq \mathsf{P}$. Formally, there is a $\mathsf{VPV}^1$ function symbol $G$ such that $\mathsf{VPV}^1 \nvdash \Psi_G$.*

*Proof.* If Item 1 does not hold, then there are constants $c, d$ such that $\mathsf{VPV}^1 \vdash \Phi_{c,d}$. As before, we can apply Parikh's Theorem (Theorem 2.18) and the paddability of SAT to conclude that Search-SAT $\in$ SIZE[poly] is provable in $\mathsf{VPV}^1$. Moreover, the latter inclusion is formalized by a $\forall\exists\forall$ sentence. By the KPT Witnessing for $\mathsf{VPV}^1$ (Theorem 2.16), we get that

$$\text{Search-SAT} \in \text{LEARN}^{\text{Search-SAT-EQ}[\ell]}\text{-uniform SIZE[poly]}$$

for some constant $\ell \geq 0$; recall that here the equivalence queries for an incorrect candidate Search-SAT circuit $C$ must be answered with a pair $(\varphi, w)$: a formula $\varphi$ satisfied by an assignment $w$, but such that $C(\varphi)$ fails to find a satisfying assignment for $\varphi$ (cf. Remark 3.11).

Note that if $\ell = 0$, then NP = P. If Item 2 does not hold, then for every $\mathsf{VPV}^1$ function symbol $G$, we have $\mathsf{VPV}^1 \vdash \Psi_G$. This contradicts the soundness of $\mathsf{VPV}^1$, and we are done.

For $\ell > 0$, the idea is to use the provability of Item 2 in $\mathsf{VPV}^1$ to eliminate the equivalence queries, and thereby reduce to the case of $\ell = 0$, via Buss-style Witnessing.

Let $A(1^n)$ be the polytime learning algorithm for polysize Search-SAT circuits, using at most $\ell$ EQs, guaranteed to exist by the KPT Witnessing Theorem. Let $A_1$ be the following polytime algorithm that attempts to solve Search-SAT: "On input $\varphi$ of size $n$,

- run $A(1^n)$ until it asks its first EQ for some Search-SAT candidate circuit $C$;
- use $C$ on $\varphi$, i.e., output $C(\varphi)$."

If we assume the negation of Item 2, no polytime algorithm (including $A_1$) can solve Search-SAT on all large enough input lengths. Moreover, $\mathsf{VPV}^1$ (or $\mathsf{V}^1$) proves that for every polytime algorithm (including $A_1$), there exist counterexamples (satisfiable formulas where the algorithm fails to find a satisfying assignment) for infinitely many input lengths.

By an analogue of Buss's Witnessing Theorem for $\mathsf{VPV}^1$, Theorem 2.15, (or by Buss's Witnessing for $\mathsf{V}^1$, Theorem 2.14), we get polytime algorithms $f(1^n)$ and $F(1^n)$ such that, for all $n \geq 1$, $F(1^n)$ outputs a pair $(\varphi, w)$, where $\varphi$ is a formula of size $m = f(1^n) \geq n$ that is satisfied by the assignment $w$ but such that $A_1(\varphi)$ fails to find a satisfying assignment for $\varphi$. Note that such a pair $(\varphi, w)$ is an answer to the first EQ of $A(1^m)$. This allows us to eliminate one equivalence query, getting a new polytime learning algorithm $A'$ for Search-SAT circuits, which uses at most $(\ell - 1)$ EQs, as follows: "On input $1^n$,

- compute $m = f(1^n)$;
- run $F(1^n)$ to get a pair $(\varphi, w)$, where $\varphi$ is a formula of size $m = f(1^n)$ satisfied by the assignment $w$;
- run $A(1^m)$ until it asks its first EQ for some Search-SAT candidate circuit $C$;
- answer the EQ for $C$ with $(\varphi, w)$, and continue running $A(1^m)$, asking the remaining at most $\ell - 1$ EQs as usual."

The new learning algorithm $A'(1^n)$ is still a polytime algorithm, albeit a larger polytime than the runtime of $A$, since we now also need to run polytime algorithms $f$ and $F$. On input $1^n$, it will output a Search-SAT circuit $C_m$ for inputs of size $m$, where $n \leq m \leq \mathsf{poly}(n)$, and hence $|C_m| \leq \mathsf{poly}(m) \leq \mathsf{poly}(n)$, which can still be used for input formulas of size $n$ by the paddability of SAT. Since the original algorithm $A(1^n)$ is correct for all input lengths $n$, the new algorithm $A'$ is also correct for all input lengths $n$; we are essentially restricting the old algorithm $A$ to those, infinitely many and at most poly-spaced apart, input lengths $n$ where the first EQ can be successfully eliminated via the algorithms $f$ and $F$.

Thus, the algorithm $A'$ uses at most $(\ell - 1)$ $\mathsf{EQ}$s, and learns $\mathsf{Search\text{-}SAT}$ circuits of polynomial size, for all input lengths $n \geq 1$. We can apply our reasoning above to this new learning algorithm $A'$, and eliminate another $\mathsf{EQ}$. After at most $\ell$ steps, we will eliminate all $\mathsf{EQ}$s, and have an algorithm that solves $\mathsf{Search\text{-}SAT}$ for all input lengths, and still runs in polytime, because $\ell$ is a constant. A contradiction. $\square$

**Connection to "dream breakers".** A line of work starting with [GST07], with followups by [Ats06; BTW10], considered the question of getting efficient "refuters" for any candidate $\mathsf{SAT}$ algorithms, under the assumption that $\mathsf{NP} \neq \mathsf{P}$ (or $\mathsf{NP} \not\subseteq \mathsf{BPP}$). Namely, the task is: for a given polytime algorithm $A$ trying to solve $\mathsf{SAT}$, design another polytime algorithm $R$ that will generate $\mathsf{SAT}$ instances $x$ such that $A(x)$ is wrong. Most relevant to us is the following result of [BTW10] for the case of $\mathsf{Search\text{-}SAT}$: if $\mathsf{NP} \neq \mathsf{P}$, then for every polytime algorithm $A$ attempting to solve $\mathsf{Search\text{-}SAT}$, there is a polytime algorithm $R$ such that, for infinitely many input lengths $n$, $R(1^n)$ outputs a pair $(\varphi, w)$, where $\varphi$ is a formula satisfied by $w$, but such that $A(\varphi)$ fails to find a satisfying assignment for $\varphi$. That is, this refuter $R$ from [BTW10] finds, infinitely often, a satisfiable formula *together with its satisfying assignment* on which the given candidate $\mathsf{Search\text{-}SAT}$ algorithm fails. Such refuters were termed *dream breakers* by [GST07] (attributed to Adam Smith) because they unequivocally show to the algorithm $A$ that it was wrong on a satisfiable formula.

It is still not known if dream breakers exist in the almost-everywhere setting:

> Assuming that $\mathsf{NP} \not\subseteq \mathsf{io\text{-}P}$, is it the case that for every $\mathsf{Search\text{-}SAT}$ algorithm $A$ there is a polytime refuter $R$ such that, for all sufficiently large input lengths $n$, $R(1^n)$ finds a satisfiable formula $\varphi$ of size $n$ and its satisfying assignment $w$, where $A(\varphi)$ fails?[25]

We observe that *if such "almost everywhere" dream breakers $R$ existed*, then we would get the following implication: If $\mathsf{VPV}^1 \vdash \mathsf{NP} \subseteq \mathsf{io\text{-}P/poly}$ (as formalized above), then (in the standard model, over $\mathbb{N}$) $\mathsf{NP} \subseteq \mathsf{io\text{-}P}$. This would follow immediately from the proof of Theorem 5.1, since we could use the assumed refuters $R$ instead of Witnessing functions coming from the $\mathsf{VPV}^1$ provable separation $\mathsf{NP} \neq \mathsf{P}$.

## 5.2 $\mathsf{NP} \subset \mathsf{io\text{-}P/poly}$ vs. $\mathsf{NP} \not\subseteq \mathsf{BPP}$ in $\mathsf{VAPC}^1$

Here we generalize Theorem 5.1 to the randomized case: we will work with $\mathsf{VAPC}^1$, and show the unprovability within $\mathsf{VAPC}^1$ of the conjunction: $\mathsf{NP} \subset \mathsf{io\text{-}P/poly}$ and $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

FORMALIZATIONS. First we formalize the statements we wish to talk about within $\mathsf{VAPC}^1$. We will actually work with a *conservative extension* $\mathsf{VAPC}^1(\alpha)$ due to [Jeř07a], with extra axioms stating that $\alpha(n)$ is a size $n$ truth table of a Boolean function on $(\log n)$-bit inputs that has exponential $(n^{1/4})$ average-case circuit complexity (i.e., cannot be computed by any $n^{1/4}$-size circuit on more than $1/2 + n^{-1/4}$ fraction of inputs). In this extension $\mathsf{VAPC}^1(\alpha)$, one can define a $\mathsf{VPV}^1(\alpha)$ function symbol that can approximate, to within any additive constant error $\epsilon > 0$, the probability $\mathbf{Pr}_r[p(x, r)]$, for any polytime predicate $p(x, r)$ on any input $x$.

- $\mathsf{NP} \subset \mathsf{io\text{-}P/poly}$: The formalization of this statement remains the same: For some constants $c, d > 0$,
$$\forall m \;\; \exists n \geq m \; \exists C \; (\mathsf{size}(C) \leq cn^d) \;\; \forall (\varphi, w) \leq n \quad [\varphi(w) \;\Rightarrow\; \varphi(C(\varphi))], \tag{5}$$

[25]Actually, even if $R(1^n)$ were to find such pairs $(\varphi, w)$ infinitely often, but for only at most polynomially spaced apart input lengths $n$, it would be very interesting, and sufficient for our applications.

where $C$ is an encoding of a $\mathsf{Search\text{-}SAT}$ circuit of $\mathsf{poly}(n)$ size, $\varphi$ is a formula, and $w$ is an assignment for $\varphi$.

- $\mathsf{NP} \not\subseteq \mathsf{BPP}$: It is equivalent to $\mathsf{SAT} \notin \mathsf{RP}$, which in turn is equivalent to $\mathsf{Search\text{-}SAT} \notin \mathsf{FRP}$. The latter means that, for every $\mathsf{VPV}^1$ function $G(x, r)$, defining a randomized multi-valued function mapping input $x$ to an output $G(x, r)$ for a random string $r$, where $|r| \leq \mathsf{poly}(|x|)$, the following holds:

$$\forall n \;\; \exists m \geq n \; \exists (\varphi, w) \leq m \quad [\varphi(w) \;\wedge\; \mathbf{Pr}_r[\neg\varphi(G(\varphi, r))] \geq 1/2] \,, \tag{6}$$

  i.e., there are infinitely many input lengths $m$, where there is at least one satisfiable formula $\varphi$ of size $m$ with a satisfying assignment $w$ such that $G$ fails to find a satisfying assignment for this $\varphi$, on at least $1/2$ of its random strings $r$.

**Theorem 5.2** (KPT vs. Buss: Randomized Case)**.** *At least one of the following must be true:*

1. $\mathsf{VAPC}^1(\alpha)$ *does not prove that* $\mathsf{NP} \subset \mathsf{io\text{-}P/poly}$*.*
2. $\mathsf{VAPC}^1(\alpha)$ *does not prove that* $\mathsf{NP} \not\subseteq \mathsf{BPP}$*.*[26]

*Proof.* If Item 1 does not hold, then, as before, we can apply Parikh's Theorem (Theorem 2.18) and the paddability of $\mathsf{SAT}$ to conclude that $\mathsf{Search\text{-}SAT} \in \mathsf{SIZE}[\mathsf{poly}]$. By the KPT Witnessing for $\mathsf{APC}^1$ (Theorem 4.7), we get that

$$\mathsf{Search\text{-}SAT} \in \mathsf{FZPP\text{-}LEARN}^{\mathsf{Search\text{-}SAT\text{-}EQ}[\ell]}\text{-}\mathsf{uniform\ SIZE}[\mathsf{poly}]$$

for some constant $\ell \geq 0$. Moreover, thanks to the error reduction via parallel repetition as in the proof of Lemma 3.26, we may assume that the success probability of going to a "good" next round of this $\ell$-round learning algorithm is at least $1 - 2^{-n}$, for every round.

Note that if $\ell = 0$, then $\mathsf{NP} \subseteq \mathsf{BPP}$, and we are done. For $\ell > 0$, the idea is to use the provability of Item 2 in $\mathsf{VAPC}^1$ to eliminate the equivalence queries, via Buss-style Witnessing.

Let $A(1^n)$ be the randomized polytime learning algorithm for polysize $\mathsf{Search\text{-}SAT}$ circuits, using at most $\ell$ $\mathsf{EQ}$s, guaranteed to exist by the KPT Witnessing Theorem. Let $A_1$ be the following randomized polytime algorithm that attempts to solve $\mathsf{Search\text{-}SAT}$:

> "On input $\varphi$ of size $n$, run $A(1^n)$ until it asks its first $\mathsf{EQ}$ for some $\mathsf{Search\text{-}SAT}$ candidate circuit $C$. Output $C(\varphi)$."

If we assume the negation of Item 2, by an analogue of Buss's Witnessing Theorem for $\mathsf{VAPC}^1(\alpha)$ (see [Kra95, Section 7.3] and [Jeř04]), we get randomized polytime algorithms $f(1^n)$ and $F(1^n)$ such that, for all $n \geq 1$, with high probability (say, at least $3/4$), $F(1^n)$ outputs a pair $(\varphi, w)$, where $\varphi$ is a formula of size $m = f(1^n) \geq n$ that is satisfied by the assignment $w$ but such that $A_1(\varphi)$ fails to find a satisfying assignment for $\varphi$, on at least $1/2$ of its internal random strings.

This allows us to eliminate one equivalence query, getting a new randomized polytime learning algorithm $A'$ for $\mathsf{Search\text{-}SAT}$ circuits, which uses at most $(\ell - 1)$ $\mathsf{EQ}$s, as follows: "On input $1^n$,

1. compute $f(1^n) = m$ and $F(1^n) = (\varphi, w)$, where $\varphi$ is a formula of size $m$ satisfied by $w$;
2. run $A(1^m)$ until it asks its first $\mathsf{EQ}$ for some $\mathsf{Search\text{-}SAT}$ candidate circuit $C$;

[26]We could even allow $\mathsf{V}^1(\alpha) + \mathsf{dWPHP}(\alpha)$ in this item. Actually, adding the axiom $\alpha$ about hard-on-average functions to $\mathsf{V}^1$ eliminates the need for the $\mathsf{dWPHP}(\alpha)$ axioms, as the latter can be proved by $\mathsf{V}^1(\alpha)$ [Jeř04].

3. answer the EQ for $C$ with $(\varphi, w)$, and continue running $A(1^m)$, asking the remaining at most $\ell - 1$ EQs as usual."

Let us analyze the success probability of this new learning algorithm, by analyzing its success probability of eliminating the first EQ. Step 1 fails with probability at most $1/4$ (the error of the randomized Buss Witnessing). Step 2 fails (chooses bad randomness that doesn't take the learning algorithm to the next good round) with probability at most $2^{-n}$ (by the error reduction performed on the KPT Learning algorithm). Step 3 fails if the Buss-witness $(\varphi, w)$ is not correct for the circuit $C$ produced in Step 2 by $A$. We know this Buss-witness is good for at least $1/2$ of the random strings used by $A$ in Step 2. Thus, the randomness used by $A$ in Step 2 is bad for either making a good transition to the next round of learning, or obtaining a correct Buss-witness, with total probability at most $1/2 + 2^{-n}$ (by the union bound). Hence, the probability that $A'$ successfully eliminates its first EQ is at least $1 - (1/4) - (1/2) - 2^{-n} > (1/5)$ (by the union bound), and so $A'$ successfully learns a correct Search-SAT circuit within $\ell - 1$ rounds of interaction with the EQ oracle with probability at least $(1/5)(1 - 2^{-n}) \geq (1/6)$.

We can again boost the success probability of $A'$ to $1 - 2^{-n}$ via parallel repetition as in Lemma 3.26, keeping $\ell - 1$ as the number of rounds (EQs). The new randomized learning algorithm $A'(1^n)$ is still a polytime algorithm, and is correct on all input lengths. We repeat our process of EQ elimination for at most $\ell$ steps, eventually getting a BPP algorithm for Search-SAT. A contradiction. □

## References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[Arv+95] Vikraman Arvind, Johannes Köbler, Uwe Schöning, and Rainer Schuler. "If NP has polynomial-size circuits, then MA = AM". In: *Theoretical Computer Science* 137.2 (1995), pp. 279–282. DOI: https://doi.org/10.1016/0304-3975(95)91133-B.

[Ats06] Albert Atserias. "Distinguishing SAT from Polynomial-Size Circuits, through Black-Box Queries". In: *21st Annual IEEE Conference on Computational Complexity* (CCC). 2006, pp. 88–95. DOI: 10.1109/CCC.2006.17.

[BH91] Samuel R. Buss and Louise Hay. "On Truth-Table Reducibility to SAT". In: *Inf. Comput.* 91.1 (1991), pp. 86–102. DOI: 10.1016/0890-5401(91)90075-D.

[BKO20] Jan Bydzovsky, Jan Krajíček, and Igor C. Oliveira. "Consistency of circuit lower bounds with bounded theories". In: *Logical Methods in Computer Science* 16.2 (2020). DOI: 10.23638/LMCS-16(2:12)2020.

[BKT14] Samuel R. Buss, Leszek Aleksander Kolodziejczyk, and Neil Thapen. "Fragments of Approximate Counting". In: *J. Symb. Log.* 79.2 (2014), pp. 496–525. DOI: 10.1017/jsl.2013.37.

[BM20] Jan Bydzovsky and Moritz Müller. "Polynomial time ultrapowers and the consistency of circuit lower bounds". In: *Arch. Math. Log.* 59.1-2 (2020), pp. 127–147. DOI: 10.1007/s00153-019-00681-y.

[Bsh+96] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. "Oracles and Queries That Are Sufficient for Exact Learning". In: *J. Comput. Syst. Sci.* 52.3 (1996), pp. 421–433. DOI: 10.1006/jcss.1996.0032.

[BTW10] Andrej Bogdanov, Kunal Talwar, and Andrew Wan. "Hard Instances for Satisfiability and Quasi-one-way Functions". In: *Innovations in Computer Science* (ICS). 2010, pp. 290–300.

[Bus+20] Sam Buss, Valentine Kabanets, Antonina Kolokolova, and Michal Koucký. "Expander construction in $\mathrm{VNC}^1$". In: *Ann. Pure Appl. Log.* 171.7 (2020), p. 102796. DOI: 10.1016/j.apal.2020.102796.

[Bus86] Samuel R. Buss. *Bounded Arithmetic.* Bibliopolis, 1986.

[Bus97] Samuel R. Buss. "Bounded Arithmetic and Propositional Proof Complexity". In: *Logic of Computation.* Springer Berlin Heidelberg, 1997, pp. 67–121.

[CK07] Stephen A. Cook and Jan Krajíček. "Consequences of the provability of $\mathsf{NP} \subseteq \mathsf{P/poly}$". In: *J. Symb. Log.* 72.4 (2007), pp. 1353–1371. DOI: 10.2178/jsl/1203350791.

[CLW20] Lijie Chen, Xin Lyu, and R. Ryan Williams. "Almost-Everywhere Circuit Lower Bounds from Non-Trivial Derandomization". In: *61st IEEE Annual Symposium on Foundations of Computer Science* (FOCS). 2020, pp. 1–12. DOI: 10.1109/FOCS46700.2020.00009.

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge University Press, 2010. DOI: 10.1017/CBO9780511676277.

[Coo75] Stephen A. Cook. "Feasibly Constructive Proofs and the Propositional Calculus (Preliminary Version)". In: *Symposium on Theory of Computing* (STOC). 1975, pp. 83–97. DOI: 10.1145/800116.803756.

[DPV18] Peter Dixon, Aduri Pavan, and N. V. Vinodchandran. "On Pseudodeterministic Approximation Algorithms". In: *43rd International Symposium on Mathematical Foundations of Computer Science* (MFCS). 2018, 61:1–61:11. DOI: 10.4230/LIPIcs.MFCS.2018.61.

[GG11] Eran Gat and Shafi Goldwasser. "Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications". In: *Electron. Colloquium Comput. Complex.* 18 (2011), p. 136.

[GST07] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "If NP Languages are Hard on the Worst-Case, Then it is Easy to Find Their Hard Instances". In: *Comput. Complex.* 16.4 (2007), pp. 412–441. DOI: 10.1007/s00037-007-0235-8.

[GZ11] Oded Goldreich and David Zuckerman. "Another Proof That $\mathsf{BPP} \subseteq \mathsf{PH}$ (and More)". In: *Studies in Complexity and Cryptography.* 2011, pp. 40–53. DOI: 10.1007/978-3-642-22670-0_6.

[Hem89] Lane A. Hemachandra. "The Strong Exponential Hierarchy Collapses". In: *J. Comput. Syst. Sci.* 39.3 (1989), pp. 299–322. DOI: 10.1016/0022-0000(89)90025-1.

[HS65] Juris Hartmanis and Richard Edwin Stearns. "On the computational complexity of algorithms". In: *Trans. Amer. Math. Soc.* 117 (1965), pp. 285–306. DOI: 10.1090/S0002-9947-1965-0170805-7.

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. "The Power of Natural Properties as Oracles". In: *33rd Computational Complexity Conference* (CCC). Vol. 102. 2018, 7:1–7:20. DOI: 10.4230/LIPIcs.CCC.2018.7.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In search of an easy witness: exponential time vs. probabilistic polynomial time". In: *J. Comput. Syst. Sci.* 65.4 (2002), pp. 672–694. DOI: 10.1016/S0022-0000(02)00024-7.

[IW01] Russell Impagliazzo and Avi Wigderson. "Randomness vs Time: Derandomization under a Uniform Assumption". In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 672–688. DOI: 10.1006/jcss.2001.1780.

[IW97] Russell Impagliazzo and Avi Wigderson. "P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing* (STOC). 1997, pp. 220–229. DOI: 10.1145/258533.258590.

[Jeř04] Emil Jeřábek. "Dual weak pigeonhole principle, Boolean complexity, and derandomization". In: *Ann. Pure Appl. Log.* 129.1-3 (2004), pp. 1–37. DOI: 10.1016/j.apal.2003.12.003.

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

[Jeř07a] Emil Jeřábek. "Approximate counting in bounded arithmetic". In: *J. Symb. Log.* 72.3 (2007), pp. 959–993. DOI: 10.2178/jsl/1191333850.

[Jeř07b] Emil Jeřábek. "On independence of variants of the weak pigeonhole principle". In: *J. Log. Comput.* 17.3 (2007), pp. 587–604. DOI: 10.1093/logcom/exm017.

[Jer09] Emil Jerábek. "Approximate counting by hashing in bounded arithmetic". In: *J. Symb. Log.* 74.3 (2009), pp. 829–860. DOI: 10.2178/jsl/1245158087.

[Kab00] Valentine Kabanets. "Easiness Assumptions and Hardness Tests: Trading Time for Zero Error". In: *Proceedings of the 15th Annual IEEE Conference on Computational Complexity* (CCC). 2000, pp. 150–157. DOI: 10.1109/CCC.2000.856746.

[Kan82] Ravi Kannan. "Circuit-size lower bounds and non-reducibility to sparse sets". In: *Information and Control* 55.1 (1982), pp. 40–56. DOI: https://doi.org/10.1016/S0019-9958(82)90382-5.

[KL80] Richard M. Karp and Richard J. Lipton. "Some Connections between Nonuniform and Uniform Complexity Classes". In: *Proceedings of the 12th Annual ACM Symposium on Theory of Computing* (STOC). 1980, pp. 302–309. DOI: 10.1145/800141.804678.

[KL82] Richard M. Karp and Richard J. Lipton. "Turing machines that take advice". In: *L'Enseignement Mathématique* 28 (1982), pp. 191–209.

[KO17] Jan Krajíček and Igor C. Oliveira. "Unprovability of circuit upper bounds in Cook's theory PV". In: *Logical Methods in Computer Science* 13.1 (2017). DOI: 10.23638/LMCS-13(1:4)2017.

[KPS90] Jan Krajíček, Pavel Pudlák, and Jiří Sgall. "Interactive Computations of Optimal Solutions". In: *International Symposium on Mathematical Foundations of Computer Science* (MFCS). Vol. 452. 1990, pp. 48–60. DOI: 10.1007/BFb0029595.

[KPT91] Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti. "Bounded Arithmetic and the Polynomial Hierarchy". In: *Ann. Pure Appl. Log.* 52.1-2 (1991), pp. 143–153. DOI: 10.1016/0168-0072(91)90043-L.

[Kra17] Jan Krajıécek. "Extensions of models of PV". In: *Logic Colloquium '95: Proceedings of the Annual European Summer Meeting of the Association of Symbolic Logic, held in Haifa, Israel, August 9–18, 1995.* Ed. by Johann A. Makowsky and Elena V.Editors Ravve. Lecture Notes in Logic. Cambridge University Press, 2017, pp. 104–114. DOI: 10.1017/9781316716830.011.

[Kra19] Jan Krajíček. *Proof Complexity*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019. DOI: 10.1017/9781108242066.

[Kra92] Jan Krajíček. "No counter-example interpretation and interactive computation". In: *Proceedings of the Workshop Logic From Computer Science*. Vol. 21. Mathematical Sciences Research Institute Publication, Springer-Verlag, 1992, pp. 287–293.

[Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*. Vol. 60. Encyclopedia of mathematics and its applications. Cambridge University Press, 1995.

[KW98] Johannes Köbler and Osamu Watanabe. "New Collapse Consequences of NP Having Small Circuits". In: *SIAM J. Comput.* 28.1 (1998), pp. 311–324. DOI: 10.1137/S0097539795296206.

[LC11] Dai Tri Man Le and Stephen A. Cook. "Formalizing Randomized Matching Algorithms". In: *Log. Methods Comput. Sci.* 8.3 (2011). DOI: 10.2168/LMCS-8(3:5)2012.

[Lê14] Dai Tri Man Lê. "Bounded Arithmetic and Formalizing Probabilistic Proofs". PhD thesis. University of Toronto, 2014.

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. "Pseudodeterministic algorithms and the structure of probabilistic time". In: *Annual Symposium on Theory of Computing* (STOC). 2021.

[MP20] Moritz Müller and Ján Pich. "Feasibly constructive proofs of succinct weak circuit lower bounds". In: *Ann. Pure Appl. Log.* 171.2 (2020). DOI: 10.1016/j.apal.2019.102735.

[MW20] Cody D. Murray and R. Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-polytime from a New Easy Witness Lemma". In: *SIAM J. Comput.* 49.5 (2020). DOI: 10.1137/18M1195887.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs Randomness". In: *J. Comput. Syst. Sci.* 49.2 (1994), pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1.

[Oja04] Kerry Ojakian. "Combinatorics in Bounded Arithmetic". PhD thesis. Carnegie Mellon University, 2004.

[Par71] Rohit Parikh. "Existence and Feasibility in Arithmetic". In: *Journal of Symbolic Logic* 36.3 (1971), pp. 494–508.

[Pic15a] Ján Pich. "Circuit lower bounds in bounded arithmetics". In: *Ann. Pure Appl. Log.* 166.1 (2015), pp. 29–45. DOI: 10.1016/j.apal.2014.08.004.

[Pic15b] Ján Pich. "Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic". In: *Log. Methods Comput. Sci.* 11.2 (2015). DOI: 10.2168/LMCS-11(2:8)2015.

[PS21] Ján Pich and Rahul Santhanam. "Strong co-nondeterministic lower bounds for NP cannot be proved feasibly". In: *Annual Symposium on Theory of Computing* (STOC). 2021.

[Pud92] Pavel Pudlák. "Some relations between subsystems of arithmetic and the complexity of computations". In: *Proceedings of the Workshop Logic From Computer Science*. Vol. 21. Mathematical Sciences Research Institute Publication, Springer-Verlag, 1992, pp. 499–519.

[RST84] Walter L. Ruzzo, Janos Simon, and Martin Tompa. "Space-Bounded Hierarchies and Probabilistic Computations". In: *J. Comput. Syst. Sci.* 28.2 (1984), pp. 216–230. DOI: 10.1016/0022-0000(84)90066-7.

[San09] Rahul Santhanam. "Circuit Lower Bounds for Merlin–Arthur Classes". In: *SIAM J. Comput.* 39.3 (2009), pp. 1038–1061. DOI: 10.1137/070702680.

[SW14] Rahul Santhanam and Ryan Williams. "On Uniformity and Circuit Lower Bounds". In: *Computational Complexity* 23.2 (2014), pp. 177–205. DOI: 10.1007/s00037-014-0087-y.

[TV07] Luca Trevisan and Salil P. Vadhan. "Pseudorandomness and Average-Case Complexity Via Uniform Reductions". In: *Comput. Complex.* 16.4 (2007), pp. 331–364. DOI: 10.1007/s00037-007-0233-x.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999. DOI: 10.1007/978-3-662-03927-4.

[Wig19] Avi Wigderson. *Mathematics and Computation: A Theory Revolutionizing Technology and Science.* Princeton University Press, 2019.

[Zam96] Domenico Zambella. "Notes on Polynomially Bounded Arithmetic". In: *J. Symb. Log.* 61.3 (1996), pp. 942–966. DOI: 10.2307/2275794.

## A On the Power of Equivalence Queries in Circuit Uniformity

In the result stated below, we observe that the ability to make a *single* equivalence query guarantees the existence of *efficient* uniform constructions that are *provably* impossible otherwise. We establish the result for a language $L$ whose deterministic time complexity is only barely super-polynomial, and note that proving a similar result for a language $L \in \mathsf{P}$ would show that $\mathsf{P} \neq \mathsf{NP}$.

We say that a function $\gamma \colon \mathbb{N} \to \mathbb{N}$ is *constructible* if there is a deterministic polynomial-time algorithm that on input $1^n$ outputs $\gamma(n)$.

**Proposition A.1.** *Let $k \geq 1$ be an arbitrary constant. The following results hold.*

1. *Let $\gamma \colon \mathbb{N} \to \mathbb{N}$ be a constructible function such that $\gamma(n) = \omega(1)$. Then there is a language $L \in \mathsf{DTIME}[n^{\gamma(n)}]$ such that $L \in \mathsf{LEARN}^{\mathsf{EQ}[1]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$ and $L \notin \mathsf{P}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$.*

2. *If there is $L \in \mathsf{P}$ such that $L \in \mathsf{LEARN}^{\mathsf{EQ}[1]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$ and $L \notin \mathsf{P}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$, then $\mathsf{P} \neq \mathsf{NP}$.*

*Proof.* Let $k \geq 1$ be arbitrary. In order to prove the first item, consider a constructible function $\gamma$ such that $\gamma(n) = \omega(1)$. Let $L$ be the language defined by the following procedure: on an input $x \in \{0,1\}^n$, simulate the $n$-th Turing machine $M_n$ on input $1^n$ for $n^{\gamma(n)/2}$ steps, and accept $x$ if and only if $M_n$ rejects $1^n$ or does not terminate its computation. By construction, we see that $L \in \mathsf{DTIME}[n^{\gamma(n)}]$. In addition, since $n^{\gamma(n)/2}$ grows faster than any polynomial, by a standard diagonalization argument we get that $L \notin \mathsf{P}$. For this reason, we have $L \notin \mathsf{P}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$. Finally, note that on every input length $n$, $L$ computes either the constant 0 function, or the constant 1 function, since its output depends on the length of $x$ and not on the bits of $x$. Given that checking which case holds can be done via a single equivalence query, it follows that $L \in \mathsf{LEARN}^{\mathsf{EQ}[1]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[1] \subseteq \mathsf{LEARN}^{\mathsf{EQ}[1]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$.

For a proof of the second item, notice that if $L \in \mathsf{P}$ and $\mathsf{NP} \subseteq \mathsf{P}$, then it is possible to answer the equivalence query in polynomial time. This shows that if $L \in \mathsf{LEARN}^{\mathsf{EQ}[1]}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$ then $L \in \mathsf{P}\text{-}\mathsf{uniform}\ \mathsf{SIZE}[n^k]$. □

## B $\mathsf{FP}^{\mathsf{NP}[\mathsf{wit},O(\log n)]}$ Uniformity for NP

Santhanam and Williams [SW14] proved the following.

**Theorem B.1** ([SW14])**.** *For any constant $k \geq 1$,*

$$\mathsf{NP} \not\subset \mathsf{P}_{\|}^{\mathsf{NP}}\text{-uniform } \mathsf{SIZE}[O(n^k)].$$

Here we generalize Theorem B.1 to the case of weaker notion of *search* FC-uniformity. Recall that $\mathsf{P}_{\|}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$ [BH91; Hem89]. We can strengthen Theorem B.1 by relaxing uniformity to a weaker notion of $\mathsf{FP}^{\mathsf{NP}[\mathsf{wit},O(\log n)]}$-uniformity. Here there is no unique circuit, but rather a circuit provided by the uniformity-machine depends on the actual witnesses provided to the NP witness queries.

We shall need the following lemma.

**Lemma B.2** (Implicit in [CK07])**.** *For any $\mathsf{poly}(n)$-computable function $r(n) \leq \mathsf{poly}(n)$, if*

$$\mathsf{SAT} \in \mathsf{FP}^{\mathsf{NP}[\mathsf{wit},r(n)]}\text{-uniform } \mathsf{SIZE}[\mathsf{poly}],$$

*then*

1. $\mathsf{PH} = \mathsf{P}^{\mathsf{NP}[r(\mathsf{poly}(n))+1]}$, *and*
2. $\mathsf{PH} \subset \mathsf{NP}/r(\mathsf{poly}(n))$.

*Proof.* For some $r = r(n)$, suppose we have an $\mathsf{FP}^{\mathsf{NP}[\mathsf{wit},r]}$ algorithm $A$ that, on input $1^n$, constructs a $\mathsf{poly}(n)$-size circuit $C_n$ that correctly decides SAT on all $n$-bit input formulas. Note that $r(n)$ is a polynomial-time computable upper bound on the number of queries made by $A(1^n)$, in the sense that if correct answers are provided, we are guaranteed that at most $r(n)$ queries are made regardless of the witnesses returned by the oracle. By adding dummy queries to the oracle if necessary, we can assume that $A(1^n)$ makes *exactly* $r(n)$ queries.

PROOF OF ITEM 1. Consider the following NP algorithm $M$: "On input $1^n$ and $\alpha \in \{0,1\}^{r(n)}$,

- guess the full transcript of $A$ on input $1^n$, including the answers $a_1, \ldots, a_r \in \{0,1\}$ to all $r = r(n)$ NP oracle calls, with witnesses for all 'YES'-answered calls (i.e., for all those calls $i$ where $a_i = 1$),
- accept if
  - the transcript is valid (assuming the guessed answers to oracle calls are correct),
  - the guessed witnesses for 'YES'-answered oracle calls are correct witnesses,
  - the guessed NP oracle answers match $\alpha$, i.e., $\alpha = a_1 \ldots a_r$."

For each $n$ and $r = r(n)$, there is at least one string $\alpha \in \{0,1\}^r$ such that $M(1^n, \alpha)$ accepts (since there is correct transcript of $A(1^n)$). Let $\alpha_n^* \in \{0,1\}^r$ be the *lexicographically largest* binary string such that $M(1^n, \alpha_n^*)$ accepts.

Such a string $\alpha_n^*$ can be computed in $\mathsf{FP}^{\mathsf{NP}[r(n)]}$ in a straightforward way: Start with the empty string $\alpha$. For $r$ iterations, keep asking the NP oracle if there is an extension $\alpha' \in \{0,1\}^r$ of the prefix $\alpha 1$ such that $M(1^n, \alpha')$ accepts; if the NP oracle answers 'YES', then set $\alpha = \alpha 1$, else $\alpha = \alpha 0$.

The crucial observation is the following.

**Claim B.3.** *Every accepting branch (witness) $W$ of the NTM $M$ on input $(1^n, \alpha_n^*)$ contains a correct transcript of the computation of $A(1^n)$, including a* correct $\mathsf{SAT}$ circuit $C_n$ *within this transcript.*

*Proof of Claim B.3.* Indeed, consider such a transcript $W$. All its 'YES'-answered $\mathsf{NP}$ oracle queries are correct by the definition of $M$. If some 'NO'-answered $\mathsf{NP}$ oracle query is an error, then let $1 \leq i^* \leq r$ be the first 'NO'-answered query that should have been answered 'YES'. But then $\alpha_n^*$ is not the lexicographically largest string accepted by $M$, as there is some extension of the prefix $(\alpha_n^*)_{[1..(i^*-1)]}1$ that is also accepted by $M$. A contradiction. $\square$

Finally, since $\mathsf{SAT} \in \mathsf{SIZE}[\mathsf{poly}]$ implies $\mathsf{PH} = \Sigma_2^p$ by Theorem 2.4, the lemma will follow once we show that $\Sigma_2^p \subseteq \mathsf{P}^{\mathsf{NP}[r(\mathsf{poly}(n))+1]}$. Consider an arbitrary language $L \in \Sigma_2^p$, defined by a formula $\eta(x) = \exists y\, \forall z \;\; R(x,y,z)$, where $y$ and $z$ have the lengths polynomial in the length of $x \in \{0,1\}^n$, and the predicate $R$ is computable in $\mathsf{P}$.

For each pair of strings $(x,y)$ (of polynomially related lengths), define a propositional formula $\varphi_{x,y}(z')$ such that $\forall z' \;\; \varphi_{x,y}(z') \iff \forall z \;\; R(x,y,z)$. Such a formula can be constructed in $\mathsf{poly}(|x|)$-time using the Cook-Levin Theorem that $\mathsf{SAT}$ is $\mathsf{NP}$-complete. Let $m(|x|) \in \mathsf{poly}(|x|)$ be the size of $\neg\varphi_{x,y}$, the negation of $\varphi_{x,y}$.

The following algorithm $E$ will decide $L$: "On input $x \in \{0,1\}^n$,

- for $m = m(n)$, compute $\alpha_m^*$ (as defined above), in polytime, using at most $r(m)$ $\mathsf{NP}$ oracle queries,
- construct the formula $\Psi(x)$ expressing that "$\exists y\, \exists W$ such that
  - $W$ is an accepting branch of the computation of $M(1^m, \alpha_m^*)$,
  - the circuit $C_m$ contained within $W$ is such that $C_m(\neg\varphi_{x,y}) = 0$."
- ask another $\mathsf{NP}$ oracle query to determine if the constructed formula $\Psi(x)$ is satisfiable, accepting if it is, and rejecting otherwise."

First, note that the described algorithm $E$ is computable in $\mathsf{P}^{\mathsf{NP}[r(m)+1]}$ by construction. It accepts $x$ iff $\Psi(x) \in \mathsf{SAT}$. Note that $\Psi(x) \in \mathsf{SAT}$ iff a circuit $C_m$ contained within an accepting computation of $M(1^m, \alpha_m^*)$ (implying that $C_m$ must be a correct $\mathsf{SAT}$ circuit by Claim B.3) says that $\neg\varphi_{x,y}$ is unsatisfiable, for some string $y$. In turn, $\neg\varphi_{x,y} \in \mathsf{UNSAT}$ for some $y$ is equivalent to $\exists y\, \forall z \;\; R(x,y,z)$. Thus $E$ accepts $x$ iff $\eta(x)$, as required.

Proof of Item 2. Rather than computing the values $\alpha_n^*$, we will assume they are given to us as advice. For any language $L \in \mathsf{coNP}$, defined by a formula $\eta(x) = \forall y \;\; R(x,y)$, where $y$ has the length polynomial in the length of $x \in \{0,1\}^n$, and the predicate $R$ is in $\mathsf{P}$. For each string $x$, define a propositional formula $\varphi_x(y')$ such that $\forall y' \;\; \varphi_x(y') \iff \forall y \;\; R(x,y)$. Such a formula can be constructed in $\mathsf{poly}(|x|)$-time using the Cook-Levin Theorem that $\mathsf{SAT}$ is $\mathsf{NP}$-complete. Let $m(|x|) \in \mathsf{poly}(|x|)$ be the size of $\neg\varphi_x$, the negation of $\varphi_x$.

The following $\mathsf{NP}/r(m(n))$ algorithm $E'$ will decide $L$: "On input $x \in \{0,1\}^n$, given $\alpha = \alpha_{m(n)}^*$ as advice of $r(m(n))$ bits,

- guess a string $W \in \{0,1\}^{O(t)}$, where $t = t(m(n))$ is the runtime of $M(1^{m(n)}, \alpha_{m(n)}^*)$,
- accept if
  - $W$ is an accepting branch of the computation of $M(1^{m(n)}, \alpha_{m(n)}^*)$,

– the circuit $C_m$ contained within $W$ is such that $C_m(\neg\varphi_x) = 0$."

For correctness, observe that $E'$ accepts $x$ iff a circuit $C_m$ contained within an accepting computation of $M(1^m, \alpha^*_m)$ (implying that $C_m$ must be a correct $\mathsf{SAT}$ circuit by Claim B.3) says that $\neg\varphi_x$ is unsatisfiable, which is equivalent to saying that $\forall y \;\; R(x,y)$. Thus $E'$ accepts $x$ iff $\eta(x)$, as required.

Hence we get that $\Sigma_2^p \subset \mathsf{NP}/r(\mathsf{poly}(n))$, implying the collapse $\mathsf{PH} = \Sigma_2^p \subset \mathsf{NP}/r(\mathsf{poly}(n))$. □

The following theorem strengthens Theorem B.1.

**Theorem B.4.** *At least one of the following must be true:*

1. $\mathsf{SAT} \notin \mathsf{FP}^{\mathsf{NP}[\mathsf{wit},n^{o(1)}]}$-uniform $\mathsf{SIZE}[\mathsf{poly}(n)]$.
2. *For any constant* $k \geq 1$, $\mathsf{NP} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$.[27]

*In particular, for every* $k \geq 1$,

$$\mathsf{NP} \not\subset \mathsf{FP}^{\mathsf{NP}[\mathsf{wit},n^{o(1)}]}\text{-uniform } \mathsf{SIZE}[O(n^k)].$$

*Proof.* If the first item is false, then by Item 2 of Lemma B.2, we get that $\mathsf{PH} \subset \mathsf{NP}/n$. By Lemma 2.3, we know that, for every $k \geq 1$, $\mathsf{PH}$ contains a language $L_k \notin \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$. By the collapse of the $\mathsf{PH}$, we conclude that each such $L_k \in \mathsf{NP}/n$. Let $M_k$ be a polytime NTM that decides $L_k$, given advice $\{\alpha_n\}_{n\geq 1}$ for input length $n$, where $|\alpha_n| \leq n$. Consider the language $L'_k = \{(x,a) \mid M_k(x,a) \text{ accepts}\}$. Clearly, $L'_k \in \mathsf{NP}$. If it were the case that $L'_k \in \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$, then we would also get that $L_k \in \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$ by fixing the $a$ inputs of the circuit for $L'_k$ to the values $\alpha_n$. Hence, for every $k \geq 1$, $\mathsf{NP} \not\subset \text{io-}\mathsf{SIZE}^{\mathsf{SAT}}[O(n^k)]$. □

## C On the Weakness of the Search-SAT-EQ Oracle

Here we prove Proposition 4.13, re-stated below.

**Proposition C.1.** $\mathsf{BPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}} \subset \text{io-}\mathsf{SIZE}[\mathsf{poly}]$.

*Proof.* Consider any $L \in \mathsf{BPP}^{\mathsf{Search\text{-}SAT\text{-}EQ}}$, decided by a time $O(n^c)$ randomized algorithm $A$, for some $c > 0$, and making Search-SAT-EQ queries about $m$-input circuits of size at most $m^d$ for some fixed $d$. By definition, $A$ decides $L$ given access to any Search-SAT-EQ oracle. For propositional formulas $\varphi$, denote by $\mathcal{A}(\varphi)$ the set of all assignments to the variables of $\varphi$. We argue by cases on hardness of SAT.

Case 1: $\mathsf{SAT} \in \text{io-}\mathsf{P/poly}$. Then, for a fixed $k$, $\mathsf{SAT}_n$ has $O(n^k)$-size circuits, infinitely often. We'll use this sequence of SAT circuits to answer EQs asked by $A$. First, observe that the following NP statement is only true of a proposed Search-SAT circuit $C$ if it errs on some input:

$$\Phi(C) := \exists\psi\; \exists w \in \mathcal{A}(\psi) \;\; \psi(w) = 1 \wedge \psi(C(\psi)) = 0.$$

To simulate a Search-SAT-EQ about $C$, we first check if $\Phi(C)$ is satisfiable. If it is not satisfiable, then we answer that $C$ is a correct circuit. If $\Phi(C)$ is satisfiable, then we perform the standard downward self-reduction to search for $\psi$ and $w$ that witness $\Psi(C)$, and return these. We'll now show that this requires only a polynomial amount of advice.

[27] Actually, by Lemma 2.3, we could use $\mathsf{SIZE}^O$ for any fixed language $L \in \mathsf{PH}$, not just $L = \mathsf{SAT}$.

First, by the time bounds on $A$, no queried circuit can be too large: $|C| \leq O(n^c)$ for every candidate circuit $C$. So, applying a standard Cook-Levin translation to $\Phi(C)$ for any such bounded $C$, we know $|\Phi(C)| \leq O(n^{3c})$. Because we only have $O(n^k)$-size circuits for $\mathsf{SAT}$ infinitely often, it could be the case that every input length of the form $O(n^{3c})$ is not a length where $\mathsf{SAT}$ is easy. However, there are only fixed polynomial gaps between the input lengths we need to query $\Phi$ on. Therefore, infinitely many of these polynomial *ranges* contain a length where $\mathsf{SAT}$ is easy. $\mathsf{SAT}$ is paddable. So, for every $n$, if there is a small $\mathsf{SAT}$ circuit in the associated range of sizes around $|\Phi|$, we can supply that circuit and the appropriate amount of padding to use as advice.

This suffices to answer EQs as outlined above, giving $L \in \mathsf{io\text{-}BPP/poly}$. Standard averaging arguments then show that we can fix good coins for our simulation of $A$, and supply them as $\mathsf{poly}$-bounded advice, putting $L \in \mathsf{io\text{-}P/poly}$, as desired.

Case 2: $\mathsf{SAT} \notin \mathsf{io\text{-}P/poly}$. We use the following result on "anti-checkers".

**Claim C.2** ([Kra17], see also [Pic15a])**.** *For every constant $k \geq 1$, if $\mathsf{SAT} \notin \mathsf{io\text{-}SIZE}[n^{2k}]$, then there is a sequence of sets $R_n$ of satisfiable $n$-bit formulas of size $\mathsf{poly}(n)$, with $|R_n| = \mathsf{poly}(n)$, and every candidate $\mathsf{Search\text{-}SAT}$ circuit of size $O(n^k)$ fails to find a satisfying assignment for some formula in $R_n$.*

The algorithm $A$, on inputs of size $n$, is restricted to ask about $m$-input circuits of at most $m^d$ size, where $m^d$ is at most $n^c$ by the runtime bound on $A$. Because we assumed $\mathsf{SAT} \notin \mathsf{io\text{-}P/poly}$, we have immediately that $\mathsf{SAT}_m \notin \mathsf{io\text{-}SIZE}[m^{2d}]$. So, no circuit queried by $A$ can be a correct circuit for $\mathsf{Search\text{-}SAT}_m$. Furthermore, we can supply as advice the set $R_m$ and a satisfying assignment for each formula in $R_m$, for each $m$ such that $A$ could query with an $m$-input circuit. This is a polynomial amount of information. We can now answer EQs by testing a proposed $m$-input circuit $C$ on each element $\varphi$ of $R_m$, until $C$ produces an assignment that does not satisfy $\varphi$, which is guaranteed to happen by Claim C.2 above. We respond to the query with $\varphi$ and the satisfying assignment for $\varphi$ recorded in our advice. This suffices to answer all possible $\mathsf{EQ}$'s, placing $L \in \mathsf{BPP/poly}$. Finally, we conclude with a standard averaging argument to fix good coins for our simulation of $A$, and supply them as $\mathsf{poly}$-bounded advice, putting $L \in \mathsf{P/poly}$. □

## D Space Complexity: Theory VLV and Branching Programs

Our $\mathsf{LEARN}$-uniform lower bounds and unprovability results for $\mathsf{P}$ and associated theory $\mathsf{VPV}^1$ do not rely on particularly "special" features of the complexity class or theory. The proof of circuit lower bounds uses properties that many computational resources share, including closure under polynomial padding, efficient simulation of Boolean devices, and constructive hierarchy theorems. The proof of non-provability uses properties that many efficient second-order theories share: expressibility of the relevant upper-bound statement and KPT witnessing. Any subclass of $\mathsf{P}$ that enjoys these properties should admit similar $\mathsf{LEARN}$-uniform complexity lower bounds and unprovability results for an associated theory. Here, we demonstrate this explicitly by giving both such results for the class $\mathsf{L}$ (logspace) and associated theory $\mathsf{VLV}$. Formally, we obtain the following:

**Theorem D.1.** *For any constant $k \geq 1$, we have*

$$\mathsf{L} \not\subset \mathsf{FL\text{-}LEARN}^{\mathsf{EQ[const]}}\text{-uniform } \mathsf{BP\text{-}SIZE}[O(n^k)].$$

**Theorem D.2.** *For any constant $k \geq 1$, $\mathsf{VLV}$ does* not *prove that $\mathsf{L} \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$.*

## D.1 Preliminaries

The following definitions of space complexity reflect the exact number of bits required to write down a snapshot of a space-bounded machine. This simplifies diagonalization proofs.

**Definition D.3** (Space Complexity)**.** Fix deterministic multi-tape Turing Machines as a concrete model. Input is presented on a special read-only tape, and output is expected on a special write-only tape. Let $M$ be such a machine with states $Q$ and alphabet $\Sigma$. The *cell complexity* of $M$, denoted $\tilde{s}_M : \mathbb{N} \to \mathbb{N}$, is the maximum number of cells on the work-tape scanned by $M$ on an $n$-bit input. The *space complexity* of $M$ is

$$s_M(n) = \log|\Sigma| \cdot \tilde{s}_M(n) + \log|Q|$$

We will carefully track non-uniformity. To the concrete model above, we add an additional read-only tape containing "advice" — a binary string that depends only on the input length.

**Definition D.4** (Non-Uniform Space Complexity)**.** Let $\alpha = \{\alpha_n\}_{n\in\mathbb{N}}$ be an infinite sequence of binary strings with bounded length: $|\alpha_n| = a(n)$. Let $M$ be a deterministic Turing Machine as above, with an additional read-only *advice tape* that contains $\alpha_n$ whenever the input is of length $n$. The *non-uniform cell complexity* of $M$, denoted $\tilde{s}_M(n, \alpha_n) : \mathbb{N} \to \mathbb{N}$, is the maximum number of cells on the work-tape scanned by $M$ on an $n$-bit input when supplied with $\alpha_n$ on the advice tape. The *non-uniform space complexity* of $M$ with advice $\alpha$ is

$$s_M(n) = \log|\Sigma| \cdot \tilde{s}_M(n, \alpha_n) + \log|Q|$$

We say that a language $\mathcal{L} \in \mathsf{DSPACE}[s(n)]/a(n)$ if there exists an advice sequence $\alpha$ with length-bound $a(n)$ and a Turing Machine $M$ that has non-uniform space complexity $s(n)$ with $\alpha$ deciding $\mathcal{L}$. Just as unrestricted circuits are the non-uniform Boolean device corresponding to $\mathsf{DTIME}$, branching programs are the non-uniform Boolean device corrosponding to $\mathsf{DSPACE}$.

**Definition D.5** (Branching Programs)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function on variables $x_1, \ldots, x_n$. A *Branching Program* is a DAG with one source and two sinks, where every non-sink node has out-degree two. The sinks are labeled with 0 and 1. Each non-sink node is labeled with a variable $x_i$. Each edge is labeled with 0 or 1. Evaluate a branching program by following the unique path induced by an assignment to $x$ until you reach a sink; the sink's label is the output of the program.

Just as with circuits, a family $\{P_n\}_{n\geq 1}$ of $n$-input branching programs is called uniform if there is an algorithm that, given $n$, computes a fixed binary encoding $\langle P_n \rangle$ of the program $P_n$. We'll use the *direct connection* encoding: bit $i$ of $\langle P_n \rangle = 1$ iff $i$ encodes a tuple $(g, h, t, v, b)$ such that $g$ and $h$ are node names and $t$ denotes the *type* of $g$: 1 if $g$ is a sink, and 0 otherwise. If $g$ is a sink, $v = 0$ and $b$ is the label of $g$. Otherwise, $v$ encodes the variable label on $g$, and $b$ encodes the bit we test for on the edge between $g$ and $h$. To capture uniformity with a decision problem, we associate this encoding with the Direct Connection Language (DCL).

**Definition D.6** (Padded DCL for Branching Programs)**.** Let $P = \{P_n\}_{n\geq 1}$ be a circuit family, and let $pad(n) \geq 0$. Define the padded Direct Connection Language for $P$ as follows:

$$pad(n)\text{-}\mathsf{DCL}(P) = \{(n, 1^{pad(n)}, i) \mid \langle P_n \rangle_i = 1\},$$

where $n$ is given in binary, and the padding $1^{pad(n)}$ in unary.

We define $\mathsf{C}$-uniformity for Branching Programs identically to $\mathsf{C}$-uniformity for circuits. Then, just as for $\mathsf{DTIME}$, there is a compressible-counterexample hierarchy theorem for $\mathsf{DSPACE}$.

**Lemma D.7** (Almost Everywhere $\mathsf{L}$ Hierarchy with compressibile counter-examples)**.** *There is an absolute constant $c$ and language $H_b \in \mathsf{DSPACE}[(2b+c)\log n]$ satisfying the following:*

- COUNTEREXAMPLES: *Every candidate $b(\log n)$-space TM $M$ with advice $\alpha = \{\alpha_n\}_{n\in\mathbb{N}}$, for $|\alpha_n| \in o(n)$, that tries to compute $H_b$ will make a mistake an $n$-bit input*
$$x_{error} = \langle M \rangle \circ \alpha_n \circ \pi,$$
*where $\pi \in \{0\}^*$ is a padding string whose length is chosen so that $|x_{error}| = n$, for all sufficiently large input lengths $n \geq 1$.*

- COMPRESSIBILITY OF COUNTEREXAMPLES: *The counterexample input $x_{error} \in \{0,1\}^n$ is efficiently compressible to $|\alpha_n| + O(1)$ bits by dropping the padding $\pi$, and can be efficiently reconstructed from $\langle M \rangle \circ \alpha_n$ by adding back the padding $\pi$ of appropriate length.*

*Proof.* Define $H_b$ to be decidable by the following TM $A$: "On input $x \in \{0,1\}^n$, try to interpret $x = \langle M \rangle \circ \alpha_n \circ \pi$ for some TM $M$ and strings $\alpha_n \in \{0,1\}^*$ and $\pi \in \{0\}^*$. If not possible, then reject. Otherwise, simulate TM $M$ with advice $\alpha_n$ on input $x$ using at most $b \log n$ space, and accept iff $M$ rejects." In space $(2b+c)\log n$, the machine $A$ can detect looping and track simulated space usage, and so correctly simulate every $M$ that executes in at most $b \log n$ space. $\square$

Finally, the definition of $\mathsf{LEARN}$-uniformity for $\mathsf{DSPACE}$ is nearly identical to that for $\mathsf{DTIME}$ — except we must ensure that the machine does not somehow "cheat" by using any oracle tape as storage. The following adaptation of Ruzzo, Simon, Tompa [RST84] oracle-access suffices.

**Definition D.8** (Space Bounded Equivalence Queries)**.** A space-bounded EQ-oracle TM has a query tape and an answer tape in addition to input tape, working tape, and output tape. The query tape is write-only, and the answer tape is read-only. As soon as the machine enters the query state, the query tape is erased. The answer tape has delimiter symbols that allow the machine to traverse the sequence of previous counter-examples.

## D.2 A LEARN-Uniform Branching Program Lower Bound for L

The structure and ideas are the same as in LEARN-Uniform circuit lower bounds for $\mathsf{P}$. Steps 1 and 2 are identical up to substitution of the appropriate complexity resources; they follow directly from the hierarchy theorems and assumptions. Steps 3 and 4 go through through similarly for $\mathsf{L}$ because it is closed under $\mathsf{poly}$-length padding. Steps 5 and 6 are analogous: identical in function but differing in execution. We give abbreviated steps where the difference is not substantial, and mark where specialization for $\mathsf{L}$ was required.

**Base Case: Zero EQs.**

**Theorem D.9** (Implicit in [SW14])**.** *For any constant $k \geq 1$, we have*
$$\mathsf{L} \not\subset \left(\mathsf{FL\text{-}LEARN}^{\mathsf{EQ}[0]}/n^{1/(2k)}\right)\text{-uniform } \mathsf{BP\text{-}SIZE}[O(n^k)].$$

*Proof.* The proof is by contradiction. Suppose $\mathsf{L} \subset \left(\mathsf{L}/n^{1/(2k)}\right)$-uniform $\mathsf{BP\text{-}SIZE}[O(n^k)]$ for some constant $k \geq 1$. We use the uniformity assumption to argue that then there is a constant $k' \geq 1$ such that $\mathsf{L} \subseteq \mathsf{SPACE}[k' \log n]/o(n)$. Finally, we appeal to the classical Space Hierarchy Theorem to get a contradiction. We give more details next.

1. PICK A "HARD" LANGUAGE: Consider $H_b \in \mathsf{L}$ from Lemma D.7 for $b > k$ to be determined.

2. USE UNIFORMITY: Since $H_b \in \mathsf{L}$, our assumption implies that there is a family of branching programs $P = \{P_n\}_{n \geq 1}$ of $dn^k$-size programs $P_n$ computing $H_b$, for some constant $d \geq 0$, such that $n\text{-}\mathsf{DCL}(P) \in \mathsf{L}/n^{1/(2k)}$.

3. PAD DOWN: For $\varepsilon = 1/(2k)$, we can use the same advice-taking logspace algorithm for $n$-$\mathsf{DCL}$ as above to also get that $n^\varepsilon\text{-}\mathsf{DCL}(C) \in \mathsf{L}/m$, where its input length is $m \leq 2n^\varepsilon$.

4. COMPRESS $P$: Since $\mathsf{L} \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$ implies that $\mathsf{L}/n \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$, we get that there is a family $D = \{D_m\}_{m \geq 1}$ of $O(m^k)$-size branching programs $D_m$ for the language $n^\varepsilon\text{-}\mathsf{DCL}(P)$. That is, $D_m(n, 1^{n^\varepsilon}, i) = \langle P_n \rangle_i$, for all $1 \leq i \leq 3k \log n$. Note that each $D_m$ has at most $O((2n^\varepsilon)^k) \leq O(\sqrt{n})$ states.

5. USE $D$ AS ADVICE TO SPEED UP $H_b$ (**L-specific**): Each $D_m$ can be encoded as a binary string of length $d(n) = O(\sqrt{n}(\log n))$. We will use these $d(n)$ bits of advice in the following "evaluator" TM $\mathcal{E}$ that computes $H_b$ on $n$-bit inputs.

   Given as advice a description of $D_m$, TM $\mathcal{E}$ on input $x \in \{0,1\}^n$ does the following:
   (a) Evaluate $D_m(n, 1^{n^\varepsilon}, i)$ over all $1 \leq i \leq 3k \log n$ to identify the start state of $P_n$.
   (b) Evaluate $P_n(x)$ implicitly using "on-demand" logspace-transducer composition:
   - Keep a "pointer" $g$ naming the current state of $P_n$
   - Evaluate $D_m$ on padded $\mathsf{DCL}$-tuples with $g$ fixed and all other entries free. One of two conditions will occur:
     – $g$ is a sink, and we discover its label. Halt and answer accordingly.
     – $g$ is not a sink, so we find exactly two successor states and the variable $v$ tested by $g$. Test $x_v$ and update $g$ to the appropriate successor state. Continue.

6. DIAGONALIZE (**L-specific**): It is clear that, with correct advice, the TM $\mathcal{E}/d(n)$ correctly computes $H_b$ on all $n$-bit strings. The space complexity of $\mathcal{E}$ is at most $O(9k \log n)$ to name three state of $P_n$ plus $O(c \log n)$ for some absolute constant $c$ to evaluate each state of $D_m$ on-demand (see [Vol99], proof of Theorem 4.38). Thus the overall space usage of $\mathcal{E}/d(n)$ is at most $9ck \log n$. Choosing $b > 9ck$, this contradicts Lemma D.7 since $d(n) \in o(n)$.

□

**Induction: Eliminating One EQ.** Just as in Section 3.1.2, we can trade equivalence queries for advice by exploiting the compressibility of counterexamples in the Space Hierarchy Theorem of Lemma D.7. We'll eliminate equivalence queries one by one, once again marking where the proof here differs substantially from the $\mathsf{DTIME}$ case.

**Lemma D.10** (EQ Elimination via Extra Advice)**.** *Suppose, for some constants $k \geq 1$ and $b \geq 4k$, $\mathsf{L} \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$ and*

$$H_b \in \left(\mathsf{FL\text{-}LEARN}^{\mathsf{EQ}[r]}/a(n)\right)\text{-uniform } \mathsf{BP\text{-}SIZE}[O(n^k)],$$

*where $a(n) = n^\delta$ for some $0 < \delta < 1/(2k)$, and $r \geq 1$ is arbitrary. Then*

$$H_b \in \left(\mathsf{FL\text{-}LEARN}^{\mathsf{EQ}[r-1]}/a'(n)\right)\text{-uniform } \mathsf{BP\text{-}SIZE}[O(n^k)],$$

*where $a'(n) \leq (c \log n) \cdot a(n)^k$, for some universal constant $c > 0$.*

*Proof.* Let $P = \{P_n\}_{n \geq 1}$ be the family of branching programs with at most $O(n^k)$ state that the learning algorithm makes the first equivalence query on to find out if $P$ correctly computes $H_b$. Since the first EQ is computable in $\mathsf{FL}/a(n)$, we get that $n\text{-}\mathsf{DCL}(P) \in \mathsf{L}/a(n)$.

We next argue that such a uniform program family $P$ cannot compute $H_b$, and moreover, we will find a compressible counterexample to this first equivalence query.

1. PAD DOWN: We can use the same advice-taking logspace algorithm for $n$-$\mathsf{DCL}$ to also get that $a(n)\text{-}\mathsf{DCL}(C) \in \mathsf{L}/m$, where its input length is $m \leq 2a(n)$.

2. COMPRESS $C$: Since $\mathsf{L} \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$ implies that $\mathsf{L}/n \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$, we get that there is a family $D = \{D_m\}_{m \geq 1}$ of $O(m^k)$-size circuits $D_m$ for the language $a(n)\text{-}\mathsf{DCL}(C)$. Each $D_m$ has at most $O((2a(n))^k) \leq O(n^{\delta k}) \leq O(\sqrt{n})$ states, and so can be encoded as a binary string $\beta_n$ of length $d(n) = O(n^{\delta k}(\log n)) \in o(n)$.

3. EVALUATOR TM $\mathcal{E}$ **($\mathsf{L}$-specific)**: Given as advice a description $\beta_n$ of $D_m$, TM $\mathcal{E}$ on input $x \in \{0,1\}^n$ repeatedly evaluates $D_m(n, 1^{a(n)}, i)$ over all $1 \leq i \leq 3k \log n$ to evaluate $P_n(x)$ using implicit logspace composition, exactly as in the proof of Theorem D.9 above.

4. DIAGONALIZE: **($\mathsf{L}$-specific)** Since the space complexity of $\mathcal{E}$ is at most $9kc \log n$, it cannot compute $H_b$ by Lemma D.7. Moreover, $\mathcal{E}$ with advice $\beta_n$ disagrees with $H_b$ on the input

$$x_{error} = \langle \mathcal{E} \rangle \circ \beta_n \circ \pi,$$

for some $\pi \in \{0\}^*$ of appropriate length so that $|x_{error}| = n$.

5. ELIMINATE THE EQUIVALENCE QUERY THROUGH ADVICE: Since, for the advice $\beta_n$, we have by construction that $(\mathcal{E}/\beta_n)(x) = C_n(x)$ for all $x \in \{0,1\}^n$, it follows that

$$C_n(x_{error}) \neq H_b(x_{error}).$$

We add to the advice of our learning algorithm the succinct encoding $\langle \mathcal{E} \rangle \circ \beta_n$ of $x_{error}$ to be used as the answer to the first equivalence query, and eliminate that query. The new learning algorithm now makes $(r-1)$ equivalence queries, and has advice of length $a(n) + O(a(n)^k(\log a(n))) \leq O(a(n)^k(\log n))$, as claimed. Even though it interacts with the oracle access mechanism, this step is effectively identical to the argument for $\mathsf{P}$. We only replaced the *first* query and the advice tape is read-only — just as the query answer tape.

□

**Concluding the Proof of Theorem D.1** Towards a contradiction, suppose that, for some constants $k$ and $r$, $\mathsf{L} \subset \mathsf{FL\text{-}LEARN}^{\mathsf{EQ}[r]}\text{-}\mathsf{uniform\ BP\text{-}SIZE}[O(n^k)]$. We'll apply Lemma D.10 for $r$ rounds, to get a learning algorithm with no EQs and advice size $a_r(n)$. First, set $a_0(n) = n^\gamma$ for $\gamma = 1/(3k^{r+1})$ as determined in the proof of Theorem 3.6. The same parameters will work for $\mathsf{L}$, because query-elimination accrues advice at the same rate. So, imagine giving a (useless) advice string $1^{a_0(n)}$ to our assumed learning algorithm for $H_b$. We apply Lemma D.10 getting a learning algorithm with no EQ's and advice size

$$a_r(n) \leq (c \log n)^{rk^{r-1}} \cdot a_0(n)^{k^r}.$$

Setting $\gamma = 1/(3k^{r+1})$ ensures that $a_r(n) < n^{1/(2k)}$, and we get a contradiction by Lemma D.9.

## D.3 Unprovability of Non-Uniform Branching Program Upper Bounds in VLV

Just as P has associated theory VPV, L has associated theory VLV [CN10], capturing ”logspace reasoning”. The language of VLV contains function symbols for all functions in FL. Similarly to VPV, VLV is a universal theory, and thus KPT witnessing theorem applies, with logspace-computable terms.

For complexity classes within P, string functions have to be defined by their bit graphs: that is, each bit of the output of a function is computable by a predicate of the respective complexity, representable in the language of the theory. In particular, for each string function $F(\bar{x}, \bar{Y})$ in the language of VLV, there is a predicate $B_F(i, \bar{x}, \bar{Y}) \leftrightarrow F(\bar{x}, \bar{Y})(i)$, where $B_F()$ is computable in logspace, and thus by a branching program.

**Formalizing Branching Program Upper Bounds.** In VLV, branching program size and branching program evaluation $P(X)$ are all expressible in a straightforward way. We already used space-efficient evaluation of branching programs in the proofs above, and it is clear that there is a function in FL which returns the number of states in a given branching program under any reasonable fixed encoding of branching programs. Finally, $O(n^k)$ is logspace-constructible. So, VLV can indeed formalize “$\mathsf{L} \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$”, just as VPV formalizes “$\mathsf{P} \subset \mathsf{SIZE}[O(n^k)]$” in Section 4.

**KPT Witnessing.** Recall that the KPT Witnessing of Theorem 2.16 requires only that $T$ be a universal theory. When applied to VLV, the resulting sequence of string-terms is a composition of FL functions, meaning that the “student” in the resulting Student-Teacher protocol is an FL function. Thus, KPT-witnessing a VLV-proof gives $\mathsf{FL\text{-}LEARN}^{\mathsf{EQ[const]}}$ constructions of the implicated objects. Combining this with the above, we get

*Proof of Theorem D.2.* Suppose there exists some constant $k \geq 1$ such that VLV proves “$\mathsf{L} \subset \mathsf{BP\text{-}SIZE}[O(n^k)]$.” Then by the KPT Witnessing for VLV, we get that

$$\mathsf{L} \subset \mathsf{FL\text{-}LEARN}^{\mathsf{EQ[const]}}\text{-uniform } \mathsf{BP\text{-}SIZE}[O(n^k)].$$

contradicting Theorem D.1. □