Electronic Colloquium on Computational Complexity, Report No. 167 (2019)

UTIME Easy-witness Lemma & Some Consequences

Anant Dhayal
University of California San Diego, La Jolla, CA, USA
adhayal@eng.ucsd.edu

Russell Impagliazzo
University of California San Diego, La Jolla, CA, USA
russell@eng.ucsd.edu

Abstract

We prove an easy-witness lemma (EWL) for unambiguous non-deterministic verfiers. We show that if $\texttt{UTIME}(t) \subset \mathcal{C}$, then for every $L \in \texttt{UTIME}(t)$, for every $\texttt{UTIME}(t)$ verifier V for L, and for every $x \in L$, there is a certificate y satisfing $V(x, y) = 1$, that can be encoded as a truth-table of a $\mathcal{C}$ circuit. Our technique is simple compared to the NTIME EWLs [12, 38, 26], and yields fine-grained results in terms of the time and size parameters. It also works for all *typical* non-uniform circuit classes without any additional machinery. Using this EWL we prove a Karp-Lipton [20] style theorem (KLT) for UEXP. We show that $\texttt{UEXP} \subset \texttt{SIZE}(poly) \implies \texttt{UEXP} = \texttt{MA}$. We also prove similar EWL and KLT for $\texttt{UEXP} \cap \texttt{Co-UEXP}$ and FewEXP.

Circuit lower bound techniques that entail natural properties of Razborov and Rudich [31] are called natural, and are known to contradict widely believed cryptographic assumptions in the course of proving strong lower bounds. Thus attempts have been made to understand un-natural techniques. Natural properties satisfy three conditions: usefulness, constructiveness, and largeness. Usefulness is unavoidable in any lower-bound technique. In [36, 29] it was shown that obtaining NEXP lower bounds is equivalent to obtaining P-constructive (with $\log n$ advice) properties.

In this paper we consider properties that avoid largeness. We introduce a new notion called unique properties, which is opposite to natural properties in the sense of largeness. A unique property contains exactly one element of each input length (that is a power of 2). We show that P-constructivity and uniqueness (opposite of largeness) both are unavoidable for certain lower bounds. We prove, $\texttt{UEXP} \cap \texttt{Co-UEXP} \not\subset \mathcal{C}$ if and only if there is a P-constructive unique property against $\mathcal{C}$. We also establish equivalences between lower bounds against UEXP (with and without advice), and the existence of different restrictions of P-constructive unique properties that use advice.

The "derandomization (of BPP) from uniform/non-uniform lower bounds for Γ" type of results are known for $\Gamma = \texttt{EXP}, \texttt{NEXP}, \texttt{NEXP} \cap \texttt{Co-NEXP}, \texttt{REXP}$ [28, 4, 16, 12, 36]. Using the above equivalences we obtain a super-set of these results that also includes the classes $\texttt{UEXP}, \texttt{UEXP} \cap \texttt{Co-UEXP}, \texttt{ZPEXP}$.

One important application of the NEXP EWL and KLT is the connection between fast (SAT and learning) algorithms and NEXP lower bounds [38, 8, 29]. Using our UTIME EWL and KLT we derive connections between fast unambiguous algorithms and UTIME lower bounds. Finally we show results that generalize the lower bound frameworks – that work only for unrestricted Boolean circuits – such that they work for any restricted typical circuit class. This will help us to get lower bounds against any typical circuit class from fast algorithms that work for that particular class (and not for the super-class of unrestricted Boolean circuits).

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Circuit complexity

Keywords and phrases easy-witness lemma, lower bounds, unique-properties, derandomization

Funding Work supported by the Simons Foundation and NSF grant CCF-1909634

Acknowledgements We want to thank Marco Carmosino, Sasank Mouli and Sam McGuire for useful discussions, and for comments and corrections on the manuscript.

ISSN 1433-8092

1 Introduction

We often think of algorithm design and lower bounds as being antithetical, but there have been a series of results showing that in certain circumstances, efficient algorithms imply circuit lower bounds [12, 38, 39, 37, 26, 19]. Unfortunately, most of these results are only known to show circuit lower bounds or conditional lower bounds in relatively large complexity classes such as NEXP or $\mathtt{E}^{\mathtt{NP}}$ (although [26] extends this to scaled-down versions of these classes). This raises the question of whether similar lower bounds for other classes, ideally deterministic or randomized classes such as EXP or BPEXP, could be obtained through improved algorithms. Here, we consider possible extensions to the class UEXP of languages recognized by unambiguous non-deterministic machines, and to related classes. Since UEXP lies between EXP and NEXP, lower bounds for UEXP based on algorithms would be progress towards making similar connections for EXP.

A key technique used to make these connections is the "easy witness technique" ([18, 12, 26]). The easy witness technique relates the circuit complexity of witnesses for non-deterministic algorithms to the circuit or algorithmic complexity of decision problems. We give easy witness lemmas for UEXP and related classes; these are much simpler than the analogous results for NEXP, which needed a rather indirect argument. We then explore consequences of these easy witness lemmas to normal forms for circuit lower bounds in these classes, in terms of useful properties in the sense of Razborov and Rudich ([31], see also [36]). We show how a combination of faster learning algorithms and SAT algorithms for a circuit class would imply a circuit lower bound for UEXP.

Another application of the easy witness technique has been to prove "Karp-Lipton" style theorems ([20]), relating the non-uniform and uniform complexities of classes. An example is Meyer's Theorem from [20]: $\mathtt{EXP} \subset \mathtt{P/poly} \implies \mathtt{EXP} = \Sigma_2$. An extension to NEXP was given in [12], using the easy witness technique. We give analogous Karp-Lipton style results for UEXP and related classes (Section 3).

More particularly, we derive analogous EWL and KLT for UTIME and $\mathtt{UTIME} \cap \mathtt{Co\text{-}UTIME}$. Our results are fine-grained in terms of the time and size parameters, and work for all *typical* non-uniform circuit classes. We look at EWL as a special search to decision reduction, where the output of the search problem is canonical in some natural way. For language L and non-deterministic verifier V for L, we define the language

$$L_{ewl(V)} = \{(x, i) \mid \exists y \; [V(x, y) = 1 \wedge y_i = 1 \wedge \forall (z <_{l.o.} y) \; V(x, z) = 0]\} \tag{1}$$

where $z <_{l.o.} y$ stands for "z is lecigraphically smaller than y", y_i is the short-hand for the i^{th} bit of the string y, and the subscript $ewl(V)$ in $L_{ewl(V)}$ stands for "easy-witness language for V". We prove that, for $L \in \mathtt{UTIME}(t)$, and $\mathtt{UTIME}(t)$ verifier V for L, $L_{ewl(V)} \in \mathtt{UTIME}(t)$. Thus, $L_{ewl(V)}$ also has circuits from any class $\mathcal{C}$ that L has, and we get the desired EWL, which in turn gives the desired $\mathtt{KLT} : \mathtt{UEXP} \subset \mathtt{P/poly} \implies \mathtt{UEXP} = \mathtt{MA}$. Similar results for related classes are also derived.

1.1 Useful Properties

Razborov and Rudich [31] defined the concept of natural property as a formalization of a barrier that circuit lower bounds need to circumvent. Natural Proofs (or properties) satisfy three conditions: they are constructive (an efficient algorithm $\mathcal{A}$ is embedded in them), have largeness ($\mathcal{A}$ accepts a large fraction of strings), and are useful ($\mathcal{A}$ rejects all strings which are truth tables of small circuits). Circuit lower bound techniques that entail natural properties are called natural, and are known to contradict widely believed cryptographic

assumptions in the course of proving strong lower bounds. Thus they are self-limiting, and in order to prove stronger circuit lower bounds the techniques should be un-natural in some sense. Unfortunately, the vast majority of known circuit lower bound techniques are natural and can't be applied even to low-level complexity classes such as $\mathtt{TC}_0$ [27, 23, 25].

Williams [36] showed, using the easy witness lemma, that any lower bound for a problem in NEXP implies a property with two of the conditions (constructivity and usefullness) of Razborov and Rudich, but not necessarily the third (largeness). So while natural properties for circuit classes cannot exist if there are strong pseudo-random functions in the class, it seems likely that dropping largeness means that such properties do exist.

Our results (Section 4): To understand properties that avoid largeness, we look at properties that go to the extreme in the other direction. We introduce a new notion called unique properties, those that contain exactly one function of each input length. Useful, unique properties are implicitly proving a circuit lower bound for a specific function: the one function that has the property, but might not explicitly spell out which function the lower bound holds for.

We extend the proofs in [36, 29] to show that: obtaining NEXP lower-bounds is equivalent to obtaining useful NP-unique (with $\log n$ advice) properties; and obtaining $\mathtt{NEXP} \cap \mathtt{Co\text{-}NEXP}$ lower-bounds is equivalent to obtaining useful NP-unique (without advice) properties. The next task is to prove equivalence with P-unique properties. So in attempt to understand P-unique properties better, we take the next obvious step and move to UEXP lower bounds.

We show that P-constructivity and uniqueness both are unavoidable for UEXP lower bounds. We prove, $\mathtt{UEXP} \cap \mathtt{Co\text{-}UEXP} \not\subset \mathcal{C}$ if and only if there is a P-unique property against $\mathcal{C}$. We also establish equivalences between lower bounds against UEXP (with and without advice), and the existence of different restrictions of P-unique properties that use advice.

1.2 Derandomization from Lower Bounds

Apart from proving lower bounds and showing limitations of the current lower bound techniques, another interesting line of research that has received a lot of attention is, the study of the consequences of lower bounds in a hypothetical world where they exist. One way in which this world is better is – non-tirivial derandomization [28, 4, 16, 12, 36, 34, 32, 15, 5] – which otherwise seems very difficult to achieve. Lower bounds can be viewed as hardness (of a certain class over the other), which when fed to the "hardness to randomness" connections, results in derandomization.

The following result of [16]

$$\mathtt{EXP} \neq \mathtt{BPP} \implies \forall \epsilon > 0\ \mathtt{BPP} \subset \textit{io-Heur-}\mathtt{DTIME}(2^{n^\epsilon}) \tag{2}$$

was generalized in [36] by replacing EXP with NEXP or REXP, and DTIME with NTIME or ZPTIME (using Ko's Theorem [22]). In the process they also proved several intermediate results including a connection between, the non-existence of P-natural properties, and the derandomization of ZPEXP and REXP. Generalizing this connection and other intermediate results we get a cleaner and more general set of "lower bounds to derandomization" results.

Our results (Section 5): Generalizing the definition of ZPTIME, for $\mathtt{C} = \mathtt{N}, \mathtt{R}, \mathtt{U}$ we define $\mathtt{ZCTIME}(t)$ to be the class of languages that are accepted by $\mathtt{CTIME}(t)$ machines, that on any input and any computation branch, either output the correct answer, or output '?' (don't know). With the formal definition of ZCTIME in the next section, we will also see that $\mathtt{ZCTIME} = \mathtt{CTIME} \cap \mathtt{Co\text{-}CTIME}$. Even though they are equal, in the case of non-uniform advice, ZCTIME can be used to capture more information in certain cases – for example, when languages L and $\overline{L}$ have $\mathtt{CTIME}(t)/a$ algorithms $\mathcal{A}$ and $\mathcal{A}'$ that use the same advice,

$L \in \texttt{ZCTIME}(t)/a$ is always true but $L \in (\texttt{CTIME}(t) \cap \texttt{Co-CTIME}(t))/a$ is not (unless $\mathcal{A}$ and $\mathcal{A}'$ complement each other on every advice string - correct or incorrect). So we need to settle with $L \in \texttt{CTIME}(t)/a \cap \texttt{Co-CTIME}(t)/a$, which is a loss of the information – that both the algorithms use the same correct advice. Also, the presentation is not clean.

We show the following for $\texttt{C} = \texttt{N}, \texttt{R}, \texttt{U}$:

(a) $\texttt{CEXP} \not\subset \texttt{SIZE}(poly)$ or $\texttt{CEXP} \neq \texttt{EXP} \implies \forall \epsilon > 0\ \texttt{BPP} \subset \text{io-}\texttt{ZCTIME}(2^{n^\epsilon})/n^\epsilon$

(b) $\texttt{CEXP} \neq \texttt{BPP} \implies \forall \epsilon > 0\ \texttt{BPP} \subset \textit{io-Heur-}\texttt{ZCTIME}(2^{n^\epsilon})/n^\epsilon$

(c) $\texttt{CEXP} \neq \texttt{MA} \implies \forall \epsilon > 0\ \texttt{BPP} \subset \textit{io-}\texttt{ZCTIME}(2^{n^\epsilon})/n^\epsilon$

(d) $\texttt{ZCEXP} \not\subset \texttt{SIZE}(poly)$ or $\texttt{ZCEXP} \neq \texttt{EXP} \implies \forall \epsilon > 0\ \texttt{BPP} \subset \text{io-}\texttt{ZCTIME}(2^{n^\epsilon})$

(e) $\texttt{ZCEXP} \neq \texttt{BPP} \implies \forall \epsilon > 0\ \texttt{BPP} \subset \textit{io-Heur-}\texttt{ZCTIME}(2^{n^\epsilon})$

(f) $\texttt{ZCEXP} \neq \texttt{MA} \implies \forall \epsilon > 0\ \texttt{BPP} \subset \text{io-}\texttt{ZCTIME}(2^{n^\epsilon})$

To the best of our knowledge: These result were not know for $\texttt{C} = \texttt{U}$. For $\texttt{C} = \texttt{R}$ only (a) and (b) were known [36], but not the others. For $\texttt{C} = \texttt{N}$ stronger versions of (a),(c),(d),(f) were known [12], and (b) was known [36], but not (e).

1.3 Fast algorithms to non-uniform lower bounds

Other than proving uniform and non-uniform lower bounds researches have also shown interest in improving upper-bounds by designing better algorithms. Ckt-SAT is the canonical NP complete problem [7, 24] and no algorithm faster than the trivial brute-force algorithm is known to solve Ckt-SAT. Another interesting question is, whether non-determinism helps in solving Ckt-TAUT (a Co-NP complete problem) faster than the trivial brute-force algorithm. Negative answer to these questions lead to the formulation of several conjectures [14, 13, 6].

In [38] they show that if we get super-polynomial savings in any non-deterministic algorithm for Ckt-TAUT of polynomial size circuits then $\texttt{NEXP} \not\subseteq \texttt{SIZE}(poly)$. This shows that designing fast SAT algorithms is at least as hard as proving non-uniform circuit lower bounds.

Our results (Section 6): We show that (using UTIME EWL and KLT), if unambiguous non-determinism helps in solving Ckt-SAT, Ckt-TAUT and other related problems, in faster than the trivial brute-force algorithms, then we get lower bounds for UTIME (with advice). As fast algorithms (that beat brute-force with a good margin) have only been designed for restricted classes, we give results that generalize the lower bound frameworks – that work only for unrestricted Boolean circuits – such that they work for any restricted typical circuit class (to get lower bounds against them). The current best algorithms for these restricted classes are still not good enough to yield unconditional lower bounds, but its good to have these generalizations, as progress in the restricted setting seems more likely than in the unrestricted setting.

2 Preliminaries

Notations: We use t to denote time-constructible functions $n \leq t(n) \leq 2^{O(n)}$, Γ for uniform complexity classes, a for advice functions $0 \leq a(n) \leq poly(n)$, s for circuit sizes (number of wires) $n \leq s(n) \leq 2^n$ – unless a new range is declared during the usage. For language L we use $L_n = \{x \mid x \in L \wedge |x| = n\}$ to denote the n^{th}-slice of L. For circuit C, we use $tt(C)$ to denote its truth-table, $|C|$ to denote its size.

Uniform classes: We assume that the reader is familiar with the standard complexity classes such as P, NP, RP, UP, BPP, ZPP, AM, MA, PH, Σ_2, Π_2 (see [3]) and their corresponding complexity measures, DTIME, NTIME, RTIME, UTIME, BPTIME, ZPTIME. For $\texttt{C} = \texttt{D}, \texttt{N}, \texttt{R}, \texttt{U}, \texttt{BP}, \texttt{ZP}, \texttt{CE}$ denotes the class $\texttt{CTIME}(2^{O(n)})$, and CEXP denotes the class $\cup_{c \geq 0}\texttt{CTIME}(2^{O(n^c)})$. $\texttt{CTIME}(t)$

denotes the class of languages accepted by $\texttt{CTIME}$ machines that run in $O(t)$ time. We assume familiarity with $\texttt{SAT}$ (satisfiability), $\texttt{TAUT}$ (tautology), $\texttt{k-SAT}$, $\Sigma_2\texttt{-SAT}$ and $\Pi_2\texttt{-SAT}$.

Circuit classes: We assume basic familiarity with Boolean circuits and their sub-classes. We use $\mathcal{C}$ to denote any *typical* non-uniform circuit class, i.e., any class from the set $\{\texttt{AC}_0, \texttt{ACC}_0, \texttt{TC}_0, \texttt{NC}_1, \texttt{NC}, \texttt{P/poly}\}$. All these circuit classes are of polynomial size. We use $\mathcal{C}(s)$ to denote the class of $O(s)$-size $\mathcal{C}$ circuits. For truth-table tt, we use $ckt_{\mathcal{C}}(tt)$ to denote its exact $\mathcal{C}$ circuit complexity – the minimum size of any $\mathcal{C}$ circuit whose truth-table is the string tt. In the case of unrestricted Boolean circuits, instead of $\mathcal{C}(s)$ and $ckt_{\mathcal{C}}(tt)$, we use $\texttt{SIZE}(s)$ and $ckt(tt)$.

Non-uniform classes: $L \in \Gamma/a$ if there exists a Γ Turing machine M, and advice sequence $\{a_n\}_{n\in\mathbb{N}}$ satisfying $\forall n\ |a_n| = a(n)$, such that for any n-length input x, $x \in L \iff M(x, a_n) = 1$. For semantic classes, the machine M only needs to satisfy the semantic promise on the advice sequence $\{a_n\}_{n\in\mathbb{N}}$ (and not on all advice strings).

Heuristic classes: For uniform/non-uniform class Λ, $L \in Heur\text{-}\Lambda$ if $\exists L' \in \Lambda$, such that for all polynomially samplable distribution $\mathcal{D}$, $\forall n\ Pr_{x\sim\mathcal{D},|x|=n}[L_n(x) = L'_n(x)] \geq 1 - \frac{1}{n}$.

Infinitely-often classes: For uniform/non-uniform, heuristic/non-heuristic class Λ, $L \in io\text{-}\Lambda$ if $\exists L' \in \Lambda$, and an infinite subset $S \subset \mathbb{N}$, such that $n \in S \implies L_n = L'_n$.

Zero-error classes: $L \in \texttt{ZCTIME}(t)$ if $\exists M$, that for input (x, y) with $|x| = n$ and $|y| = t(n)$ runs in time $t(n)$ for $\forall n \in \mathbb{N}$, and whose output lies in $\{1, ?\}$ if $x \in L$, and in $\{0, ?\}$ if $x \notin L$. Additionally, M satisfies the condition:

(a) **Uniqueness for $\texttt{C} = \texttt{U}$:** $\Sigma_{y:M(x,y)\in\{0,1\}} 1 = 1$

(b) **Largeness for $\texttt{C} = \texttt{R}$:** $Pr_y[M(x, y) \in \{0, 1\}] \geq \frac{2}{3}$

(c) **Existence for $\texttt{C} = \texttt{N}$:** $\Sigma_{y:M(x,y)\in\{0,1\}} 1 \geq 1$

Note that $\texttt{ZRTIME} = \texttt{ZPTIME}$, and for $\texttt{C} = \texttt{N}, \texttt{R}, \texttt{U}$, $\texttt{ZCTIME}(t) = \texttt{CTIME}(t) \cap \texttt{Co-CTIME}(t)$ follows by a similar argument that shows $\texttt{ZPTIME}(t) = \texttt{RTIME}(t) \cap \texttt{Co-RTIME}(t)$.

Seeds for $\texttt{ZCE}$*:* $\texttt{ZCE}$ has seeds in $\mathcal{C}$ if for every $\texttt{ZCE}$ predicate V, there is a k such that for all x, there is a $|x|^k$-size $\mathcal{C}$ circuit C_x such that $V(x, tt(C_x)) \in \{0, 1\}$.

Hitting-sets for $\texttt{CE}$*:* $\texttt{CE}$ has l-size hitting-sets in $\mathcal{C}$ if for every $\texttt{CE}$ predicate V, $\exists k\ \forall n \in \mathbb{N}$, there is an n^k-size $\mathcal{C}$ circuit C_n such that $tt(C_n)$ when partitioned into l strings $\{str_1, \ldots, str_l\}$ of equal lengths, satisfies $\forall(x : |x| = n \wedge x \in L)\ \exists(i \in [1, l])\ V(x, str_i) = 1$.

Witness: A non-deterministic verifier V has witness in s-size $\mathcal{C}$ circuits, if for every $x \in L$, there is a $s(|x|)$-size $\mathcal{C}$ circuit C_x, such that $V(x, tt(C_x)) = 1$.

Oblivious witness: Let $y_1, \ldots, y_{2^n}$ denote the n-length strings arranged in the lexicographical order. A non-deterministic verifier V has oblivious witness in s-size $\mathcal{C}$ circuits, if $\forall n \in \mathbb{N}$, there is a $s(n)$-size $\mathcal{C}$ circuit C_n, such that $tt(C_n)$ when partitioned into 2^n strings $\{str_1, \ldots, str_{2^n}\}$ of equal lengths, satisfies $\forall(i \in [1, 2^n])\ y_i \in L \implies V(y_i, str_i) = 1$. For i with $y_i \notin L$, str_i is the all 0s string.

Circuit lower bounds: There are two types of lower bounds: (i) $\texttt{NEXP} \not\subset \texttt{SIZE}(poly)$, i.e., an $L \in \texttt{NEXP}$ satisfies $\forall k\ L \notin \texttt{SIZE}(n^k)$; (ii) $\forall k\ \texttt{NE} \not\subset \texttt{SIZE}(n^k)$, i.e., a fix slice of $\texttt{NEXP}$ (in this case $\texttt{NE}$) has L_k for each k, such that $L_k \notin \texttt{SIZE}(n^k)$. $\texttt{NEXP} \not\subset \texttt{SIZE}(poly) \iff \forall k\ \texttt{NE} \not\subset \texttt{SIZE}(n^k)$: Forward direction follows from a simple padding argument, and reverse uses a complete language for $\texttt{NE}$. Same holds for $\texttt{DTIME}$. But for $\texttt{ZNTIME}$, $\texttt{UTIME}$, $\texttt{ZUTIME}$, $\texttt{RTIME}$, $\texttt{ZRTIME}$, $\texttt{BPTIME}$ only the forward direction holds and these two lower bounds are not known to be equivalent (due to the lack of a complete problem). In this paper we only focus on the case (i), but our results can be easily extended to any reasonable lower bound (including (ii)).

Useful properties: We define a generalized version of the natural properties.

▶ **Definition 1** (Useful uniform properties). *A Γ algorithm $\mathcal{A}$ is a Γ-C property if it satisfies the first condition (stated below) on the inputs that are powers of 2 (interpreted as truth-tables). $\mathcal{A}$ is said to be useful against s-size $\mathcal{C}$ circuits if it satisfies the second condition.*

1. a. **Uniqueness for $\mathtt{C} = \mathtt{U}$:** $\forall n \in \mathbb{N}\ \Sigma_{x:|x|=2^n \wedge \mathcal{A}(x)=1} 1 = 1$
   b. **Largeness for $\mathtt{C} = \mathtt{R}$:** $\forall n \in \mathbb{N}\ Pr_{x:|x|=2^n}[\mathcal{A}(x) = 1] \geq \frac{1}{2^n}$
   c. **Existence for $\mathtt{C} = \mathtt{N}$:** $\forall n \in \mathbb{N}\ \Sigma_{x:|x|=2^n \wedge \mathcal{A}(x)=1} 1 \geq 1$
2. **Usefulness:** *for infinitely many* $n \in \mathbb{N}$, $\forall (x : |x| = 2^n)\ \mathcal{A}(x) = 1 \implies ckt_{\mathcal{C}}(x) > s(n)$

Note that, in the case where s is $poly(n)$, the same algorithm $\mathcal{A}$ should be useful against n^k for all k. That is, for each k, there should be infinitely many $n \in \mathbb{N}$, such that $\forall (x : |x| = 2^n)\ \mathcal{A}(x) = 1 \implies ckt_{\mathcal{C}}(x) > n^k$.

▶ **Definition 2** (Useful properties that use advice). *A Γ/a algorithm $\mathcal{A}$ is a Γ/a-C property if it satisfies the first condition of the Definition 1 on an advice sequence $\{a_n\}_{n\in\mathbb{N}}$ that satisfies $\forall n\ |a_n| = a(n)$. $\mathcal{A}$ is said to be useful against s-size $\mathcal{C}$ circuits if it satisfies the second condition of the Definition 1 on the advice sequence $\{a_n\}_{n\in\mathbb{N}}$. For $\mathtt{C} = \mathtt{U}$, based on how $\mathcal{A}$ behaves on the advice sequences other than $\{a_n\}_{n\in\mathbb{N}}$, it is divided into the following categories:*

1. **Γ/a-strong-unique or Γ/a-$u_{=1}$:** $\forall n \in \mathbb{N}\ \forall (a_n : |a_n| \leq a(n))\ \Sigma_{x:|x|=2^n \wedge \mathcal{A}(x)/a_n=1} 1 = 1$
2. **Γ/a-mild-unique or Γ/a-$u_{\leq 1}$:** $\forall n \in \mathbb{N}\ \forall (a_n : |a_n| \leq a(n))\ \Sigma_{x:|x|=2^n \wedge \mathcal{A}(x)/a_n=1} 1 \leq 1$
3. **Γ/a-weak-unique or Γ/a-u_*:** *no restriction*

3 EWL and KLT for UTIME, ZUTIME, and FewTIME

We derive EWL using a specific search to decision reduction for UTIME (Section 3.1). Using this reduction we give the EWL and KLT for UTIME (Section 3.2). We describe similar results for ZUTIME (Section 3.3) and FewTIME (Section 3.4).

3.1 Search to decision reduction for UTIME

For $L \in \mathtt{NP}$ and verifier V for L, there is a standard $\mathtt{P}^{\mathtt{NP}}$ algorithm for the corresponding search problem. This algorithm can be easily made into an algorithm for $L_{ewl(V)}$. So if $\mathtt{P} = \mathtt{NP}$, then $L_{ewl(V)} \in \mathtt{P}$. For $L \in \mathtt{NEXP}$ and verifier V for L, such results are not known, i.e., its not known whether $\mathtt{NEXP} = \mathtt{EXP}$ yields an EXP algorithm for $L_{ewl(V)}$. In [12] it was shown that $L_{ewl(V)} \in \mathtt{EXP}$ if, $\mathtt{NEXP} = \mathtt{AM}$ or NEXP has witness in $\mathtt{SIZE}(poly)$. It has been shown that the later condition is equivalent to $\mathtt{NEXP} = \mathtt{MA}$ (due to [12, 36] and van Melkebeek), and since $\mathtt{MA} \subseteq \mathtt{AM}$, $\mathtt{NEXP} = \mathtt{AM}$ is the weakest collapse we need to put $L_{ewl(V)}$ in EXP.

In this section we show that for $L \in \mathtt{UTIME}(t)$ and unambiguous verifier V for L, $L_{ewl(V)} \in \mathtt{UTIME}(t)$. We also show why it would be difficult to extend this to all ambiguous verifiers.

▶ **Theorem 3.** *For $L \in \mathtt{UTIME}(t)$ and unambiguous verifier V for L, $L_{ewl(V)} \in \mathtt{UTIME}(t)$. Moreover if this statement is true for every non-deterministic verifier (ambiguous and unambiguous), then $\mathtt{ZNTIME}(t) = \mathtt{ZUTIME}(t)$.*

Proof. *Algorithm for $L_{ewl(V)}$* : For input (x, i), guess a certificate y and simulate $V(x, y)$. Accept if V accepts and the i^{th} bit of y is 1, otherwise reject. This algorithm is correct and unambiguous as V is unambiguous. It runs in time $O(t(|x|)) \leq O(t(|x| + |i|))$.

The moreover part : For $L \in \mathtt{ZNTIME}(t)$, let V_1 and V_0 be its $\mathtt{NTIME}(t)$ and $\mathtt{Co\text{-}NTIME}(t)$ verifiers respectively. Consider the $\mathtt{UTIME}(t)$ language $L' = \{0, 1\}^*$. Using V_1 and V_0 we construct a verifier V' for L' with the following property – if the first bit of the certificate is i, V' simulates V_i using the rest of the certificate. Using a $\mathtt{UTIME}(t)$ algorithm $\mathcal{A}$ for $L'_{ewl(V')}$ we give a $\mathtt{UTIME}(t)$ algorithm for L. On input x, simulate $\mathcal{A}$ on $(x, 1)$. If $\mathcal{A}$ accepts then we

know that $x \in L$ because then there is no positive certificate for V' that starts with 0 (or in other words, no positive certificate for V_0). So we accept iff $\mathcal{A}$ accepts. Similarly, there is a $\texttt{UTIME}(t)$ algorithm for $\overline{L}$, and thus $L \in \texttt{ZUTIME}(t)$. ◀

3.2 EWL and KLT for UTIME

Using the search to decision reduction from Theorem 3 we derive EWL for unambiguous verifiers of languages in $\texttt{UTIME}(t)$. Here again we see why it might be difficult to extend this to all ambiguous verifiers. Using the EWL we also get a KLT for UTIME.

▶ **Theorem 4.** *The following statements are true for constants c and k:*

(a) *For time-constructible $t \in 2^{O(n)}$, $\texttt{UTIME}(t) \subseteq \mathcal{C}(n^k)$ implies that all $\texttt{UTIME}(t)$ verifiers have oblivious witness in $\mathcal{C}(n^k)$. Moreover if this statement is true for every non-deterministic verifier (ambiguous and unambiguous) of every $\texttt{UTIME}(t)$ language, then $\texttt{ZNTIME}(t) \subseteq \texttt{DTIME}(2^{n^{k+1}}t)$.*

(b) *If $\texttt{UTIME}(2^{n^c})/a \subseteq \mathcal{C}(n^k)$, then $\texttt{UTIME}(2^{n^c})/a$ has oblivious witness in $\mathcal{C}(n^{ck})$ for all verifiers that are unambiguous given the correct advice.*

(c) $\texttt{UEXP}/a \subseteq \texttt{SIZE}(poly) \implies \texttt{UEXP}/a = \texttt{MA}/a$.

Proof. *Proof of (a):* For $L \in \texttt{UTIME}(t)$, let $x \in L$ be an n-length input, and V be an unambiguous verifier for L whose certificate length is $\leq d \cdot t$ for some constant d. The $\texttt{UTIME}(t)$ algorithm of $L_{ewl(V)}$ from Theorem 3 puts it into $\mathcal{C}(m^k)$ for input size m. The $\mathcal{C}$ circuit for input length $m = (|x| + \log t + \log d) \in O(n)$ is the oblivious witness circuit for n-length inputs.

The moreover part: For $L \in \texttt{ZNTIME}(t)$, construct the same verifier V' for the language $L' = \{0,1\}^*$ as in the proof of Theorem 3. As $L' \in \texttt{UTIME}(t)$, V' will have witness in $\mathcal{C}(n^k)$. Now a $\texttt{DTIME}(2^{n^{k+1}}t)$ algorithm for L is – for n-length input x, go through all the circuits in $\mathcal{C}(n^k \log n)$ one at a time, compute their truth-tables tt, and then compute $V'(x, tt)$. Due to the way V' is constructed, all of its positive certificates have the same first bit. If V accepts on any tt whose first bit is 1, then $x \in L$. Else $x \notin L$.

Proof of (b): It is analogous to the proof of (a).

Proof of (c): Let $L \in \texttt{UEXP}/a$, and V be an unambiguous (given the correct advice) verifier V for L that runs in time $O(2^{n^c})$ for some constant c. From the assumption $\texttt{UEXP}/a \subseteq \texttt{SIZE}(poly)$ and part (b), V has witness in $\texttt{SIZE}(n^{ck})$ for some k.

Using this we first give an $\texttt{EXP}/a$ algorithm for L. On n-length input x, go through all the circuits in $\texttt{SIZE}(n^{ck} \log n)$ one at a time, compute their truth-tables tt, and then compute $V(x, tt)$. Accept if V accepts for any tt, else reject. This is an $\texttt{EXP}/a$ algorithm as simulation of V needs the original advice.

Once we get $\texttt{UEXP}/a = \texttt{EXP}/a$, $\texttt{EXP}/a \subseteq \texttt{SIZE}(poly)$ gives $\texttt{UEXP}/a = \texttt{MA}/a$ [20]. ◀

3.3 EWL and KLT for ZUTIME

We extend the techniques from the previous section to give similar results for ZUTIME. Note that EWL and KLT for ZNTIME are not known. The main difference in the proof of search to decision reduction is: we also show $\overline{L_{ewl(V)}} \in \texttt{UTIME}(t)$ using unambiguous verifiers of L and $\overline{L}$ both. Then, EWL and KLT follow from a similar argument as in the previous section.

▶ **Theorem 5.** *The following statements are true for constants c and k:*

(a) *For $L \in \texttt{ZUTIME}(t)$ and unambiguous verifier V for L, $L_{ewl(V)} \in \texttt{ZUTIME}(t)$. Moreover if this statement is true for all non-deterministic verifiers (ambiguous and unambiguous), then $\texttt{ZNTIME}(t) = \texttt{ZUTIME}(t)$.*

(b) *For time-constructible* $t \in 2^{O(n)}$*, if* $\texttt{ZUTIME}(t) \subseteq \mathcal{C}(n^k)$*, then* $\texttt{ZUTIME}(t)$ *has oblivious witness in* $\mathcal{C}(n^k)$ *for all unambiguous verifiers. Moreover if this statement is true for all non-deterministic verifiers (ambiguous and unambiguous), then* $\texttt{ZNTIME}(t) \subseteq \texttt{DTIME}(2^{n^{k+1}}t)$.

(c) *If* $\texttt{ZUTIME}(2^{n^c}) \subseteq \mathcal{C}(n^k)$*, then* $\texttt{ZUTIME}(2^{n^c})$ *has oblivious witness in* $\mathcal{C}(n^{ck})$ *for all unambiguous verifiers.*

(d) $\texttt{ZUEXP} \subseteq \texttt{SIZE}(poly) \implies \texttt{ZUEXP} = \texttt{MA}$.

Proof. *Proof of (a):* From the above theorem we get that $L_{ewl(V)} \in \texttt{UTIME}(t)$. The part remaining to show is $\overline{L_{ewl(V)}} \in \texttt{UTIME}(t)$. Let V' be an unambiguous verifier for $\overline{L}$. For input (x, i), guess a bit z. If $z = 0$, simulate V' on x and accept if it accepts. If $z = 1$, guess a certificate y and simulate $V(x, y)$. Accept if V accepts and the i^{th} bit of y is 0. This is a $\texttt{UTIME}(t)$ algorithm because for any x only one of the two branches – $z = 0$ and $z = 1$ – accepts, and they both accept unambiguously.

The moreover part: As $\{0,1\}^* \in \texttt{ZUTIME}(t)$, the moreover part's proof is the same as that in Theorem 3.

Proofs of (b), (c) & (d): The proofs are analogous to the proofs of (a), (b), & (c) of the above theorem, respectively. ◀

3.4 EWL and KLT for FewTIME

One variant of $\texttt{UTIME}(t)$ is $\texttt{FewTIME}(t)$. $L \in \texttt{FewTIME}(t)$, if there exists a constant c and a non-deterministic verifier V, such that the number of accepting certificates on any input is bounded by t^c. The search to decision reduction of UTIME doesn't work here, because we don't know the exact number of accepting certificates (and only know an upper bound). We get rid of this problem, by either assuming $\texttt{UE} = \texttt{Co-UE}$ (a clever induction argument), or by using advice (that encodes the total number of accepting certificates for all the 2^n inputs). After the search to decision reduction is obtained, arguments for the EWL and the KLT are similar to the ones used for UTIME.

▶ **Theorem 6.** *The following statements are true if* $\texttt{UE} = \texttt{Co-UE}$*:*

1. $\texttt{FewE} = \texttt{UE} = \texttt{ZUE}$
2. $L \in \texttt{FewE} \implies \forall(\texttt{FewE}$ *verifier* V *for* $L)$ $L_{ewl(V)} \in \texttt{UE}$
3. $\texttt{EWL} : \texttt{UE} \subset \mathcal{C} \implies$ *every* FewE *verifier has oblivious witness in* $\mathcal{C}$
4. $\texttt{KLT} : \texttt{UE} \subset \texttt{SIZE}(poly) \implies \texttt{FewE} \subset \texttt{MA}$

Proof. *Proofs of 1 & 2:* For any $L \in \texttt{FewE}$, let V be a verifier whose number of accepting certificates, and running time, both are bounded by 2^{cn}, for some constant c. For $p \in [1, 2^{cn}]$ we construct a new language $L_p = \{x \mid p \leq \sum_y V(x, y) \leq 2^{cn}\}$. Using induction we prove that $\forall p L_p$ has $2^{2cn}(2^{cn} - p + 1)$ UTIME algorithm. Its easy to check that $L_{2^{cn}}$ has 2^{2cn} UTIME algorithm – guess 2^{cn} distinct accepting certificates. Now assuming that $L_{2^{cn}}, \ldots, L_{p+1}$ satisfy the induction condition we give a $2^{2cn}(2^{cn} - p + 1)$ UTIME algorithm for L_p.

Under the assumption $\texttt{UE} = \texttt{Co-UE}$, L_{p+1} and $\overline{L_{p+1}}$ both have $2^{2cn}(2^{cn} - p)$ UTIME algorithms. On input x, guess a non-deterministic bit z. If $z = 1$, run the UTIME algorithm for L_{p+1} on x, and accept if it accepts (since $L_{p+1} \subseteq L_p$). If $z = 0$, run the UTIME algorithm for $\overline{L_{p+1}}$ on x. If it accepts, then the only way x could be in L_p is by $\sum_y V(x, y) = p$. So guess p distinct accepting certificates of V on x. Accept if V accepts all of them. This is an unambiguous algorithm because only one branch, either $z = 0$ or $z = 1$, leads to acceptance, and both branches are unambiguous. The total time of this algorithm is $1 + 2^{2cn}(2^{cn} - p) + p2^{cn} \leq 2^{2cn}(2^{cn} - p + 1)$ since $p < 2^{cn}$.

Now L, which is essentially L_1, belongs to $\texttt{UTIME}(2^{3cn})$. For $L_{ewl(V)}$, guess p and run the UTIME algorithms for L_p and $\overline{L_{p+1}}$ on x. If both of them accept, then we know that p is the exact number of accepting certificates of V on x. So for input (x, i) of $L_{ewl(V)}$, guess p distinct accepting certificates and output the i^{th} bit of the lexicographically smallest certificate. This is a $\texttt{UTIME}(2^{3cn})$ algorithm.

Proof of 3: From 2, we know that for every $L \in \texttt{FewE}$, and every FewE verifier V for L, $L_{ewl(V)} \in \texttt{UE}$. Thus, $L_{ewl(V)} \in \mathcal{C}$, and V has oblivious witness in $\mathcal{C}$ (similar argument as in Theorem 4).

Proof of 4: This directly from the UTIME KLT (Theorem 4). ◀

▶ **Theorem 7.** *The following statements are true (unconditionally):*

1. $\texttt{FewE}/O(n) = \texttt{UE}/O(n) = \texttt{ZUE}/O(n)$
2. $L \in \texttt{FewE}/O(n) \implies \forall$ (FewE$/O(n)$ *verifier* V *for* L) $L_{ewl(V)} \in \texttt{ZUE}/O(n)$
3. $\texttt{EWL} : \texttt{UE}/O(n) \subset \mathcal{C} \implies$ *every* FewE$/O(n)$ *verifier has oblivious witness in* $\mathcal{C}$
4. $\texttt{KLT} : \texttt{UE}/O(n) \subset \texttt{SIZE}(poly) \implies \texttt{FewE}/O(n) \subset \texttt{MA}/O(n)$

Proof. *Proofs of 1 & 2:* For $L \in \texttt{FewE}/O(n)$, and FewE verifier V for L, we give a $\texttt{ZUE}/O(n)$ algorithm for L. The advice of the ZUE algorithm is – the $a \in O(n)$ original advice used by the $fewe$ algorithm – plus extra $O(n)$ bits to encodes the sum of the total number of accepting certificates for V on all n length inputs, let's call this number p. On any n length input x, guess a set S of p pairs (c, d). Output '?', if $\exists(c, d) \in S : \ V(c, d)/a = 0$. Output 1, if $\exists d : \ (x, d) \in S$. Output 0, if $\forall d : \ (x, d) \notin S$. Its easy to check that exactly one non-deterministic branch outputs in the set $\{0, 1\}$. So this algorithm is $\texttt{ZUE}/O(n)$.

Now we give a $\texttt{ZUE}/O(n)$ algorithm for $L_{ewl(V)}$. For input (x, i), the advice part, and the algorithm part before the output step, are same as that for L. Output '?', if $\exists(c, d) \in S : \ V(c, d)/a = 0$. Output 1, if the i^{th} bit of the lexicographically smallest d such that $(x, d) \in S$ is 1. Output 0, if $\forall d(x, d) \notin S$, or if the i^{th} bit of the lexicographically smallest d such that $(x, d) \in S$ is 0. Its easy to check that this is also a $\texttt{ZUE}/O(n)$ algorithm.

Proof of 3: From 2, we know that for every $L \in \texttt{FewE}/O(n)$, and every FewE verifier V for L, $L_{ewl(V)} \in \texttt{UE}/O(n)$. Thus, $L_{ewl(V)} \in \mathcal{C}$, and V has oblivious witness in $\mathcal{C}$ (similar argument as in Theorem 4).

Proof of 4: This directly from the UTIME KLT (Theorem 4). ◀

4 Unique Properties vs UTIME/ZUTIME Lower Bounds

In this section we establish relationships between different types of unique properties and lower bounds against UTIME and ZUTIME.

In all the connections we use the following connection between UP-U and P-U properties. The proof of the Lemma 8 is along the same lines as the original connection [1, 29, 36]: an useful NP (RP-natutal) property yields an useful P (P-natural) property.

▶ **Lemma 8.** UP$/a$ *property* $\mathcal{U}$ *can be converted into a* P$/a$ *property* $\mathcal{P}$ *such that:*

1. $\mathcal{U}$ *is* UP$/a$-U *property* $\implies$ $\mathcal{P}$ *is* P$/a$-U *property;*
2. *for* $\sqcap = u_{=1}, u_{\leq 1}, u_*$*:* $\mathcal{U}$ *is* UP$/a$-$\sqcap$ *property* $\implies$ $\mathcal{P}$ *is* P$/a$-$\sqcap$ *property;*
3. $\mathcal{U}$ *is useful against* $\mathcal{C} \implies$ $\mathcal{P}$ *is useful against* $\mathcal{C}$*.*

Proof. Let V be the unambiguous verifier corresponding to $\mathcal{U}$'s algorithm. Let c be a constant such that $2^{cn} - 2^n$ is the length of the certificates that V guesses for the inputs of size 2^n. Now we design $\mathcal{P}$ which satisfies the promises of the theorem statement. For m which is not a multiple of c, among all the inputs of length 2^m, $\mathcal{P}$ only accepts the all

0s string. For $m = cn$ for some n, for any input xy where $|x| = 2^n$ and $|y| = 2^{cn} - 2^n$, $\mathcal{P}$ simulates V on (x, y), and accepts if and only if V accepts. For any $n \in \mathbb{N}$, $\mathcal{P}$ uses the same advice for 2^{cn}-size inputs, that $\mathcal{U}$ uses for 2^n-size inputs.

Proofs of 1 & 2: The construction of $\mathcal{P}$ ensures this for the inputs of size 2^m, where m is not a multiple of c. For all the other input sizes this is ensured by the fact that $\mathcal{U}$ is a UP property, and the behavior of $\mathcal{U}$ on different advice strings. For any $n \in \mathbb{N}$, and any advice string, the number of 2^{cn}-size inputs $\mathcal{P}$ accepts, is same as the number of 2^n-size inputs $\mathcal{U}$ accepts.

Proof of 3: If $\mathcal{U}$ is useful against $\mathcal{C}$, then for each k there exists an infinite subset S_k such that for each $n \in S_k$, $\mathcal{U}(x) = 1 \implies ckt_{\mathcal{C}}(x) > n^k$. For any x, let y be the unique certificate such that $V(x, y) = 1$. Since $ckt_{\mathcal{C}}(x) > n^k \implies ckt_{\mathcal{C}}(xy) > n^k \geq (cn)^{k-1}$, for each k, $\mathcal{P}$ is also useful against n^{k-1}-size $\mathcal{C}$ circuits, and hence is useful against $\mathcal{C}$. ◀

Main results of this section can be summarized as follows:

1. **(Section 4.1)** $\exists$ P/$O(\log n)$-strong-unique or P-U property useful against $\mathcal{C} \iff$ ZUE doesn't have witness in $\mathcal{C} \iff$ ZUE $\not\subseteq \mathcal{C}$
2. **(Section 4.2)** $\exists$ P/$O(\log n)$-mild-unique property useful against $\mathcal{C} \iff$ UE doesn't have witness in $\mathcal{C}$
3. **(Section 4.3)** $\exists$ P/$O(\log n)$-weak-unique property useful against $\mathcal{C} \iff$ UE/$O(n)$ doesn't have witness in $\mathcal{C} \iff$ UE/$O(n) \not\subseteq \mathcal{C}$
4. **(Section 4.4)** $\exists$ NP/$O(\log n)$-strong-unique or NP-U property useful against $\mathcal{C} \iff$ ZNE $\not\subseteq \mathcal{C}$
5. **(Section 4.4)** $\exists$ NP/$O(\log n)$-weak-unique property useful against $\mathcal{C} \iff$ NE doesn't have witness in $\mathcal{C} \iff$ NE $\not\subseteq \mathcal{C}$

Note that, as the lower bounds get weaker, the properties become less restrictive (or the constructivity goes higher).

4.1 ZUE & P/$O(\log n)$-$u_{=1}$ (or P-U) properties

▶ **Theorem 9.** *The following statements are equivalent:*

1. ZUE $\not\subseteq \mathcal{C}$
2. ZUE *doesn't have oblivious witness in* $\mathcal{C}$ *(for some unambiguous verifier)*
3. ZUE *doesn't have witness in* $\mathcal{C}$ *(for some unambiguous verifier)*
4. $\exists$ P-U *(or* UP-U*) property useful against* $\mathcal{C}(poly)$
5. $\exists$ P/$O(\log n)$-$u_{=1}$ *(or* UP/$O(\log n)$-$u_{=1}$*) property useful against* $\mathcal{C}$

Proof. (1 $\implies$ 4) Let $L \in$ UE $\cap$ Co-UE $\setminus \mathcal{C}$, and let V_0 and V_1 be $2^{O(n)}$-time unambiguous verifiers for $\overline{L}$ and L, respectively. For any n, L_n can be viewed as a function f_n, where $f_n^{-1}(1) = \{x \in L \mid |x| = n\}$.

Now using V_0 and V_1 we give a UP-U property $\mathcal{U}$ that is useful against $\mathcal{C}$. For any input y of length 2^n, $\mathcal{U}$ goes through all the n-length strings, one by one. If the i^{th} bit of y is 0, it simulates V_0 on the i^{th} n-length string (to verify its inclusion in $\overline{L}$). If the i^{th} bit of y is 1, it simulates V_1 on the i^{th} n-length string (to verify its inclusion in L). $\mathcal{U}$ accepts if and only if it succeeds in all 2^n verifications.

Uniqueness: For $n \in \mathbb{N}$, $\mathcal{U}$ unambiguously accepts the truth table corresponding to the function f_n, and rejects all the other strings. As it runs for $2^{O(n)}$ time on 2^n-length inputs, it is UP-U.

Usefulness: As $L \notin \mathcal{C}$, for each k, there are infinitely many input lengths n, such that f_n doesn't have n^k-size $\mathcal{C}$ circuits. Thus $\mathcal{U}$ is useful against $\mathcal{C}$.

($4 \implies 3$) If 4 is true, then there is a P-unique property $\mathcal{P}$ useful against $\mathcal{C}$. Using $\mathcal{P}$ we construct an unambiguous verifier V for the UE$\cap$Co-UE language $\{0,1\}^*$ such that V doesn't have witness in $\mathcal{C}$.

For any n-length input x, V guesses a string y of length 2^n and accepts if and only if $\mathcal{P}$ accepts y. Since $\mathcal{P}$ is P-unique property useful against $\mathcal{C}$, the unique accepting witnesses of V are not in $\mathcal{C}$.

($3 \implies 2$) This is trivial.

($2 \implies 1$) The contrapositive follows from the ZUTIME EWL (Theorem 5).

($4 \iff 5$) The forward direction is trivial. For the reverse direction, for constant $c \geq 0$ and P/$c \log n$-$u_{=1}$ property $\mathcal{P}$, we convert $\mathcal{P}$ to a P-U property $\mathcal{P}'$.

For m which is not a multiple of $c+1$, among all the inputs of length 2^m, $\mathcal{P}'$ only accepts the all 0s string. For $m = (c+1)n$ for some n, for any 2^m length input $x_1 x_2 \dots x_{2^{cn}}$ where $\forall i\ |x_i| = 2^n$, $\mathcal{P}'$ accepts – if and only if – for each i, $\mathcal{P}$ accepts input x_i with the advice y_i (i^{th} cn-length string in lexicographical order).

Uniqueness: The uniqueness of $\mathcal{P}'$ directly follows from the fact that $\mathcal{P}$ is a strict-unique property.

Usefulness: If $\mathcal{P}$ is useful against $\mathcal{C}$ with advice sequence $\{a_n\}_{n\in\mathbb{N}}$, then for each k there exists an infinite subset S_k such that for each $n \in S_k$, $\mathcal{P}(x, a_n) = 1 \implies ckt_{\mathcal{C}}(x) > n^k$. For any 2^n-length string x and cn length string a_n – let $y = x_1 \dots x_{b_n} \dots x_{2^{cn}}$, where b_n is the lexicographical rank of a_n among all the cn-length strings – such that $\mathcal{P}'(y) = 1$. Since $ckt_{\mathcal{C}}(x) > n^k \implies ckt_{\mathcal{C}}(y) > n^k \geq ((c+1)n)^{k-1}$, for each k, $\mathcal{P}'$ is also useful against n^k-size $\mathcal{C}$ circuits, and hence is useful against $\mathcal{C}$. ◀

4.2 UE & P/$O(\log n)$-$u_{\leq 1}$ properties

We use a fine-grained version of the techniques from [36], to prove the following two theorems. Unfortunately, the "no oblivious witness → no witness" connection of NTIME doesn't go through in the case of UTIME. If we try to establish a "no oblivious witness → P/$\log n$ property" connection, we get a weak-unique property instead of mild-unique property. In the next section we will see that this connection can be established in the presence of advice.

▶ **Theorem 10.** *The following statements are equivalent:*

1. UE *doesn't have witness in* $\mathcal{C}$ *(for some unambiguous verifier)*
2. $\exists$ P/$O(\log n)$-$u_{\leq 1}$ *(or* UP/$\log n$-$u_{\leq 1}$*) property useful against* $\mathcal{C}$

Proof. ($1 \implies 2$) If 1 is true, then there exists $L \in$ UE, and an unambiguous verifier V for L that doesn't have witness in $\mathcal{C}$.

If the inputs are given as advice, and the certificates are given as inputs, then V becomes a P/$O(\log n)$ property $\mathcal{P}$, that is useful against $\mathcal{C}$.

For $\mathcal{P}$ to be a $u_{\leq 1}$ property, it should be unique with respect to the same advise that makes it useful. At this point, all we know is that for every input length and every advise, $\mathcal{P}$ accepts at most one truth-table (since V is unambiguous). The advise that makes $\mathcal{P}$ useful may not be present for all input lengths. For one of these input lengths n where no such advise is present, it is also possible that L_n is empty.

We will be done if we have an UE verifier that doesn't have witness in $\mathcal{C}$, and whose corresponding language is non-empty for all input lengths. Consider the two modifications of V – (i) V_0, that changes its behavior on the all 0s string and always accepts them (unambiguously) – (ii) V_1 that changes its behavior on the all 1s string and always accepts them (unambiguously). The modified languages, and the modified verifiers, are also UE. We show that, at least one of these two modifications doesn't have witness in $\mathcal{C}$. If V_0 has witness

in $\mathcal{C}$, then V's witnesses corresponding to the all 0s strings must be the ones that were not in $\mathcal{C}$, so then V_1 doesn't have witness in $\mathcal{C}$.

($2 \implies 1$) If 3 is true, then there exists a constant c and a $\mathtt{P}/c\log n\text{-}u_{\leq 1}$ property $\mathcal{P}$ that is useful against $\mathcal{C}$. Define $L = \{x \mid \frac{|x|}{c} \in \mathbb{N} \wedge \exists y\ \mathcal{P}(y)/x = 1\}$. Let V be the verifier for L – that rejects any input whose length is not a multiple of c – on any other input x, it guesses a string y of length $2^{\frac{|x|}{c}}$, and simulates $\mathcal{P}$ on y using x as advice. V is a $\mathtt{UE}$ verifier since $\mathcal{P}$ is a mild-unique property. Clearly V doesn't have witness in $\mathcal{C}$ since $\mathcal{P}$ is useful against $\mathcal{C}$. ◀

▶ **Theorem 11.** $\mathtt{UE}/a \not\subseteq \mathcal{C} \iff \mathtt{UE}/a$ *doesn't have oblivious witness in* $\mathcal{C}$ *(for some verifier that is unambiguous given the correct advice)*

Proof. ($\neg 1 \implies \neg 2$) This follows from the $\mathtt{UTIME\ EWL}$ (Theorem 4).

($\neg 2 \implies \neg 1$) Assume that $\mathtt{UE}/a$ has oblivious witness (for all verifiers that are unambiguous given the correct advice) in $\mathcal{C}$. Let L be a $\mathtt{UE}/a$ language and V be an unambiguous verifier for L that has oblivious witness in n^k-size $\mathcal{C}$ circuits. Now we show that $L \in \mathcal{C}$.

Using V we construct an unambiguous verifier V' for the $\mathtt{UEXP}/a$ language $\{0,1\}^*$, such that for $n \in \mathbb{N}$, an oblivious witness circuit of V' for n-length inputs computes L_n.

For any input $x \in L$ (this can be verified by brute forcing through all the n^{k+1}-size circuits), $V'(x,y)/a = 1$ only when y is the all 1s string. For any input $x \notin L$, $V'(x,y)/a = 1$ only when y is the all 0s string. Since $\mathtt{UE}/a$ has oblivious witness in $\mathcal{C}$, by a simple padding argument $\mathtt{UEXP}/a$ too has oblivious witness in $\mathcal{C}$). Let $\{C_n\}_{n\in\mathbb{N}}$ be the $\mathcal{C}$ circuit family encoding the oblivious witnesses of V'. Then, the $\mathcal{C}$ circuit family defined by $D_n(x) = C_n(x,1)$, encodes the language L (since the first bit of the unique accepting certificate of V' for x is 1 – if and only if – $x \in L$). ◀

4.3 $\mathtt{UE}/O(n)$ & $\mathtt{P}/O(\log n)\text{-}u_*$ properties

Arguments from Section 4.1 when extended to the advice setting, yield the following.

▶ **Theorem 12.** *The following statements are equivalent for any constant* $k \geq 1$*:*

1. $\mathtt{UE}/O(n^k) \not\subseteq \mathcal{C}$
2. $\exists\ \mathtt{P}/O(\log^k n)\text{-}u_*$ *(or* $\mathtt{UP}/O(\log^k n)\text{-}u_*$*) property useful against* $\mathcal{C}$
3. $\mathtt{UE}/O(n^k)$ *doesn't have witness in* $\mathcal{C}$ *(for some verifier that is unambiguous given the correct advice)*
4. $\mathtt{UE}/O(n^k)$ *doesn't have oblivious witness in* $\mathcal{C}$ *(for some verifier that is unambiguous given the correct advice)*

Proof. ($1 \implies 2$) Let $L \in \mathtt{UE}/cn^k \setminus \mathcal{C}$ for some constant c, and let V be $2^{O(n)}$-time unambiguous verifier for L. For any n, L_n can be viewed as a function f_n, where $f_n^{-1}(1) = \{x \in L \mid |x| = n\}$.

Now using V we give a $\mathtt{UP}/(c\log^k m + \log m)\text{-}u_*$ property $\mathcal{U}$ that is useful against $\mathcal{C}$. For any input y of length $m = 2^n$, $\mathcal{U}$ goes through all the n-length strings, one by one. If the i^{th} bit of y is 0, it does nothing. If the i^{th} bit of y is 1, it simulates V on the i^{th} n-length string (to verify its inclusion in L). The first $c\log^k(2^n) = cn^k$ bits of advice is the advice required for the simulation of V. The last $\log(2^n) = n$ bits of advise encodes the number of n-length inputs that V accepts. $\mathcal{U}$ accepts if and only if it succeeds in all 2^n verifications and the hamming weight of y is equal to the number encoded by the last n bits of advise.

Weak uniqueness: For each $n \in \mathbb{N}$, $\mathcal{U}$ unambiguously accepts the truth table corresponding to the function f_n, and rejects all the other strings. As it runs for $2^{O(n)}$ time for 2^n-length inputs, it is $\mathtt{UP}/(cn^k + n)\text{-}u_*$.

Usefulness: As $L \notin \mathcal{C}$, for each l, there are infinitely many input lengths n, such that f_n doesn't have n^l-size $\mathcal{C}$ circuits. Thus $\mathcal{U}$ is useful against $\mathcal{C}$.

(2 $\implies$ 3) Let c be a constant and $\mathcal{P}$ be a $\mathtt{P}/c\log^k n$-u_* property useful against $\mathcal{C}$. We construct an unambiguous verifier V for the $\mathtt{UE}/c\log^k n$ language $\{0,1\}^*$, that doesn't have witness in $\mathcal{C}$. For any n-length input x, guess a 2^n-length string y and simulate $\mathcal{P}$ on y, and accept if an only if $\mathcal{P}$ accepts.

Since $\mathcal{P}$ is useful against $\mathcal{C}$, V doesn't have witness in $\mathcal{C}$. As $\mathcal{P}$ is unique, V is $\mathtt{UE}/cn^k$.

(3 $\implies$ 4) This is trivial.

(4 $\implies$ 1) This follows from the `UTIME EWL` (Theorem 4). ◀

4.4 `ZNE` (`NE`) & `NP`/$O(\log n)$-$u_{=1}$ (`NP`/$O(\log n)$-u_*) properties

In [29] it was conjectured, "$\mathtt{ZNE} \not\subset \mathcal{C} \iff \exists$ `P-N` (or `NP-N`) property useful against $\mathcal{C}$" while only forward direction was proved. We use a fine-grained version of the proof to establish the equivalence in the case of unique properties (first part, Theorem 13). So if this conjecture is true, then any `NP-N` property has an equivalent `NP-U` property. We show a slightly weaker result where this equivalence holds with $O(\log n)$ advice (second part, Theorem 13).

▶ **Theorem 13.** *The following statements are true:*

1. $\mathtt{ZNE} \not\subseteq \mathcal{C} \iff \exists$ $\mathtt{NP}/O(\log n)$-$u_{=1}$ *(or* `NP-U`*) property useful against* $\mathcal{C}$
2. $\mathtt{NE} \not\subseteq \mathcal{C} \iff \exists$ $\mathtt{NP}/O(\log n)$-u_* *(or* $\mathtt{NP}/O(\log n)$`-N`*, or* $\mathtt{P}/O(\log n)$`-N`*) property useful against* $\mathcal{C} \iff$ `NE` *doesn't have witness in* $\mathcal{C} \iff$ `NE` *doesn't have oblivious witness in* $\mathcal{C}$

Proof. *Equivalence of useful* `NP-U` *&* $\mathtt{NP}/O(\log n)$-$u_{=1}$ *properties:* The forward direction is trivial. For the reverse direction, for constant $c \geq 0$ and $\mathtt{NP}/c\log n$-$u_{=1}$ property $\mathcal{P}$, we convert $\mathcal{P}$ to an `NP-U` property $\mathcal{P}'$.

For m which is not a multiple of $c+1$, among all the inputs of length 2^m, $\mathcal{P}'$ only accepts the all 0s string. For $m = (c+1)n$ for some n, for any 2^m length input $x_1x_2\ldots x_{2^{cn}}$ where $\forall i\ |x_i| = 2^n$, $\mathcal{P}'$ accepts – if and only if – for each i, $\mathcal{P}$ accepts input x_i with the advice y_i (i^{th} cn-length string in lexicographical order).

Uniqueness: The uniqueness of $\mathcal{P}'$ directly follows from the fact that $\mathcal{P}$ is a strict-unique property.

Usefulness: If $\mathcal{P}$ is useful against $\mathcal{C}$ with advice sequence $\{a_n\}_{n\in\mathbb{N}}$, then for each k there exists an infinite subset S_k such that for each $n \in S_k$, $\mathcal{P}(x, a_n) = 1 \implies ckt_{\mathcal{C}}(x) > n^k$. For any 2^n-length string x and cn length string a_n – let $y = x_1\ldots x_{b_n}\ldots x_{2^{cn}}$, where b_n is the lexicographical rank of a_n among all the cn-length strings – such that $\mathcal{P}'(y) = 1$. Since $ckt_{\mathcal{C}}(x) > n^k \implies ckt_{\mathcal{C}}(y) > n^k \geq ((c+1)n)^{k-1}$, for each k, $\mathcal{P}'$ is also useful against n^k-size $\mathcal{C}$ circuits, and hence is useful against $\mathcal{C}$.

Proof of 1:

($\implies$) Let $L \in \mathtt{NE} \cap \mathtt{Co\text{-}NE} \setminus \mathcal{C}$, and let V_0 and V_1 be $2^{O(n)}$-time non-deterministic verifiers for $\overline{L}$ and L, respectively. For any n, L_n can be viewed as a function f_n, where $f_n^{-1}(1) = \{x \in L \mid |x| = n\}$.

Now using V_0 and V_1 we give an `NP-U` property $\mathcal{U}$ that is useful against $\mathcal{C}$. For any input y of length 2^n, $\mathcal{U}$ goes through all the n-length strings, one by one. If the i^{th} bit of y is 0, it simulates V_0 on the i^{th} n-length string (to verify its inclusion in $\overline{L}$). If the i^{th} bit of y is 1, it simulates V_1 on the i^{th} n-length string (to verify its inclusion in L). $\mathcal{U}$ accepts if and only if it succeeds in all 2^n verifications.

Uniqueness: For $n \in \mathbb{N}$, $\mathcal{U}$ accepts the truth table corresponding to the function f_n, and rejects all the other strings. As it runs for $2^{O(n)}$ time on 2^n-length inputs, it is `NP-U`.

Usefulness: As $L \notin \mathcal{C}$, for each k, there are infinitely many input lengths n, such that f_n doesn't have n^k-size $\mathcal{C}$ circuits. Thus $\mathcal{U}$ is useful against $\mathcal{C}$.

($\Longleftarrow$) Let $\mathcal{U}$ be an NP-unique property useful against $\mathcal{C}$. Using $\mathcal{U}$ we construct a language in $\mathtt{ZNE} \setminus \mathcal{C}$ by designing verifier V_0 for $\overline{L}$, and verifier V_1 for L.

For any n-length input x whose lexicograhical rank (among all n-bit strings) is j, V_i (for $i \in \{0,1\}$) guesses a string y of length 2^n and simulates $\mathcal{U}$ on it. It accepts if and only if $\mathcal{U}$ accepts y, and y's j^{th}-bit is equal to i.

Since $\mathcal{U}$ is NP-unique property useful against $\mathcal{C}$, for each n the 2^n-length string y it accepts is unique. Thus the languages corresponding to V_0 and V_1 are compliments of each other, and for each k there are infinitely many values of n where L_n (which is represented by the string y), doesn't have n^k-size $\mathcal{C}$ circuits.

Proof of 2:

We will only show that $\mathtt{NE} \not\subset \mathcal{C}$ implies the existence of an $\mathtt{NP}/\log n$-u_* property that is useful against $\mathcal{C}$. All the other implications follow from [36, 29] since an $\mathtt{NP}/\log n$-u_* property is only a special case of an $\mathtt{NP}/\log n$-N property.

Let $L \in \mathtt{NE} \setminus \mathcal{C}$, and let V be $2^{O(n)}$-time non-deterministic verifier for L, respectively. For any n, L_n can be viewed as a function f_n, where $f_n^{-1}(1) = \{x \in L \mid |x| = n\}$.

Now using V we give an $\mathtt{NP}/\log n$-u_* property $\mathcal{U}$ that is useful against $\mathcal{C}$. For any input y of length 2^n, $\mathcal{U}$ goes through all the n-length strings, one by one. If the i^{th} bit of y is 0, it does nothing. If the i^{th} bit of y is 1, it simulates V on the i^{th} n-length string (to verify its inclusion in L). $\mathcal{U}$ accepts if and only if it succeeds in all 2^n verifications and the hamming weight of y (the size of L_n) is equal to the number encoded by the advice.

The proof of uniqueness and usefulness is similar to that of the first case. ◀

5 Mild Derandomization from Uniform/Non-uniform Lower Bounds

In this section we extend the "lower-bounds to derandomization" frame-work of [36] to get unified results for the three one-sided error classes – NEXP, REXP and UEXP – and their zero-error versions – ZNEXP, ZREXP, and ZUEXP. We use the following "hardness to randomness" connection in our derandomization results.

▶ **Theorem 14** ([34]). *There is a universal constant g and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for all s and Y satisfying $CC(Y) \geq s^g$, and for all circuits C of size s, $|Pr_{x \in \{0,1\}^{g \log |Y|}}[C(G(Y,x) = 1)) - Pr_{x \in \{0,1\}^s}[C(x) = 1]]| < 1/s$. Furthermore, G is computable in poly$(|Y|)$ time.*

As a first step towards unification, we generalize the connection between "derandomization of ZRE (RE) through seeds (hitting-sets)" and P-R (P/$\log n$-R) properties [36], to NE, UE, ZNE,ZUE. Note that, for NE, UE, ZUE one can also use the "lower bounds to easy-witness" connections.

▶ **Lemma 15.** *The following statements are true for* $\mathtt{C} = \mathtt{N}, \mathtt{R}, \mathtt{U}$*:*

1. ZCE *doesn't have seeds in* $\mathcal{C} \iff \exists$P-C *property useful against* $\mathcal{C}$
2. CE *doesn't have* 2^n*-size hitting sets in* $\mathcal{C} \implies \exists$P/$\log n$-C *property useful against* $\mathcal{C}$

Proof. *Proof of 1:* ($\Longrightarrow$) Let V be a ZCE predicate for $L \in$ ZCE whose length of certificates is 2^{cn} for some constant c. For each k, let S_k be an infinite size set whose each element n satisfies – $\exists x$ such that $|x| = n$ and $\forall y\ V(x,y) \in \{0,1\} \implies ckt_{\mathcal{C}}(y) \geq n^k$. Using V we construct a P-C property $\mathcal{P}$ useful against $\mathcal{C}$. For constant d, view any $2^{(c+d)n}$-length input as a collection of 2^{dn} certificates.

For $\mathtt{C} = \mathtt{N}, \mathtt{U}$: Let d be 1. $\mathcal{P}$ accepts if and only if, for each $i \in [1, 2^n]$, V outputs 0/1 on the i^{th} n-length input when given the i^{th} certificate from the collection. Clearly, $\mathcal{P}$

is a P-C property. For each k, it is useful against n^k-size $\mathcal{C}$ circuits, because for inputs $n \in S_{k+1}$, it only accepts strings of length $2^{(c+1)n}$ that have a substring y of length 2^{cn} with $ckt_{\mathcal{C}}(y) \geq n^{k+1} \geq ((c+1)n)^k$ (because corresponding to each n-length input, a 0/1 outputting certificate is present as a substring).

For $\mathtt{C} = \mathtt{R} : \mathcal{P}$ accepts if and only if, for each $i \in [1, 2^n]$, V outputs 0/1 on the i^{th} n-length input, for atleast one of the 2^{dn} certificates. For carefully chosen d, $\mathcal{P}$ is a P-C property because $(1 - (\frac{1}{3})^{2^{dn}})^{2^n} \geq \frac{1}{2^{(c+d)n}}$. Usefulness of $\mathcal{P}$ follows from a similar argument as above.

($\Longleftarrow$) Let $\mathcal{P}$ be a P-C property useful against $\mathcal{C}$. For each k, let S_k be the infinite set of inputs where $\mathcal{P}$ only accepts strings str with $ckt_{\mathcal{C}}(str) \geq n^k$. Using $\mathcal{P}$ we construct a ZCE predicate V for Σ^* that doesn't have seeds in $\mathcal{C}$. For n-length input x, where $2^n \in S$, guess a string of length $2^{(d+1)n}$ for some constant d. View it as a collection of 2^{dn} strings of length 2^n each.

For $\mathtt{C} = \mathtt{N}, \mathtt{U} : V$ outputs 1 if and only if $\mathcal{P}$ accepts all of the strings in the collection. Clearly, V is a ZCE predicate. For each k, it doesn't have seeds in n^k-size $\mathcal{C}$ circuits, because for any n with $2^n \in S_k$, any $2^{(d+1)n}$-length certificate it accepts, contains a 2^n-length substring y with $ckt_{\mathcal{C}}(y) \geq n^k$.

For $\mathtt{C} = \mathtt{R} : V$ outputs 1 if and only if $\mathcal{P}$ accepts at least one of the strings in the collection. For carefully chosen d, V is a ZCE predicate because $(1 - \frac{1}{2^n})^{2^{dn}} \leq \frac{1}{3}$. From a similar argument as above it follows that V doesn't have seeds in $\mathcal{C}$.

Proof of 2: Let V be a CE predicate for $L \in$ CE whose length of certificates is 2^{cn} for some constant c. For each k, let S_k be an infinite set whose each element n satisfies – V doesn't have 2^n-size hitting sets in n^k-size $\mathcal{C}$ circuits. Using V we construct a CP/$\log n$-C property $\mathcal{P}$ (which we know has an equivalent P/$\log n$-C property) useful against $\mathcal{C}$. View any $2^{(c+1)n}$-length input as a collection of 2^n certificates (or in other words, view it as a 2^n-size hitting-set). Use the $(c+1)n$ length advice to encode the number of n-length inputs L has. Call it l_n.

For $\mathtt{C} = \mathtt{N}, \mathtt{U}$: Guess a 2^n length string str of hamming weigth l_n. $\mathcal{P}$ accepts if and only if, for each $i \in [1, 2^n]$ where the i^{th} bit of str is 1, V accepts the i^{th} n-length input when given the i^{th} certificate from the collection. For all the other values of i, the i^{th} certificate from the collection should be the all 0s string. Clearly, $\mathcal{P}$ is a CP/$\log n$-C property. For each k, it is useful against n^k-size $\mathcal{C}$ circuits as it only accepts strings of length $2^{(c+1)n}$ that are 2^n-size hitting sets for V.

For $\mathtt{C} = \mathtt{R} : \mathcal{P}$ accepts if and only if, for l_n-many $i \in [1, 2^n]$, V accepts the i^{th} n-length input, for atleast one of the 2^n certificates. $\mathcal{P}$ is a P/$\log n$-C property because $(1 - (\frac{1}{3})^{2^n})^{2^n} \geq \frac{1}{2^{(c+1)n}}$. Usefulness of $\mathcal{P}$ follows from a similar argument as above. ◀

The original connection, of which Lemma 15 can be viewed as a generalization, was used to show: REXP $\not\subset$ SIZE($poly$) or REXP $\neq$ EXP $\implies$ $\forall \epsilon > 0$ BPP $\subset$ io-ZPTIME(2^{n^ϵ})/n^ϵ. The lower bounds implied the existence of useful properties, which were used to derandomize BPP. Using Lemma 15, we get a variety of properties from a variety of lower bounds, and thus get a variety of derandomization results.

▶ **Theorem 16.** *The following statements are true for* $\mathtt{C} = \mathtt{N}, \mathtt{R}, \mathtt{U}$*:*

1. CEXP $\not\subset$ SIZE($poly$) $\implies$ $\forall \epsilon > 0$ BPP $\subset$ io-ZCTIME(2^{n^ϵ})/n^ϵ
2. CEXP $\neq$ EXP $\implies$ $\forall \epsilon > 0$ BPP $\subset$ io-ZCTIME(2^{n^ϵ})/n^ϵ
3. ZCEXP $\not\subset$ SIZE($poly$) $\implies$ $\forall \epsilon > 0$ BPP $\subset$ io-ZCTIME(2^{n^ϵ})
4. ZCEXP $\neq$ EXP $\implies$ $\forall \epsilon > 0$ BPP $\subset$ io-ZCTIME(2^{n^ϵ})

Proof. *Proof of 4:* Let's assume that ZCEXP $\neq$ EXP. Then ZCE can't have seeds in SIZE($poly$), because brute-forcing through the seeds will prove ZCEXP = EXP. Thus, there exists a P-C

property $\mathcal{P}$ useful against $\mathtt{SIZE}(poly)$ (from the above Lemma). For each c, let S_c be the infinite set of input lengths where $\mathcal{P}$ only accept strings str satisfying $ckt(str) \geq n^c$.

For $k, \epsilon > 0$ and $L \in \mathtt{BPTIME}(n^k)$, set $c = gk/\epsilon$ (where g is the constant from Theorem 14). We give a $\mathtt{ZCTIME}(2^{n^\epsilon})$ algorithm for L that works for any input length n with $2^n \in S_c$. For n-length input x of L, let C_x be the circuit corresponding to the BP computation of L (on x).

For $\mathtt{C} = \mathtt{N}, \mathtt{U}$: Non-deterministically guess a string Y of length $m = 2^{\Theta(n^\epsilon)}$. Reject if $\mathcal{P}(Y) = 0$. Else we know that $ckt(Y) \geq (n^\epsilon)^c = n^{gk}$. Feed Y to G (from Theorem 14) as the first input, and brute-force through the second input to compute the acceptance probability of the circuit C_x in time $2^{O(n^\epsilon)}$ (within $1/s = 1/n^k$ approximation). Finally use this value to output accordingly.

For $\mathtt{C} = \mathtt{R}$: Instead of one, non-deterministically guess a collection of strings $\{Y_1, \ldots, Y_c\}$ for some constant c. Reject if $\forall i\ \mathcal{P}(Y_i) = 0$. Else proceed with any Y_i with $\mathcal{P}(Y_i) = 1$, and do the same as above. For large enough c, this is a ZRTIME algorithm.

Proof of 2: It's analogous to the proof of 3, except that the properties we get are $\mathtt{P}/\log n$ and not P. The $\log n$-bit advice for this property is precisely the n^ϵ-bit advice for the $\mathtt{ZCTIME}(2^{n^\epsilon})$ algorithm we get.

(4 $\implies$ 3) We prove the contrapositive. Assume that $\exists \epsilon > 0$ such that $\mathtt{BPP} \not\subset$ io-$\mathtt{ZCTIME}(2^{n^\epsilon})$. From the Equation (2) we get $\mathtt{EXP} \subset \mathtt{SIZE}(poly)$, and from 4 we get $\mathtt{ZCEXP} = \mathtt{EXP}$. Thus, we get $\mathtt{ZCEXP} \subset \mathtt{SIZE}(poly)$.

(2 $\implies$ 1) The proof is analogous to the above proof. ◀

In [36] they got: $\exists c \geq 1\ \forall \epsilon > 0\ \mathtt{RP} \subseteq \mathtt{RE} \cap \mathtt{BPP} \subset io\text{-}\mathtt{ZPTIME}(2^{n^\epsilon})/n^c$. We get:

▶ **Corollary 17.** *For* $\mathtt{C} = \mathtt{N}, \mathtt{R}, \mathtt{U}$: $\exists c \geq 1\ \forall \epsilon > 0\ \mathtt{CE} \cap \mathtt{BPP} \subset io\text{-}\mathtt{ZCTIME}(2^{n^\epsilon})/n^c$

We also get the following generalized "uniform-separation to derandomization" results. The idea is: we break the separations into two ($\Gamma \neq \mathtt{EXP}$, and $\Gamma \neq \mathtt{MA}$ or $\Gamma \neq \mathtt{BPP}$) and then apply, Theorem 16 on the first separation, and EXP KLT [4] or Equation (2) on the second.

▶ **Theorem 18.** *For* $\mathtt{C} = \mathtt{N}, \mathtt{R}, \mathtt{U}$:

1. $\mathtt{ZCEXP} \neq \mathtt{MA} \implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}\mathtt{ZCTIME}(2^{n^\epsilon})$
2. $\mathtt{ZCEXP} \neq \mathtt{BPP} \implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}Heur\text{-}\mathtt{ZCTIME}(2^{n^\epsilon})$
3. $\mathtt{CEXP} \neq \mathtt{MA} \implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}\mathtt{ZCTIME}(2^{n^\epsilon})/n^\epsilon$
4. $\mathtt{CEXP} \neq \mathtt{BPP} \implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}Heur\text{-}\mathtt{ZCTIME}(2^{n^\epsilon})/n^\epsilon$

Proof.

$$\begin{aligned}
\textit{Proof of 1} : \mathtt{ZCEXP} \neq \mathtt{MA} &\implies \mathtt{ZCEXP} \neq \mathtt{EXP} \text{ or } \mathtt{EXP} \neq \mathtt{MA} \\
&\implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}\mathtt{ZCTIME}(2^{n^\epsilon}) \text{ or } \mathtt{EXP} \not\subset \mathtt{SIZE}(poly) \\
&\implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}\mathtt{ZCTIME}(2^{n^\epsilon}) \text{ or } \mathtt{ZCEXP} \not\subset \mathtt{SIZE}(poly) \\
&\implies \forall \epsilon > 0\ \mathtt{BPP} \subset io\text{-}\mathtt{ZCTIME}(2^{n^\epsilon})
\end{aligned}$$

where the second implication follows from the above theorem and EXP KLT, and the last implication again uses the above theorem.

Proof of 2: Its the same as above, except – instead of $\mathtt{EXP} \neq \mathtt{MA}$ we get $\mathtt{EXP} \neq \mathtt{BPP}$ – and the result follows from the Equation (2).

Proofs of 3 & 4: They are analogus to the proofs of 1 and 2, respectively. ◀

6 UEXP Lower Bounds from Fast Unambiguous Algorithms

In this section we show how to get lower bounds from fast algorithms: from half-sub-exponential algorithms combined with the UTIME KLT (Appendix A); by a generalization of the "tight reductions to lower-bounds" connection of [38] combined with the UTIME EWL (Section 6.1); by a generalization of the "learning to lower bounds" connection of [8] combined with the UTIME KLT (Section 6.2). Finally, in Section 6.3 we show some generalizations of lower bound frameworks. We use the following 'hierarchy theorem' and 'tight reductions to SAT' in our proofs.

▶ **Theorem 19** (Heirarchy for UTIME [10]). *For any time bound t such that $n \leq t \leq 2^n$, there is a constant $\epsilon > 0$ and an advice bound $a \in O(\log(t)\log(\log(t)))$ such that* $\mathtt{UTIME}(t)/a \not\subseteq \mathtt{UTIME}(t^\epsilon)/(a+1)$.

▶ **Theorem 20** (Efficient local reductions [17, 33, 9]). *Every language $L \in \mathtt{UTIME}(2^n)$ can be reduced to* 3-USAT *(uniquely satisfiable* 3-SAT*) instances of $2^n n^c$-size, for some constant c. Moreover, given an instance of L there is an n^c-size $\mathcal{C}$ (*P*-uniform) circuit that, on an integer $i \in [2^n n^c]$ in binary as input, outputs the i^{th} clause of the resulting* 3-USAT *formula.*

6.1 Lower bounds from UTIME EWL

In [38] they showed: any super-polynomial savings in designing non-deterministic algorithms for TAUT imply NEXP $\not\subseteq$ SIZE($poly$). We extend this to: faster unambiguous algorithms for, TAUT and canonization, imply UEXP lower bounds. We first formally define canonization.

Canonization : A subset S of circuits is called $\mathtt{CAN}_{(s,\mathcal{C},p)}$, if for any s-size $\mathcal{C}$ circuit C, there exists a unique circuit $C' \in S$ with $tt(C) = tt(C')$, and $|C'| \leq p(ckt_{\mathcal{C}}(tt(C')))$. $\mathtt{CAN}_{(s,\mathcal{C},p)} \in \Gamma/a$ means there is a Γ/a algorithm that decides $\mathtt{CAN}_{(s,\mathcal{C},p)}$.

$\mathtt{TAUT}_{(s,\mathcal{C})}$ ($\mathtt{SAT}_{(s,\mathcal{C})}$) denotes the TAUT (SAT) for s-size $\mathcal{C}$ circuits. In these definitions we omit, the parameter s when it is $poly(n)$, and the circuit class when $\mathcal{C}$ = Boolean.

Main idea: In [38] they combined the witness circuit with the reduction circuit (Theorem 20), and used a faster TAUT algorithm. We do the same except: we use UTIME EWL, and we use canonization to unambiguously guess the witness circuit. The UTIME EWL works for any t, and thus also yields a corollary for USUBEXP (which is the class $\cap_{\epsilon>0}\mathtt{UTIME}(2^{\epsilon n})$).

▶ **Theorem 21.** *For $\delta \leq 1$, let a, c and ϵ be the parameters of Theorems 19 and 20 for the time bound $t = 2^{\delta n}$. Then for constant k and function $p(n) \geq n$,* $\mathtt{UTIME}(2^{\delta n})/a \not\subset \mathcal{C}(n^k)$ *if*

1. $\mathtt{TAUT}_{(p(n^{k+1})n+n^c,\mathcal{C})} \in \mathtt{UTIME}(2^{\epsilon n})$ *and* $\mathtt{CAN}_{(n^{k+1},\mathcal{C},p)} \in \mathtt{UTIME}(2^{\epsilon n})/1$*;*
2. $\mathtt{TAUT}_{(p(n^{k+1})n+n^c,\mathcal{C})} \in \mathtt{UTIME}(2^{\epsilon n})/1$ *and* $\mathtt{CAN}_{(n^{k+1},\mathcal{C},p)} \in \mathtt{UTIME}(2^{\epsilon n})$.

Proof. For $L \in \mathtt{UTIME}(2^{\delta n})/a$ and input x, let F_x be the $2^n n^c$-size 3-SAT formula we get by reducing from x (Theorem 20). Also, there is an n^c-size (P-uniform) $\mathcal{C}$ circuit D with $n + c\log n$ input wires, that outputs the i^{th} clause of F when given the input $i \in [1, 2^n n^c]$. Using the assumptions (1 or 2), we will contradict the UTIME hierarchy (Theorem 19) by designing a $\mathtt{UTIME}(2^{\epsilon n})/(a+1)$ algorithm for L.

Let V be the verifier for L that first reduces input x to the 3-SAT formula F_x, and then non-deterministically guesses a satisfying assignment for F_x. From UTIME EWL (Theorem 4) and the assumption $\mathtt{UTIME}(2^{\delta n})/a \subset \mathcal{C}(n^k)$ we know that V has witness circuits in $\mathcal{C}(n^k)$. Let E be a witness circuit of this verifier for the input length $|x| = n$. Combining D and E we construct a circuit C that satisfies: "C is a tautology $\iff$ $x \in L$".

Construction of C: On input i, the output of D is $3n + 3c\log n + 3$ bits. The first $3n + 3c\log n$ bits are the three variables of the i^{th} clause of F. Plug these output bits to

three separate copies of E. The last three bits indicate whether the corresponding literals are positive or negative. Use these three bits and the three output bits from the three copies of E to compute the value of the i^{th} clause (based on the assignment encoded by $tt(E)$).

Contradicting the first assumption: Non-deterministically guess a $p(n^{k+1})$-size $\mathcal{C}$ circuit E. Simulate the $\mathtt{CAN}_{(n^{k+1},\mathcal{C},p)}$ algorithm on E. This requires $\mathtt{UTIME}(2^{\epsilon n})/1$. Reject if the answer is negative. Continue if its positive, and construct C as described above. $|C| \leq p(n^{k+1})n+n^c$. Note that, for any truth-table only one non-deterministic branch will lead to a non-rejecting path. Now simulate the $\mathtt{TAUT}_{(p(n^{k+1})n+n^c,\mathcal{C})}$ algorithm on C. This requires $\mathtt{UTIME}(2^{\epsilon n})$. Note that, C is accepted if and only if, $x \in L$, and $tt(E)$ is the unique witness of V. This whole process requires the advice used in the $\mathtt{UTIME}(2^{\delta n})/a$ algorithm for L. So we get a $\mathtt{UTIME}(2^{\epsilon n})/(a+1)$ algorithm.

Contradicting the second assumption: The algorithm is exactly the same, expect that the extra 1 bit of advice is used at an later stage of the algorithm. ◀

▶ **Corollary 22.** *For $a \in \omega(n \log n)$, $b \in \{0,1\}$ and polynomial $p(n) \geq n$, $\mathtt{USUBEXP}/a \not\subset \mathcal{C}$ if $\mathtt{TAUT}_{\mathcal{C}} \in \mathtt{USUBEXP}/b$ and $\mathtt{CAN}_{(\mathcal{C},p)} \in \mathtt{USUBEXP}/(1-b)$.*

6.2 Lower bound from fast learning algorithms

The two commonly studied learning models are: the Angluin's exact learning model [2], and the Valiant's PAC model [35]. Fast learning algorithm in these models have been shown to yield lower bound [8, 30, 11, 21, 29]. We now formally define UTIME exact learning.

Exact UTIME *learning with membership and equivalence queries:* Let s be the size of the target concept C (the circuit to be learned). A $\mathtt{UTIME}(t)$ algorithm is called $\mathtt{Learn}_{(s,C,p)}$, if for any s-size $\mathcal{C}$ circuit C, it outputs a circuit C' of size at most $p(s)$ in time at most $t(s)$ with $tt(C) = tt(C')$, on exactly one of its non-deterministic branches, and rejects all the other branches. The algorithm is allowed to make "membership" and "equivalence" queries. A membership query is: "What is the value of $C(x)$?". An equivalence query is: "Is the current hypothesis (H) equal to C?". On any positive equivalence query, it halts and outputs the current hypothesis. On any negative query, it gets x from the oracle, such that $H(x) \neq C(x)$. If the output, and the equivalence queries are all $\mathcal{C}$ circuits, the algorithm is called $\mathtt{P\text{-}Learn}_{(s,C,p)}$ (proper learning). We omit the size parameter when $s(n) = poly(n)$, and the circuit class when $\mathcal{C}$ = Boolean.

We extend the result of [8] for this exact UTIME learning. The proof is along the same lines except: we use the SAT and TAUT algorithms for solving the equivalence queries, and we use them in a clever order to get the result of the query in an unambiguous fashion.

▶ **Theorem 23.** *Let $p \geq n$ be some polynomial. Then $\forall \delta > 0$, $\mathtt{UEXP}/n^{\delta} \not\subset \mathcal{C}$, if $\forall \epsilon > 0$, $\mathtt{SAT}, \mathtt{TAUT}$, and $\mathtt{Learn}_{(\mathcal{C},p)}$, belong to $\mathtt{UTIME}(2^{n^{\epsilon}})$.*

Proof. Fix a $\delta > 0$. Then, for $\epsilon < \delta' < \delta$, there exists an $a < n^{\delta}$, such that $\mathtt{UEXP}/a \not\subseteq \mathtt{UTIME}(2^{n^{\delta'}})/(a+1)$ (Theorem 19). Starting with the assumption $\mathtt{UEXP}/a \subset \mathcal{C}$, we contradict the UTIME hierarchy. $\mathtt{UEXP}/a \subset \mathcal{C}$ implies that $\mathtt{UEXP}/a = \mathtt{P}^{\#\mathtt{P}}/a$ (using UTIME KLT). For $L \in \mathtt{UEXP}/a$ we have a polynomial time algorithm for L that uses a amount of advice and makes oracle queries to Permanent. Since $\mathtt{P}^{\#\mathtt{P}}/a \subset \mathcal{C}$, Permanent has n^c-size $\mathcal{C}$ circuits, for some constant c. Using $\mathtt{UTIME}(2^{n^{\epsilon}})$ learning algorithms, for n^c-size $\mathcal{C}$ circuits, we learn and compute Permanent in $\mathtt{UTIME}(2^{n^{\delta'}})$. This will give a $\mathtt{UTIME}(2^{n^{\delta'}})/a$ algorithm for L.

Algorithm for computing Permanent *on input x:* For $i = 1$ to $|x|$, let c_i be a circuit that computes permanent on $i \times i$ matrix. We will inductively compute c_i for all $|x| = n$ values of i. Then we will compute $c_n(x)$ to get the final result. For $i = 1$ to $|x|$, do the following:

1. If $i = 1$, let c_i be the trivial circuit (that outputs the input bit itself).
2. Else, run the learning algorithm for c_i and simulate the queries in the following way:
   a. *Membership queries:* For any query y of length i, using downward self-reducibility of permanent we can get the answer by making i queries to the circuit c_{i-1}.
   b. *Equivalence queries:* Let's assume that our current hypothesis is h. We want to know "Does there exists an input z such that $h(z) \neq c_i(z)$?". This is an NP query as we can compute $c_i(z)$ in polynomial time using c_{i-1}. Convert this query and its compliment to SAT and TAUT instances of $poly(n)$ size. Guess a non-deterministic bit z. If $z = 0$, run the UTIME algorithm for TAUT, and in the case of acceptance output h as the c_i circuit. If $z = 1$, run the UTIME algorithm for SAT. If it accepts, then we actually need to give a certificate z such that $h(z) \neq c_i(z)$. Now we make two new NP queries (search to decision) – "Does there exists an input z starting with b such that $h(z) \neq c_i(z)$?" – one for $b = 0$, and one for $b = 1$. Guess answers to both the queries. Create - two $poly(n)$ size SAT instances, and two $poly(n)$ size TAUT instances – by reducing these queries and their compliments. Run the UTIME algorithms on theses queries to verify the two guesses. At least one guess has to have a positive answer. Repeat this procedure again after fixing the first bit of z unambiguously (fix it to 0 if possible, else fix it to 1). This way we get a UTIME algorithm for the original equivalence query.

This algorithm puts $L \in \text{UTIME}(2^{n^{\delta'}})$ as the $\text{UTIME}(2^{n^{\epsilon}})$ learning algorithm is used $poly(n)$ times (once for each c_i), and each time it makes $O(2^{n^{\epsilon}})$ SAT and TAUT queries, each of which can be computed in $\text{UTIME}(2^{n^{\epsilon}})$. ◀

6.3 Generalization of lower bound frameworks

In the above sections we saw that fast UTIME algorithms for certain $\mathcal{C}$ circuit related problems (CAN, TAUT, SAT, Learn), were fed to certain frameworks to yield lower bounds for UTIME against $\mathcal{C}$. Consider the scenario where – a framework is altogether different, or is a fine-grained version of one of the current ones – and works for Boolean circuits, but not for some restriction $\mathcal{C}$. Also consider that, the assumptions of these frameworks are satisfied for that $\mathcal{C}$, but not for unrestricted Boolean circuits. Do we get any lower bounds? In this section we prove that this question has a positive answer.

We use a win-win type argument. We show that, either $\mathtt{P} \not\subset \mathcal{C}$ (i.e., stronger lower bounds exist against $\mathcal{C}$), or faster algorithms for $\mathcal{C}$ circuits imply faster algorithms for Boolean circuits (i.e., frameworks that only work for Boolean circuits can now be used). To prove our results, we use the following folklore lemma.

▶ **Lemma 24.** *If $\mathtt{P} \subseteq \mathcal{C}$, there exists a constant c such that: for large enough n, any s-size circuit has an equivalent s^c-size $\mathcal{C}$ circuit.*

Proof. Ckt-Eval is a problem in P whose input is a Boolean circuit C and a string x, and the output is the output of C on x. If $\mathtt{P} \subseteq \mathcal{C}$, then there is a constant c such that Ckt-Eval has n^c-size $\mathcal{C}$ circuits.

Let B be a P/poly circuit of size n^k, for some constant k. Let E be $(n + kn^k \log n)^c$-size circuit corresponding to the $(n + kn^k \log n)^{th}$-slice of Ckt-Eval. Define $D(x) = E(B, x)$. It is easy to check that, D is a $(n + kn^k \log n)^c$-size $\mathcal{C}$ circuit, that is equivalent to B. ◀

In [39], assumed fast algorithms were applied on witness circuits. To extend their framework, they used the above lemma to show, "either $\mathtt{P} \not\subset \mathcal{C}$, or the Boolean witnesses have equivalent $\mathcal{C}$ circuits, and thus fast algorithms for $\mathcal{C}$ are sufficient". Note that, unlike our result, this approach was local to that particular framework.

▶ **Theorem 25.** *Either* $\mathtt{P} \not\subset \mathcal{C}$*, or* $\exists c\ \forall k$*:*

1. $\mathtt{CAN}_{(\mathcal{C},n^k)} \in \mathtt{UTIME}(t(n)) \implies \mathtt{CAN}_{n^{ck}} \in \mathtt{UTIME}(t(n))$
2. $\mathtt{CAN}_{(\mathcal{C},n^k)} \in \mathtt{UTIME}(t(n)) \wedge \mathtt{TAUT}_{\mathcal{C}} \in \mathtt{UTIME}(t'(n)) \implies \mathtt{TAUT} \in \mathtt{UTIME}((t(n^{ck})+t'(n^{ck}))n)$.
3. $\mathtt{CAN}_{(\mathcal{C},n^k)} \in \mathtt{UTIME}(t(n)) \wedge \mathtt{TAUT}_{\mathcal{C}} \in \mathtt{UTIME}(t'(n)) \wedge \mathtt{SAT}_{\mathcal{C}} \in \mathtt{UTIME}(t''(n))$
   $\implies \mathtt{SAT} \in \mathtt{UTIME}((t(n^{ck}) + t'(n^{ck}))n + t''(n^{ck}))$
4. $\mathtt{P\text{-}Learn}_{(\mathcal{C},n^k)} \in \mathtt{UTIME}(t(n)) \wedge \mathtt{TAUT}_{\mathcal{C}} \in \mathtt{UTIME}(t'(n)) \wedge \mathtt{SAT}_{\mathcal{C}} \in \mathtt{UTIME}(t''(n))$
   $\implies \mathtt{CAN}_{(\mathcal{C},n^k)} \in \mathtt{UTIME}(t(n)(t'(n^k) + t''(n^k))n)$

Proof. If $\mathtt{P} \subset \mathcal{C}$, from the above lemma we know there exists a constant c such that: for each s-size Boolean circuit B, there is an equivalent s^c-size $\mathcal{C}$ circuit C (for large enough n).

Proof of 1: By a simple modification of an algorithm $\mathcal{A}$ for $\mathtt{CAN}_{(\mathcal{C},p)}$, we obtain an algorithm $\mathcal{A}'$ for $\mathtt{CAN}_{p^c}$.

On input B, the algorithm $\mathcal{A}'$ first checks whether B belongs to $\mathcal{C}$. It rejects if the answer is negative. If the answer is positive it simulates $\mathcal{A}$ on B and accepts if and only if $\mathcal{A}$ accepts.

Proof of 2: Let $\mathcal{A}$ be a $\mathtt{UTIME}(t)$ algorithm for $\mathtt{CAN}_{(\mathcal{C},p)}$, $\mathcal{A}'$ be a $\mathtt{UTIME}(t')$ algorithm for $\mathtt{TAUT}_{\mathcal{C}}$. Using $\mathcal{A}$ and $\mathcal{A}'$, we construct a $\mathtt{UTIME}$ algorithm $\mathcal{A}''$ for $\mathtt{TAUT}$.

For input B to $\mathcal{A}''$, for each gate g of B, let B_g be the circuit corresponding to the output wire of gate g. For the output gate o, $\mathcal{A}''$ first guesses an equivalent $\mathcal{C}$ circuit C'_o. To make sure that its guess is unambiguous, it simulates $\mathcal{A}$ on C'_o and rejects if $\mathcal{A}$ rejects. Then it simulates $\mathcal{A}'$ on C'_o (to check if C'_o is a tautology) and rejects if it rejects. The only thing left to check is that C'_o is actually equivalent to C_o.

For checking the consistency of C'_o, $\mathcal{A}''$ first guesses $\mathcal{C}$ circuit C'_g, for each gate g. It then simulates $\mathcal{A}$ on each C'_g and rejects if $\mathcal{A}$ rejects on any of them. Finally it simulates $\mathcal{A}'$ on C''_g for each g, where C''_g is the circuit that captures the tautology "$C'_g = op(C'_{g_1}, \ldots, C'_{g_l})$" for $g = op(g_1, \ldots, g_l)$. It accepts if and only if $\mathcal{A}$ accepts on all of them.

Proof of 3: For input B, with the same strategy as in the proof of 2, we first unambiguously construct an $\mathcal{C}$ circuit C. Then, on this C we simulate a $\mathtt{UTIME}(t'')$ algorithm for $\mathtt{SAT}_{\mathcal{C}}$.

Proof of 4: In a proper learning algorithm, if we have access to the circuit that we are learning, then we can get a canonization algorithm for $\mathcal{C}$ (because the learning algorithm only cares about the truth-table of the circuit that it is learning, and outputs the same hypothesis for all the circuits that have same truth-tables). The membership queries can be handled directly since we have the circuit with us. For the equivalence queries, in the proof of Theorem 23 we saw that we need $\mathtt{TAUT}$ and $\mathtt{SAT}$ algorithms. Since we have the circuit with us, and the hypothesis belongs to $\mathcal{C}$, these queries can be converted into $\mathtt{TAUT}_{\mathcal{C}}$ and $\mathtt{SAT}_{\mathcal{C}}$ queries. So we get a $\mathtt{UTIME}$ algorithm for $\mathtt{CAN}_{(\mathcal{C},n^k)}$. ◀

The final (fourth) point of the above theorem shows that canonization is implied by proper learning, tautology and satisfiability algorithms. Using that, and Corollary 22 and Theorem 23 we can get the following new corollary.

▶ **Corollary 26.** *For* $a \in \omega(n \log n)$*, polynomial* $p(n) \geq n$*,* $\mathtt{USUBEXP}/a \not\subset \mathcal{C}$ *if* $\mathtt{TAUT}_{\mathcal{C}}$*,* $\mathtt{SAT}_{\mathcal{C}}$*, and* $\mathtt{P\text{-}Learn}_{\mathcal{C}}$ *belong to* $\mathtt{UTIME}(2^{n^{o(1)}})$.

7 Conclusions and Open Problems

The main open problem is whether there are any connections between faster algorithms and non-uniform lower bounds possible within determinstic classes such as $\mathtt{EXP}$. In almost all of the prior connections, non-uniformity is simulated with non-determinism, by having a non-deterministic machine guess the appropriate circuit. Can we substitute a recursive argument for non-determinism here? Our results show that, while still allowing non-determinism, the

form of non-determinism can be restricted. In what other ways could we get such connections for smaller classes by restricting the use of non-determinism? Finally, the circuit model combines two features: time and non-unformity. Can we get a finer-grained version of easy-witness lemma by distinguishing these two parameters?

References

1 Eric Allender. When worlds collide: Derandomization, lower bounds, and kolmogorov complexity. In Ramesh Hariharan, V. Vinay, and Madhavan Mukund, editors, *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science*, pages 1–15, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

2 Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. doi:10.1007/BF00116828.

3 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

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

5 Marco Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Tighter connections between derandomization and circuit lower bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 645–658, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.645.

6 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016. doi:10.1145/2840728.2840746.

7 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158, 1971. URL: http://doi.acm.org/10.1145/800157.805047, doi:10.1145/800157.805047.

8 Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009. doi:10.1016/j.jcss.2008.07.006.

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

10 Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 348–355, New York, NY, USA, 2005. ACM. URL: http://doi.acm.org/10.1145/1060590.1060642, doi:10.1145/1060590.1060642.

11 Ryan C. Harkins and John M. Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. *TOCT*, 5(4):18:1–18:11, 2013. doi:10.1145/2539126.2539130.

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

13 Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, pages 237–240, 1999. doi:10.1109/CCC.1999.766282.

14 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *39th Annual Symposium on Foundations of Computer Science,*

FOCS '98, November 8-11, 1998, Palo Alto, California, USA, pages 653–663, 1998. doi:10.1109/SFCS.1998.743516.

15 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, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997. doi:10.1145/258533.258590.

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

17 Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 749–760, 2015. doi:10.1007/978-3-662-47672-7_61.

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

19 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.

20 Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pages 302–309, New York, NY, USA, 1980. ACM. URL: http://doi.acm.org/10.1145/800141.804678, doi:10.1145/800141.804678.

21 Adam R. Klivans, Pravesh Kothari, and Igor Carboni Oliveira. Constructing hard functions using learning algorithms. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 86–97, 2013. doi:10.1109/CCC.2013.18.

22 Ker-I Ko. Some observations on the probabilistic algorithms and np-hard problems. *Information Processing Letters*, 14(1):39 – 43, 1982. URL: http://www.sciencedirect.com/science/article/pii/0020019082901399, doi:https://doi.org/10.1016/0020-0190(82)90139-9.

23 Matthias Krause and Stefan Lucks. Pseudorandom functions in tc0 and cryptographic limitations to proving lower bounds. *Comput. Complex.*, 10(4):297–313, May 2002. URL: http://dx.doi.org/10.1007/s000370100002, doi:10.1007/s000370100002.

24 Leonid A. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

25 Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 68–85, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

26 Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901, 2018. doi:10.1145/3188745.3188910.

27 Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, March 2004. URL: http://doi.acm.org/10.1145/972639.972643, doi:10.1145/972639.972643.

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

29 Igor Carboni Oliveira. Algorithms versus circuit lower bounds. *CoRR*, abs/1309.0249, 2013. URL: http://arxiv.org/abs/1309.0249, arXiv:1309.0249.

30 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18.

31 Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24 – 35, 1997. URL: http://www.sciencedirect.com/science/article/pii/S002200009791494X, doi:https://doi.org/10.1006/jcss.1997.1494.

32 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

33 Iannis Tourlakis. Time–space tradeoffs for sat on nonuniform machines. *Journal of Computer and System Sciences*, 63(2):268 – 287, 2001. URL: http://www.sciencedirect.com/science/article/pii/S0022000001917672, doi:https://doi.org/10.1006/jcss.2001.1767.

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

35 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

36 R. Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. doi:10.1137/130938219.

37 R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory of Computing*, 14(1):1–25, 2018. doi:10.4086/toc.2018.v014a017.

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

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

A UEXP Lower Bounds from Half-sub-exponential Unambiguous Algorithms

From [38] we know: a deterministic half-sub-exponential 3-SAT algorithm imply EXP $\not\subseteq$ SIZE($poly$). Two interesting algorithm design questions are: "Does faster non-deterministic algorithms exist for TAUT/CAPP?" and "Does faster unambiguous non-deterministic algorithms exist for SAT?". Here we show that (strong) positive answers imply UEXP lower bounds.

Main idea: UEXP/$a \subset$ SIZE($poly$) with UTIME KLT implies UEXP/$a = \Sigma_2/a = \Pi_2/a =$ MA/a. Then, for $L \in$ UEXP/a, we unfold the quantifiers of the MA, Σ_2, Π_2 algorithms, and use the faster algorithms (from the assumptions) to contradict the UTIME hierarchy.

▶ **Theorem 27.** *For $\delta \leq 1$, let a and ϵ be the parameters of Theorem 19 for the time bound $t = 2^{\delta n}$. Let $f, g, h : \mathbb{N} \to \mathbb{N}$ satisfy $f(g(n^k)^k) \in O(2^{\epsilon n})$ and $h(n^k) \in O(2^{\epsilon n})$ for all constants k. Then* UTIME$(2^{\delta n})/a \not\subset$ SIZE($poly$) *if*

1. 3-SAT $\in$ UTIME$(f)/1$ *and* 3-TAUT $\in$ NTIME(g)*; or*
2. 3-SAT $\in$ UTIME(f) *and* 3-TAUT $\in$ NTIME$(g)/1$*; or*
3. Σ_2-SAT $\in$ UTIME$(h)/1$ (Π_2-SAT $\in$ UTIME$(h)/1$)*; or*
4. 3-SAT $\in$ UTIME$(f)/1$ *and* CAPP $\in$ NTIME(g)*; or*
5. 3-SAT $\in$ UTIME(f) *and* CAPP $\in$ NTIME$(g)/1$.

Proof. Starting with UTIME$(2^{\delta n})/a \subseteq$ SIZE($poly$) we get UEXP/$a \subseteq$ SIZE($poly$). From UTIME KLT (Theorem 4) UEXP/$a =$ MA/a. Thus UTIME$(2^{\delta n})/a \subseteq \Sigma_2/a$ (or Π_2/a, or MA/a). For $L \in$ UTIME$(2^{\delta n})/a$, using any one of the above five assumptions, and any Σ_2/a (or Π_2/a, or MA/a) algorithm for L, we design a UTIME$(2^{\epsilon n})/(a+1)$ algorithm for L to contradict the UTIME hierarchy (Theorem 19).

Contradicting the first assumption: Let M be a Σ_2/a machine that accepts L. Let N be the Co-NP machine obtained by starting M at its alternation, i.e., after removing the existential quantifier, and including the variables under it, into the input. Assuming 3-TAUT $\in$ NTIME(g) we get that N has an equivalent machine N' that guesses $g(poly(n))$ bits and then runs for $g(poly(n))$ time. Thus M has an equivalent machine M' that guesses

$poly(n) + g(poly(n))$ bits and then runs for $g(poly(n))$ time. Note that, N, N', M' all use the same advice as M. Now this machine M' can be turned into an equivalent 3-SAT instance of $poly(g(poly(n)))$ size. Now by the assumption $\texttt{3-SAT} \in \texttt{UTIME}(f)/1$ we get a $\texttt{UTIME}(f(poly(g(poly(n)))))/(a+1)$ algorithm for L.

Contradicting the second assumption: The argument is exactly the same as that for the first assumption, expect that the extra 1 bit of advice is now used at an earlier stage of the algorithm.

Contradicting the third assumption: The Σ_2/a (Π_2/a) algorithm for L can be converted into a Σ_2-SAT (Π_2-SAT) instance of $poly(n)$ size in $poly(n)$ time. This conversion needs the original a bits of advice. Now a $\texttt{UTIME}(h)/1$ algorithm for Σ_2-SAT (Π_2-SAT) gives a $\texttt{UTIME}(h(poly(n)))/(a+1)$ algorithm for L.

Contradicting the fourth assumption: Let M be an $\texttt{MA}/a$ machine that accepts L. Let N be the BPP machine of Arthur (whose input also includes the non-determinism of Merlin). Assuming $\texttt{CAPP} \in \texttt{NTIME}(g)$ we get that N has an equivalent machine N' that guesses $g(poly(n))$ bits and then runs for $g(poly(n))$ time. Thus M has an equivalent machine M' that guesses $poly(n) + g(poly(n))$ bits and then runs for $g(poly(n))$ time. Note that, N, N', M' all use the same advice as M. The machine M' can be turned into an equivalent 3-SAT instance of $poly(g(poly(n)))$ size. By the assumption $\texttt{3-SAT} \in \texttt{UTIME}(f)/1$ we get a $\texttt{UTIME}(f(poly(g(poly(n)))))/(a+1)$ algorithm for L.

Contradicting the fifth assumption: The argument is exactly the same as that for the fourth assumption, expect that the extra 1 bit of advice is now used at an earlier stage of the algorithm. ◀

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il