Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 167 (2019)

On Limiting & Limited Non-determinism in NEXP Lower Bounds

ANANT DHAYAL, University of California San Diego, USA
RUSSELL IMPAGLIAZZO, University of California San Diego, USA

Proving circuit lower bounds is one of the most difficult tasks in computational complexity theory. The NP vs. P/*poly* problem asks whether there are small non-uniform circuits that can simulate circuit satisfiability. The answer is widely believed to be false, but so far progress has only been made in the case of restricted circuits. In 1980s the progress stalled after it was shown that NP doesn't have non-uniform AC^0 circuits that have MOD-m gates for any prime m. After almost three decades, in 2010s Williams made progress in the relaxed case of NEXP lower bounds. He first showed that non-trivial satisfiability algorithms for a circuit class entail NEXP lower bounds against that class. Then he designed a fast satisfiability algorithm for ACC circuits (AC^0 circuits with MOD-m gates for any constant m) to show that NEXP doesn't have non-uniform ACC circuits.

We make progress in bringing down the class NEXP, specifically by **limiting non-determinism** (in terms of the number of non-deterministic branches that accept). We show that slightly faster satisfiability algorithms entail lower bounds for UEXP and related classes. We believe this is progress towards making similar connections, and thus proving lower bounds, for EXP and lower complexity classes.

To investigate why progress again stalled around ACC lower bounds, and why TC^0 (AC^0 circuits with majority gates) lower bounds have not been established yet, Williams made rigorous connections between NEXP lower bounds and variations of Natural Proofs. Razborov and Rudich defined Natural Proofs to showcase the limitations of the current lower bound techniques. They showed that any technique that entails Natural Proofs, i.e. Proofs that are (i) constructive, (ii) useful, and (iii) large, fail to prove strong lower bounds. Williams showed that NEXP lower bounds, regardless of the technique, entail Proofs that satisfy the first two of the three conditions of Natural Proofs.

We make Williams connections more rigorous, and show that UEXP lower bounds entail Proofs, that in addition to the first two conditions of Natural Proofs, satisfy a third condition that is exactly the opposite of largeness condition. We call this condition, the uniqueness condition, and these Proofs, the Unique Proofs. These connections showcase that $\mathsf{NEXP} \supseteq \mathsf{UEXP} \supseteq \mathsf{EXP}$ is a viable path to approach EXP lower bounds.

We also discuss an alternate approach to improve NEXP lower bounds. We define a new form of non-determinism to capture the non-uniform circuits from the class $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, and call it promise-Single-Valued non-determinism. We show that in the current NEXP lower bounds, we can allow the non-uniform circuits some **limited non-determinism** (in terms of the number of non-deterministic inputs) of the type promise-Single-Valued. We also discuss that, how small improvements in the amount of this special type of non-determinism, even in the restricted circuits much weaker than ACC, would imply very strong lower bounds such as $\mathsf{NEXP} \not\subset \mathsf{P}/poly$.

CCS Concepts: • **Theory of computation → Complexity classes**; **Circuit complexity**.

Additional Key Words and Phrases: Circuit complexity, lower bounds, satisfiability, Natural Proofs, Unique Proofs, promise-Single-Valued non-determinism

Authors' addresses: Anant Dhayal, adhayal@eng.ucsd.edu, University of California San Diego, La Jolla, CA, USA; Russell Impagliazzo, russell@eng.ucsd.edu, University of California San Diego, La Jolla, CA, USA.

XXXX-XXXX/2020/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ISSN 1433-8092

ACM Reference Format:
Anant Dhayal and Russell Impagliazzo. 2020. On Limiting & Limited Non-determinism in NEXP Lower Bounds. 1, 1 (December 2020), 38 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The two fundamental problems in theoretical computer science are: (a) design non-trivial algorithms for computational tasks; or (b) prove that such algorithms do not exist, i.e. prove (circuit) lower bounds for the computational task in question. For instance, the famous 1970s question of Cook and Levin, is $\mathsf{NP} = \mathsf{P}$ or $\mathsf{NP} \neq \mathsf{P}$ [25, 61]? Or its non-uniform version, is $\mathsf{NP} \subset \mathsf{P}/poly$ or $\mathsf{NP} \not\subset \mathsf{P}/poly$ (it would also imply $\mathsf{NP} \neq \mathsf{P}$)? $\mathsf{P}/poly$ is the class of problems that have non-uniform polynomial-size circuits. Non-uniform computation allows the sizes of programs to grow with the sizes of inputs, and can be naturally represented as an infinite family of Boolean circuits (one for each possible input length). They are very powerful and can simulate all undecidable problems when there is no size restriction, and certain undecidable problems even with polynomial size restriction. But it's still not known if $\mathsf{P}/poly$ can simulate NP. In fact the answer is believed to be false. One of the main reasons being the collapse of polynomial hierarchy. This is second to the $\mathsf{P} = \mathsf{NP}$ collapse, in the list of collapses that are widely believed to be false by complexity theorists.

No significant progress after decades of efforts lead to the pursuit of considerably relaxed versions of the above pair of questions. Here we focus on the lower bounds side of the question. Optimists started the bottom-up approach and started proving lower bounds for restricted circuit classes with a hope that gradually they would lift the restrictions over time.

If we just consider size restrictions, the best known lower bound for any NP problem is $5n - o(n)$, for circuits over the basis of 2-bit AND, OR, and NOT gates [49, 60]. MA (with 1-bit of advice) is the lowest class that contains functions with super-linear and fixed-polynomial (size n^k for any constant k) lower bounds [79]. If we consider restrictions on gate types (basis), monotone circuits, i.e. circuits without negations or NOT gates are the most studied. A super-polynomial lower bound was proved for NP in [76] (for CLIQUE to be specific), and improved to exponential size in [5]. But later in [77] it was shown that these techniques for monotone circuit lower bounds would not extend to general circuits.

Here we only focus on the depth restrictions. Constant depth restrictions are the most studied ones. In 1980s it was shown that the parity function on n bits, which is in P (infact in even lower complexity classes), can't be computed by AC^0 circuits [2, 34]: the class of constant depth circuits over the basis of AND, OR, and NOT gates of arbitrary fan-in. Later the lower bound was improved to exponential size in [102], and eventually an optimal lower bound was established in [37]. Then the next question in line was about the power of AC^0 circuits with parity gates. Let $\mathsf{AC}^0[\oplus_p]$ denote the class of AC^0 circuits with MOD-p gates. An exponential size lower bound was proved for $\mathsf{AC}^0[\oplus_2]$ in [77] for the majority function on n bits. Majority too lies in P and lower classes. For any two primes $p \neq q$, an exponential size lower bound for the MOD-p function was proved for the $\mathsf{AC}^0[\oplus_q]$ circuits in [86]. Based on these lower bounds, the logical next step was to move towards the following two classes that are more expressive: (i) ACC, the class of AC^0 circuits with MOD-m gates for arbitrary constant $m > 1$; (ii) TC^0, the class of AC^0 circuits with majority (or equivalently, threshold) gates. Note that, TC^0 can simulate ACC.

Failing to prove that NP is not contained in polynomial-size non-uniform ACC circuits, the lower bound question was relaxed further. The next question asked was, whether this lower bound can be established for NEXP, or even $\mathsf{EXP}^{\mathsf{NP}}$. Note that, MAEXP is the smallest class known to have super-polynomial lower bounds for unrestricted Boolean circuits [13] (although, in [53] it was shown that $\mathsf{NEXP}^{\mathsf{RP}}$, that is contained in MAEXP, can't have polynomial-size circuits of both types, Boolean and arithmetic). MAEXP contains NEXP, and is incomparable to $\mathsf{EXP}^{\mathsf{NP}}$. Even for $\mathsf{EXP}^{\mathsf{NP}}$, the super-polynomial lower bound for ACC (even depth-3 $\mathsf{AC}^0[\oplus_6]$) were elusive for about three decades. In his seminal work in 2010s, Williams [99] showed super-polynomial ACC lower bounds for NEXP, and exponential ACC lower bounds for $\mathsf{EXP}^{\mathsf{NP}}$. He used connections between fast algorithms and lower bounds from his prior work [98]. For TC^0 lower bounds, MAEXP is still the smallest class we know.

Pessimists on the other hand started formulating barriers to show that all the techniques used in the bottom-up approach are not good enough to prove stronger lower bounds. There are three main barriers in the literature that any lower bound technique should overcome: (i) Relativization [10], (ii) Algebrization or Algebric Relativization [1], and (iii) Natural Proofs barrier [78]. The first two barriers essentially say that techniques that relativize, i.e. work even in presence of arbitrary oracles, fail to prove most of the lower bounds. This is because, for most of the lower bound questions, there are some oracles relative to which the lower bound is known to hold, and there are other oracles relative to which the lower bound is not known to hold. In this paper we will only focus on the third barrier.

The Natural Proofs barrier of Razborov and Rudich [78] argues that almost all known proofs of nonuniform circuit lower bounds, entail algorithms/properties that: (i) are efficient (*constructivity*), (ii) distinguish *hard* functions from *easy* by only accepting hard functions (*usefulness*), (iii) accept many hard functions (*largeness*). Here by hard (*resp.* easy) we mean that the function requires bigger (*resp.* smaller) circuits to compute, typically super-polynomial (*resp.* polynomial) in size. Any such algorithm would refute widely believed cryptography primitives, and thus Natural Proofs are self-limiting in the sense that: in order to prove weak lower bounds, they provide algorithms that refute strong lower bounds that are also believed to be true. Even the small class TC^0 supports cryptography [59, 64, 69], and to prove lower bounds against it we would need un-Natural techniques.

Main results: We focus on three different directions in this paper: (i) We describe ways in which Williams fast algorithms to lower bounds connection from [98] can be extended to lower bounds for classes smaller than NEXP, by **limiting** the non-determinism used by NEXP (in terms of the number of non-deterministic branches that accept). This can be viewed as a progress towards establishing similar connections, and hence proving lower bounds, for EXP and lower complexity classes. (ii) Then we discuss how lower bounds for these smaller classes evade the Natural Proofs barrier. (iii) Finally we extend the current NEXP lower bounds by allowing circuit classes a **limited** amount of non-determinism. We also discuss how small increments in our results would lead to much stronger lower bounds such as $\mathsf{NEXP} \not\subset \mathsf{P}/poly$. This can also be viewed as a new bottom-up approach, where gradually lifting the restriction on the amount of non-determinism used by the circuits, would lead to $\mathsf{NEXP} \not\subset \mathsf{P}/poly$.

1.1 Fast (unambiguous) algorithms imply UEXP circuit lower bounds

In the direction of algorithm-design too, the questions were relaxed. The intial question was: do NP-complete languages have polynomial time algorithms? The rlaxed version is: do they have algorithms that beat the naive brute-force strategy? For solving circuit satisfiability (Ckt-SAT), the canonical NP-complete language, the naive approach runs in $2^n m$ deterministic time for n-input m-size circuits. The relaxed questions are: Is it possible to design a $2^{cn}poly(m)$ time algorithm for any constant $c < 1$? Or even $2^n poly(m)/sp(n)$ time algorithm for some super-polynomial function sp? While pessimists formulated many conjunctures [20, 43, 45, 96] believing that no substantial progress is possible, optimists took the bottom-up approach and started designing fast algorithms for restricted classes. The most studied restrictions in this line are: $\mathsf{3\text{-}CNF} \subseteq \mathsf{k\text{-}CNF} \subseteq \mathsf{CNF} \subseteq \mathsf{AC}^0 \subseteq \mathsf{ACC} \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{NC} \subseteq \mathsf{P}/poly$ [17, 18, 23, 39, 42, 44, 63, 65, 67, 74, 75, 80–82, 84, 99, 101].

We often think of algorithm design and lower bounds as being antithetical, for instance: the two statements, $\mathsf{P} = \mathsf{NP}$ and $\mathsf{P} \neq \mathsf{NP}$, can't be simultaneously true. But there have been a series of results showing that efficient algorithm for certain problem in certain computation model, implies a lower bound for related problem in other computation model [41, 53, 68, 98, 99, 101]. The intuition behind these connections is: if there is a fast algorithm for a circuit class, then the algorithm must be exploiting a *simple* structure or pattern that exists in that class, and thus that class can't simulate *complex* classes.

Williams in his ACC lower bound result, first connects the relaxed versions of $\mathsf{P} = \mathsf{NP}$ and $\mathsf{NP} \not\subset \mathsf{P}/poly$ that we discussed above [98]: for any super-polynomial function sp, a $2^n/sp(n)$ time Ckt-SAT algorithm for polynomial-size Boolean circuits implies $\mathsf{NEXP} \not\subset \mathsf{P}/poly$. Infact, an equally fast non-deterministic algorithm for circuit tautology suffices. His connection (and many others in the literature) also preserve the nature of circuits. That is, fast satisfiability algorithm for a restricted sub-class of Boolean circuits C, implies lower bounds against C circuits. He developed fast satisfiability algorithm for ACC circuits, and then used this connection to establish NEXP and $\mathsf{EXP}^{\mathsf{NP}}$ lower bounds [99].

Unfortunately, most of these connections are only known to show circuit lower bounds in relatively large complexity classes such as NEXP or $\mathsf{EXP}^{\mathsf{NP}}$ (although [68] extends this to scaled-down versions of these classes). Establishment of similar connections for lower classes like EXP, can be seen as the first step towards establishing lower bounds for them.

One of the main focus of this paper is to extend these connections to non-uniform lower bounds for the class UEXP of languages recognized by unambiguous non-deterministic machines, and to related classes. Since UEXP lies between EXP and NEXP, we believe that lower bounds for UEXP based on algorithms would be progress towards making similar connections for EXP. We don't known how far UEXP is from NEXP. In [93] a randomized reduction was given from NP to promise-UP which only succeeds with a low probability. In [27] it was shown that derandomizing this reduction, or even increasing the success probability, will have unlikely consequences.

We show that fast unambiguous non-deterministic circuit analysis algorithms imply circuit lower bounds for UEXP. In our first result we use fast tautology and canonization algorithms. Roughly speaking, a canonization algorithm for a circuit class, is an algorithm that only accepts one circuit per function or truth-table, from that class (see Section 4.1 for a technical definition).

Theorem 1.1. USUBE *algorithms for tautology and canonization of* C *imply* $\mathsf{UE}/O(n\log n) \not\subset C$.

We use C to denote any non-uniform circuit class from the set $\{\mathsf{AC}_0, \mathsf{ACC}_0, \mathsf{TC}_0, \mathsf{NC}_1, \mathsf{NC}, \mathsf{P}/poly\}$. This set is called *typical* in the literature. We measure the complexity of any circuit analysis algorithm in terms of the input wires, and not the circuit size. We omit the size parameter for circuits if it's polynomial, i.e. $C(poly) = C$. Here by SUBE (sub-E) we denote the class of languages that have $2^{\epsilon n}$-time algorithm for each $\epsilon > 0$. USUBE is the extension to unambiguous non-deterministic algorithms.

Since canonization is not prominently used circuit analysis algorithm in the lower bound literature, we replace the assumption of fast canonization with different circuit analysis algorithms. Based on the definition of canonization, a fast $\Pi_2\mathsf{SAT}$ algorithm would be an ideal replacement. We derive the following theorem.

Theorem 1.2. *For every constant* k_1 *there is a constant* k_2*, such that if* $\Pi_2\mathsf{SAT}$ *on* n *variables and* n *clauses can be solved in* $\mathsf{UTIME}(2^{n/(\log n)^{k_2}})$*, then* $\mathsf{UE}/n \not\subset \mathsf{SIZE}(n(\log n)^{k_1})$.

Relaxing the unambiguity condition a little, helps us totally get rid of the fast canonization assumption. One other well studied variant of $\mathsf{UTIME}(t)$ is $\mathsf{FewTIME}(t)$, which was first defined in [3]. $L \in \mathsf{FewTIME}(t)$, 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. We get results for a slightly relaxed version of FewTIME, the class $\widetilde{\mathsf{FewTIME}}(t)$.

Definition 1.3. $\widetilde{\mathtt{Few}}\mathtt{TIME}(t)$ is the class of problems decidable by $\mathtt{NTIME}(t)$ verifiers, where the number of accepting paths are bounded by $2^{2^{(\log\log t)^2}}$.

Actually, even if we bound the number of accepting paths by $2^{2^{sc(t)\log\log t}}$ for any super constant function sc, the definition would satisfy all our results. We use $sc(t) = \log\log t$ for the sake of cleaner presentation. Note that, this definition allows any $\widetilde{\mathtt{Few}}\mathtt{E}$ verifier to have $2^{2^{(\log n)^2+O(\log n)}}$ many accepting paths compared to the $2^{2^{\log n+O(1)}}$ upper bound for the $\mathtt{FewE}$ verifiers. But it's still very 'few' compared to the maximum possible by an $\mathtt{NE}$ verifier, that is $2^{2^{O(n)}}$. Definition of $\widetilde{\mathtt{Few}}\mathtt{SUBE}$ is analogous to $\mathtt{USUBE}$. We derive the following result.

Theorem 1.4. *$\widetilde{\mathtt{Few}}\mathtt{SUBE}$ tautology algorithm for C circuits implies $\forall k$ $\widetilde{\mathtt{Few}}\mathtt{E}/O(n\log n) \not\subset C(n^k)$.*

Lastly, we replace canonization with exact proper learning (that makes membership and equivalence queries) and derive the following result. Here by proper we mean that any hypothesis produced by the learning algorithm is a polynomial-size C circuit, i.e., belongs to the class being learned (see Section 4.4 for a technical definition).

Theorem 1.5. *$\mathtt{USUBE}$ algorithms for tautology and proper-learning of C imply $\mathtt{UE}/O(n\log n) \not\subset C$.*

Almost all of our connections work for any typical restricted circuit class C. But still we derive some translation results to show that, any lower bound frame work (that may be different from ours), if uses certain set of fast unambiguous circuit analysis algorithms, and only works for unrestricted Boolean circuits, now can be used for C. These translations are tight enough to be useful in the scenario where the new framework only requires the algorithms to be $\mathtt{UTIME}(2^n/sp(n))$ for some super-polynomial function $sp(n)$, as opposed to our frameworks that require $\mathtt{USUBE}$ algorithms.

Theorem 1.6. *Either $\mathtt{P} \not\subset C$, or:*

(1) *$\mathtt{UTIME}(2^n/n^{\omega(1)})$ tautology and canonization algorithms for C, imply $\mathtt{UTIME}(2^n/n^{\omega(1)})$ tautology and canonization algorithm for unrestricted Boolean circuits; and*
(2) *$\widetilde{\mathtt{Few}}\mathtt{TIME}(2^n/n^{\omega(1)})$ tautology algorithm for C, implies $\widetilde{\mathtt{Few}}\mathtt{TIME}(2^n/n^{\omega(1)})$ tautology algorithm for unrestricted Boolean circuits.*

1.2 UEXP lower bounds are constructive, useful, and unique

To inquire why lower bounds for smaller classes like $\mathtt{TC}^0$ are still open after decades of efforts, Williams proved rigorous equivalences between $\mathtt{NEXP}$ lower bounds and useful properties that are constructive [100]. He showed that the lower bound $\mathtt{NEXP} \not\subset C$ is equivalent to the existence of $\mathtt{P}/\log n$ constructive property against C circuits. As it is believed that there can't be any Natural Proofs against $\mathtt{TC}^0$ and more expressive circuit classes, this is a negative result in the sense that any $\mathtt{NEXP}$ lower bound, already satisfies two of the three conditions of Natural Proofs, regardless of the technique used to obtain it.

We extend these results to characterize $\mathtt{UEXP}$ lower bounds. Our connections show that the future of $\mathtt{UEXP}$ lower bounds is brighter in the sense that, they not only 'not satisfy' the third condition of Natural Proofs, but they satisfy a different third condition that is totally opposite of largeness. We introduce a new notion called Unique Proofs. Unique properties are 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 derive the following equivalence.

Theorem 1.7. *$\mathtt{UE}/n \not\subset C$ if and only if a $\mathtt{P}/\log n$ computable unique property exists against C.*

In [72] the $\mathsf{NE} \cap \mathsf{Co\text{-}NE} \not\subset C$ lower bound was shown to yield a P computable property against C. The equivalence was also conjectured to be true. We prove that conjecture for the case of $\mathsf{UE} \cap \mathsf{Co\text{-}UE}$ lower bounds and P computable unique properties.

THEOREM 1.8. $\mathsf{UE} \cap \mathsf{Co\text{-}UE} \not\subset C$ *if and only if a* P *computable unique property exists against* C.

As an application of these connections, we get $\mathsf{USUBEXP}$ derandomization of BPP from different UEXP lower bounds.

THEOREM 1.9. (1) $\mathsf{UEXP} \not\subset \mathsf{SIZE}(poly) \implies \mathsf{BPP} \subset \cap_{\epsilon>0}$ *io*-$\mathsf{UTIME}(n^{\epsilon})/n^{\epsilon}$
(2) $\mathsf{UEXP} \neq \mathsf{BPP} \implies \mathsf{BPP} \subset \cap_{\epsilon>0}$ *io-Heur*-$\mathsf{UTIME}(n^{\epsilon})/n^{\epsilon}$
(3) $\mathsf{UEXP} \cap \mathsf{Co\text{-}UEXP} \not\subset \mathsf{SIZE}(poly) \implies \mathsf{BPP} \subset \cap_{\epsilon>0}$ *io*-$\mathsf{UTIME}(n^{\epsilon})$
(4) $\mathsf{UEXP} \cap \mathsf{Co\text{-}UEXP} \neq \mathsf{BPP} \implies \mathsf{BPP} \subset \cap_{\epsilon>0}$ *io-Heur*-$\mathsf{UTIME}(n^{\epsilon})$

1.3 Gradually increasing the non-determinism in circuits for NEXP lower bounds: With a hope to prove $\mathsf{NEXP} \not\subset \mathsf{P}/poly$

We extend the known NEXP lower bounds by allowing the restricted circuit classes some amount of non-determinism. We also discuss why it would be difficult to increase this non-determinism without proving stronger lower bounds, such as $\mathsf{NEXP} \not\subset \mathsf{P}/poly$. This can also be seen as an approach to prove $\mathsf{NEXP} \not\subset \mathsf{P}/poly$, by gradually increasing the non-determinism in circuits in our current lower bounds.

We use a weaker version of non-determinism (due to technical difficulties that we discuss in the next section), but we'll see that even this kind of non-determinism is very powerful. We first define all the types of non-determinism that circuits in our lower bounds would use.

$\mathsf{P}/poly$ is equivalent to the class of non-uniform polynomial size circuits. A language $L \in \mathsf{SIZE}(s)$ if L is accepted by a sequence of deterministic Boolean circuits $\{C_n\}_{n\in\mathbb{N}}$ of size $O(s(n))$, where C_n computes L_n (n^{th}-slice of L) and the size is measured by the number of wires.

Similarly $\mathsf{NP}/poly$ is equivalent to the class of non-uniform non-deterministic polynomial size circuits. A non-deterministic circuit has extra guess inputs, and the circuit accepts an input, if there is a setting of these guess inputs that makes the output 1.

Definition 1.10. A language $L \in \mathsf{NSIZE}(s)$ if L is accepted by a sequence of non-deterministic Boolean circuits $\{C_n\}_{n\in\mathbb{N}}$ of size $O(s(n))$. C_n receives two inputs, x of length n and guess input y. The function $f_C : \{0,1\}^n \to \{0,1\}$ computed by C satisfies, $f_C(x) = 1 \iff \exists y\, C(x,y) = 1$.

$\mathsf{NP}/poly \cap \mathsf{Co\text{-}NP}/poly$ is equivalent to the class of languages that have non-deterministic circuits for both, the language and its complement. These circuits can also be combined to output an equivalent Single-Valued or SV circuit.

Definition 1.11. A language $L \in \mathsf{SVSIZE}(s)$ if L is accepted by a sequence of non-deterministic Single-Valued Boolean circuits $\{C_n\}_{n\in\mathbb{N}}$ of size $O(s(n))$. C_n receives two inputs, x of length n and guess input y. C_n has two outputs, $Flag_{C_n}$ and $Value_{C_n}$. The circuit C_n computes function $f_C : \{0,1\}^n \to \{0,1\}$ if it satisfies the following two promises for any input x: (a) $\exists y\, Flag_{C_n}(x,y) = 1$; (b) $\forall y\, Flag_{C_n}(x,y) = 1 \implies Value_{C_n}(x,y) = f_C(x)$.

We define a new type of non-determinism to capture the class $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. This class is weaker than $\mathsf{NP}/poly \cap \mathsf{Co\text{-}NP}/poly$ as both the algorithms need to be complimentary on all the advice sequences. For any language $L \in (\mathsf{NTIME}(t) \cap \mathsf{Co\text{-}NTIME}(t))/t$, there is a $\mathsf{DTIME}(t^2)$ algorithm, that on any t size input (which is actually the advice), outputs a pair of non-deterministic circuits (that correspond to the $\mathsf{NTIME}(t)$ and $\mathsf{Co\text{-}NTIME}(t)$ algorithms) that accept complimentary set of inputs, and there is an infinite sequence of $t(n)$-size inputs (one for each $n \in \mathbb{N}$) for which the produced pair of circuits accept L and $\overline{L}$. We call such pair of circuits promise-SV or prSV, since there is an algorithm that produces these circuits, and the algorithm satisfies the promise of always producing circuits that can be combined to become SV. We call the underlying algorithm, a prSV algorithm.

Definition 1.12. A linear-time algorithm $\mathcal{A}$ is called prSV if on each input it outputs a pair of non-deterministic circuits that accepts some n'-bit function $f_{n'}$ and its compliment, for some $n' \leq n$. We say $L \in \mathsf{prSV}^{\mathcal{A}}\mathsf{SIZE}(s)$, if for $n \in \mathbb{N}$, L_n has $O(s(n))$ size non-deterministic circuits C_n and C'_n, deciding it and its compliment. Additionally, these circuits are produced by the prSV algorithm $\mathcal{A}$ on some $s(n)$-length input. $L \in \mathsf{prSVSIZE}(s)$ denotes that $L \in \mathsf{prSV}^{\mathcal{A}}\mathsf{SIZE}(s)$ for some prSV algorithm $\mathcal{A}$.

The equation $\mathsf{prSVSIZE}(poly) = (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$ follows from the definition. We also get, $(\mathsf{NTIME}(n^{k/2}) \cap \mathsf{Co\text{-}NTIME}(n^{k/2}))/n^{k/2} \subseteq \mathsf{prSVSIZE}(n^k) \subseteq (\mathsf{NTIME}(n^k) \cap \mathsf{Co\text{-}NTIME}(n^k))/n^k$.

These definitions of non-deterministic circuits naturally extend to any restricted circuit class C of Boolean circuits. For the prSV circuits, the underlying prSV algorithm satisfies an extra promise of always producing C circuits. This gives us $C(s) \subseteq \mathsf{prSV}\text{-}C(s) \subseteq \mathsf{SV}\text{-}C(s) \subseteq \mathsf{N}\text{-}C(s)$ for any size parameter $n \leq s(n) \leq 2^n$. We use $\mathsf{prSV}^a\text{-}C(s)$ to denote $C(s)$ circuit that uses a amount of prSV non-determinism. $\mathsf{SV}^a\text{-}C(s)$ and $\mathsf{N}^a\text{-}C(s)$ are defined similarly.

Now, we discuss why non-determinism, even the prSV type, is very powerful and can lead to big lower bounds.

Theorem 1.13. *The class* $\mathsf{SIZE}(s)$ *is contained in* $\mathsf{prSV\text{-}3\text{-}CNF}(s \log n)$.

Proof. For $L \in \mathsf{SIZE}(s)$ we give a $\mathsf{prSV\text{-}3\text{-}CNF}(s \log n)$ circuit sequence. The underlying prSV algorithm treats its inputs as $\mathsf{SIZE}(s)$ circuits, converts fan-in of each gate to two by adding more gates, and then converts the input circuit and its compliment into two 3-CNF circuits with $O(s \log n)$ guess inputs. The algorithm applies Tseitin transformation [90]: for each gate $g(x, y)$ it introduces a new variable y that it labels as guess input, and adds clauses for the equation $y = g(x, y)$. Finally adds a clause with just the guess variable that represents the output gate. □

Such conversions were discussed in [66] for non-deterministic circuits, we observe that they also extend to prSV circuits. In fact unambiguous prSV non-determinism suffices (the guess inputs introduced in the proof represents gates of the original deterministic circuit, and take unique values on any input). Also, the multiplicative $\log n$ factor can be removed if we start with fan-in two unrestricted Boolean circuits.

The above theorem shows that lower bounds against restricted circuits that contain 3-CNF and use prSV type of non-determinism, imply lower bounds against unrestricted Boolean circuits. We derive lower bounds for NE and E^{NP} against such classes, with limited non-determinism. Increasing the non-determinism in our results won't be possible without proving lower bounds like 'E^{NP} is not simulated by linear-size fan-in-two unrestricted Boolean circuits', which are still very far from the reach of current lower bound techniques. Our results also give hope of obtaining TC^0 lower bounds, if one can simulate threshold gates, by the use of less expressive gates and limited non-determinism.

Theorem 1.14. $\cap_{\epsilon>0}\, \mathsf{prSV}^{n^\epsilon}\text{-}\mathsf{ACC}$ *can't simulate* NE.

For E^{NP} we get a variety of lower bounds where the amount of non-determinism increases as we go down from ACC to k-CNF.

Theorem 1.15. $\cap_{\epsilon>0}\, \mathsf{prSV}^{n/(\log n)^\epsilon}\text{-}\mathsf{AC}^0$, $\cap_{\epsilon>0}\, \mathsf{prSV}^{\epsilon n/(\log n)^2}\text{-}\mathsf{k\text{-}CNF}$, *or* $\cap_{\epsilon>0}\, \mathsf{prSV}^{\epsilon n}\text{-}\mathsf{AC}^0(n)$, *can't simulate* E^{NP}.

One can directly get the lower bound $\mathsf{E}^{\mathsf{NP}} \not\subset \cap_{\epsilon>0} \mathsf{N}^{n^{\epsilon}}$-$\mathsf{ACC}$ by using Williams sub-exponential size ACC lower bound. But this direct approach won't work for lower bounds against sub-linear non-determinism, and for NE lower bounds (see the next section for full details). Our lower bounds follow from a more general connection that we build by extending Williams's connection.

THEOREM 1.16. *For super-polynomial function sp and $s(n) \leq O(n)$:*
(1) *an* $\mathsf{NTIME}(2^{n-s(n)^c}/sp(n))$ *C-tautology algorithm for every $c > 0$ implies* $\mathsf{NE} \not\subset \mathsf{prSV}^{s(n)}$-$C$
(2) *an* $\mathsf{NTIME}(2^{n-3s(n)}/sp(n))$ *C-tautology algorithm implies* $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{prSV}^{s(n)}$-$C$

We also extend Santhanam's [79] lower bound against fixed-polynomial size deterministic circuits to prSV circuits.

THEOREM 1.17. $\forall k \geq 1$ $\mathsf{prAM} \not\subset \mathsf{prSVSIZE}(n^k)$ *and* $\forall k \geq 1$ $\mathsf{AM}/\omega_n(1) \not\subset \mathsf{prSVSIZE}(n^k)$.

We also extend Williams's connection between non-trivial GAP-SAT algorithm and NEXP lower bounds. GAP-SAT is the promise problem, where the positive inputs are tautology circuits, and negative inputs are s-size circuits that have at most $2^n(1-1/s)$ satisfying assignments. Note that, a tautology or a CAPP algorithm also imply a GAP-SAT algorithm. CAPP is the problem of computing the acceptance probability of s-size circuits within an additive error of $\pm 1/s$.

THEOREM 1.18. *An* $\mathsf{NTIME}(2^n/sp(n))$ GAP-SAT *algorithm for n-input polynomial-size* $(\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP})$-*oracle circuits, for any super-polynomial function $sp(n)$, implies* $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP})/poly$.

As there is no complete language in $\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP}$, we need a (possibly different) non-trivial algorithm for each $\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP}$ oracle. Note that, for any $A \in \mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP}$, tautology (or CAPP) for poly-size A-oracle circuits has a trivial algorithm that runs in non-deterministic $poly(n)2^n$-time: for all 2^n inputs, non-deterministically guess the answers to all the oracle queries, and guess their certificates (for A for any positive answer, for $\overline{A}$ for any negative answer).

One can also view $\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP}$ as $\mathsf{P}^{\mathsf{NP} \cap \mathsf{Co}\text{-}\mathsf{NP}}$. In this view, our result works separately for any one $\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP}$ oracle A: a non-trivial GAP-SAT algorithm for A-oracle circuits implies $\mathsf{NEXP} \not\subset \mathsf{P}^A/poly$. So our result is essentially a relativized version of Williams's result upto $(\mathsf{NP} \cap \mathsf{Co}$-$\mathsf{NP})$ oracles.

1.4 Our techniques, interesting by-products, and previous work

Lower bounds from Karp-Lipton Theorems and fast tautology algorithms:

Previous work: The idea of fast algorithms to lower bounds, can be traced back to the first paper where the non-uniform class $\mathsf{P}/poly$ was discussed (by Karp and Lipton [55]), where one of the corollaries (credited to Meyer) is that $\mathsf{P} = \mathsf{NP} \implies \mathsf{EXP} \not\subset \mathsf{P}/poly$. This can be interpreted as: a polynomial time algorithm for Ckt-SAT (or any other NP-complete problem), implies $\mathsf{EXP} \not\subset \mathsf{P}/poly$. The connection in [55] was established by first establishing $\mathsf{EXP} \subset \mathsf{P}/poly \implies \mathsf{EXP} = \Sigma_2^{\mathsf{P}}$. Any such non-uniform to uniform collapse is famously called Karp-Lipton Theorem (or KLT for short) in the literature. The assumed fast SAT algorithm, along with the assumption $\mathsf{EXP} \subset \mathsf{P}/poly$, implies $\mathsf{EXP} = \Sigma_2^{\mathsf{P}} = \mathsf{P}$ and thus contradicts the deterministic time hierarchy [36, 38].

A similar KLT was proved for NEXP in [41]. Similar KLTs have also been proved for $\mathsf{EXP}^{\mathsf{NP}}$ [14], PSPACE [9], and related classes [28, 40]. Using the NEXP KLT, an $\mathsf{NSUBEXP}$ tautology algorithm contradicts the non-deterministic time hierarchy [26, 32, 51, 83] by showing $\mathsf{NEXP} = \Sigma_2^{\mathsf{P}} = \mathsf{NSUBEXP}$. The collapse was extend from $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ to MA in [9]. This collapse yields $\mathsf{NEXP} \not\subset \mathsf{P}/poly$ from an $\mathsf{NSUBEXP}$ CAPP algorithm.

Our work: As a starting point for our connection between fast algorithms and UEXP lower bounds, we design similar KLTs for the intermediate classes: $\widetilde{\mathsf{FewEXP}}$, UEXP, and $\mathsf{UEXP} \cap \mathsf{Co}$-$\mathsf{UEXP}$.

THEOREM 1.19. *For any $k \geq 1$:*

(1) $\widetilde{\mathsf{FewE}}/n \subset \mathsf{SIZE}(n^k) \implies \widetilde{\mathsf{FewEXP}} = \mathsf{MA}$
(2) $\mathsf{UEXP} \subset \mathsf{SIZE}(poly) \implies \mathsf{UEXP} = \mathsf{MA}$
(3) $\mathsf{UEXP} \cap \mathsf{Co\text{-}UEXP} \subset \mathsf{SIZE}(poly) \implies \mathsf{UEXP} \cap \mathsf{Co\text{-}UEXP} = \mathsf{MA}$

Unfortunately our KLTs doesn't give the desired lower bound from USUBEXP, or even SUBEXP algorithms. This is because the first quantifiers of Σ_2^{P} and MA are not unambiguous. In any case, SUBEXP tautology or CAPP algorithms are far from the reach of current algorithm design techniques. Soon we'll see that USUBE algorithms are sufficient for UEXP lower bounds.

Easy-witness Lemmas:

Previous work: Williams's work [98] show that even a small progress in SAT and CAPP algorithms will prove super-polynomial NEXP lower bounds. His connection is tighter because he uses the easy-witness lemma (or EWL for short) from [41], which was the core of the collapse of NEXP to EXP in the NEXP KLT. The easy-witness technique was first introduced in [52]. We also derive analogous EWL for UEXP while establishing the UEXP KLT.

EWL for NEXP states that, if NEXP has $\mathsf{P}/poly$ circuits, then any NEXP verifier V, for every input x that it accepts, has a certificate y with $V(x, y) = 1$ that is encoded by the truth-table of a $\mathsf{P}/poly$ circuit (when the truth-table is seen as a concatenation of the 2^n outputs). In short, if NEXP has small circuits, than all NEXP verifiers have small witnesses (i.e., witness encoded by small circuits). The proof is given by contradiction: if NEXP witnesses have high circuit complexity, then they can be used to derandomize MA and contradict a certain lower bound.

Our techniques: We prove same connection between the circuit complexity of $\widetilde{\mathsf{FewEXP}}$, UEXP, and $\mathsf{UEXP} \cap \mathsf{Co\text{-}UEXP}$, and the witness complexity for corresponding verifiers. Our EWL for $\widetilde{\mathsf{FewEXP}}$ uses the technique from [41], and exploits the fact that derandomization of MA only requires limited non-determinism. For UEXP and $\mathsf{UEXP} \cap \mathsf{Co\text{-}UEXP}$ EWLs, we use a simpler technique that gives even stronger results. We show that a specific version of search problem (that searches for the lexicographically smallest certificate) for UEXP verifiers lies within UEXP itself. The circuit complexity of that search version is same as the circuit complexity of witnesses for UEXP verifiers, so we directly get the EWL from this reduction. Note that, such reductions are not possible for NEXP verifiers unless $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{NEXP}$ [46].

Lower bounds from Easy-witness Lemmas and fast tautology algorithms:

Previous work: Even in his approach with NEXP EWL, Williams proves his result by contradicting the non-deterministic hierarchy. He only needs a $\mathsf{NTIME}(2^n/sp(n))$ tautology algorithm for n-input polynomial-size circuits, for any super-polynomial function sp. This is a huge improvement over the NSUBEXP tautology algorithm, that was required if one uses the NEXP KLT directly.

Our work and technique: We use Williams's framework and get UEXP lower bounds from USUBE ($\cup_{\epsilon>0}\mathsf{UTIME}(2^{\epsilon n})$) algorithm by using our UEXP EWL. The main reason why we need faster algorithms is that we don't have any hierarchy for UTIME that is as good as the hierarchies for NTIME. Note that, before this no UEXP lower bound was obtained, even from a deterministic SUBEXP tautology algorithm. Although our approach requires a canonization algorithm too, for unambiguously guessing the witness circuits, we get rid of this requirement for the case of $\widetilde{\mathsf{FewEXP}}$ lower bounds.

Related work: Similar results were also proved for BPEXP [22], where they require randomized tautology algorithms (with two-sided error). They require algorithms to run in $2^{n/(\log n)^{\omega(1)}}$, which is slightly faster than SUBE. Although, they get sharper lower bounds than $\mathsf{BPEXP} \not\subset \mathsf{P}/poly$, namely $\mathsf{BPE} \not\subset \mathsf{SIZE}(n(\log n)^{O(1)})$, and only require the fast algorithm to run for quadratic-size circuits, BPEXP is incomparable to NEXP. Only a fast randomized algorithm (with zero-sided error) to ZPEXP lower bound, or a fast randomized algorithm (with one-sided error) to REXP lower bound, would be considered a strict improvement over Williams's connection for NEXP.

In [68], they extended Williams's connection to NQP (non-deterministic quasi-polynomial time). There are two major differences with our work. First, the NQP lower bounds require fast algorithms for circuits with size sub-exponential or higher, whereas our results only need fast algorithms for polynomial-size circuits. Second, there is no known comparison between NQP and $\widetilde{\mathsf{Few}}\mathsf{EXP}$. Even if we try to distribute all non-deterministic branches of an NQP verifier within exponential time, the new EXP verifier will have more branches that $2^{n^{O(\log n)}}$ (also note that, our results would go through even for slightly strict definition of $\widetilde{\mathsf{Few}}$).

Super-linear lower bounds from fast $\Pi_2\mathsf{SAT}$ algorithms:

Previous work: In [22] they also showed that a randomized algorithm for $\Pi_2\mathsf{SAT}$ with linear-clauses that runs in $2^{n/(\log n)^{\omega(1)}}$ time implies $\mathsf{BPE} \not\subset \mathsf{SIZE}(n(\log n)^{O(1)})$.

Our work: We show an analogous result for unambiguous non-deterministic time. We show that, a $\mathsf{UTIME}(2^{n/(\log n)^{\omega(1)}})$ algorithm for $\Pi_2\mathsf{SAT}$ with linear clauses implies $\mathsf{UE}/n \not\subset \mathsf{SIZE}(n(\log n)^{O(1)})$.

Our technique: We get this result by genaralizing Williams's connection, whereas in [22] they use altogether different techniques. Note that, one can't directly get such connections between non-deterministic $\Pi_2\mathsf{SAT}$ algorithms and NE lower bounds, because the NEXP EWL is not as fine-grained as our UEXP EWL.

Lower bounds from fast learning algorithms:

Previous work: The two commonly studied learning models are: "the Angluin's exact learning model" [6], and "the Valiant's PAC model" [92]. Fast learning algorithms in these models have been known to yield lower bounds [30, 35, 57, 72, 73].

In [30] it was implicit that, if a circuit (concept) class C is exact learnable in $\mathsf{SUBEXP}^{\mathsf{NP}}$, then $\mathsf{EXP}^{\mathsf{NP}} \not\subset C$. In [35, 57] it was improved to: SUBEXP learning algorithm implies $\mathsf{EXP} \not\subset C$. In [72] it was implicit that: NSUBEXP learning algorithm (where on any input circuit regardless of its size, there is one branch where the algorithm outputs a hypothesis, and the hypothesis is guaranteed to be correct only for polynomial size circuits) implies $\mathsf{NEXP} \not\subset C$. In [73] it was shown that if C admits $2^n/n^{\omega(1)}$ randomized (weak) learning algorithm (with membership queries), then $\mathsf{BPEXP} \not\subset C$.

Our work: We show that unambiguous learning algorithm implies UE lower bounds. As far as we know this is the first lower bound result from unambiguous learning. Our results are weak in sense that: the lower bound requires UE to use $O(n \log n)$ bits of advice, we also need a tautology algorithm to assist the learning, and our algorithm only makes membership queries to polynomial-size circuits (it can make queries to sub-exponential size too, but then we would require fast tautology algorithm for sub-exponential size circuits). Even with these small weaknesses, our connection is not implied by any of the previous known connections, and our lower bound is strictly better than $\mathsf{EXP}^{\mathsf{NP}} \not\subset C$ and $\mathsf{NEXP} \not\subset C$, and incomparable to $\mathsf{BPEXP} \not\subset C$. For the $\mathsf{EXP} \not\subset C$ lower bound, they used a clever diagonalization argument in [57] that directly doesn't work for UTIME learning (and also NTIME learning, because it's not clear how the diagonalization process will beat all the non-deterministic branches for all the C circuits).

Our Technique: Unlike other results, we use Williams's framework itself, and simulate canonization using a learning algorithm that is assisted with a tautology algorithm.

Avoiding the Natural Proofs barrier:

Our work and technique: We avoid the barrier by taking the route $\mathsf{NEXP} \supseteq \mathsf{UEXP} \supseteq \mathsf{EXP}$, and hit the much safer unique properties. Using our UEXP EWL we prove equivalence between UEXP lower bounds and natural properties that are not large, instead are unique. EXP lower bounds were known to yield P computable unique properties, our work extends this to UEXP ∩ Co−UEXP lower bounds, and also establish the equivalence.

Related work: We put an upper bound on the number of non-deterministic branches that accept. If one wants to start by putting an lower bound, they would have to take the NEXP ⊇ REXP ⊇ EXP route. Williams work [100] also indicates that this other direction would not be feasible, if one wants to use the easy-witness technique, as certain lower bounds for witnesses of REXP and ZPEXP are equivalent to the existence of natural properties.

Other related work [24] that talks about properties sparser than natural properties, shows that under the same cryptography primitives that indicates the non-existence of natural properties, there is a property with density $1/2^{n^{(\log n)^{O(1)}}}$ (vs. density $1/2^{O(n)}$ of a natural property) against $\mathsf{P}/poly$, that establishes $\mathsf{NP} \not\subset \mathsf{P}/poly$. In [21] they give hope of avoiding the Natural Proofs barrier by establishing equivalence between $\mathsf{NP} \not\subset \mathsf{P}/poly$ and natural properties that accept SAT and are useful against only those polynomial-size circuits that never error on SAT.

Lower bounds to derandomization:

Previous work: Relationship between derandomization and uniform/non-uniform lower bounds has been studied extensively in the past [8, 9, 19, 41, 47, 48, 70, 71, 88, 91, 100]. In our results we only focus on the lower end of this spectrum, i.e. derandomization that requires sub-exponential time. Note that, BPP is a sub-class of $\mathsf{P}/poly$, and we also don't know if $\mathsf{NEXP} \neq \mathsf{BPP}$.

First in a series of work [9, 70, 71] non-uniform lower bounds were shown to yield derandomization of the class BPP. In the lower end of the spectrum it was shown that $\mathsf{EXP} \not\subset \mathsf{P}/poly$ implies $\mathsf{BPP} \subset$ io-SUBEXP. Later in [47] this connection was extended to the uniform lower bound $\mathsf{EXP} \neq \mathsf{BPP}$. This lower bound was actually shown to be equivalent to $\mathsf{BPP} \subset$ io-Heur-SUBEXP. In [41, 100] these connections and equivalences were extended to NEXP and REXP lower bounds, and derandomization that works in NSUBEXP and ZPSUBEXP.

Our work: We extend these connections to UEXP and UEXP ∩ Co−UEXP lower bounds, and derandomization that works in USUBEXP.

Our technique: We use our connections between UEXP lower bounds and unique properties. We only get the lower bounds to derandomization connections, and not the reverse connections. It is due to the lack of complete languages and strong hierarchies for UEXP.

Unconditional super-polynomial lower bounds:

Previous work: Although, TC^0 lower bounds are still untouchable, and at this point we don't even have NEXP lower bound for depth-two circuits with linear-threshold gates, some improvements have been made after Williams's ACC lower bound. In [101] it was shown that NEXP doesn't have ACC circuits where the bottom most layer is allowed to have linear-threshold gates. In [97] the circuit class was further generalized by allowing the top gate to be any sparse symmetric function (exact majority is one example).

Our work: We extend the NEXP lower bound against ACC by allowing the circuits to use sub-polynomial amount of prSV non-determinism. For the case of E^{NP}, our lower bounds allow almost sub-linear amount of non-determinism as we go down to non-uniform k−CNF circuits. Recall that, even k−CNF circuits are very powerful with such type of non-determinism (Theorem 1.13), and if we increase the amount to linear we will get super-linear lower bounds for E^{NP}.

Our techniques: One can directly get the lower bounds against $\cap_{\epsilon>0}\ \mathsf{N}^{n^{\epsilon}}$-ACC for E^{NP} and NEXP witnesses, by using Williams sub-exponential size ACC lower bound. Any $\mathsf{N}^{n^{\epsilon}}$-ACC circuit can be converted to a sub-exponential size ACC circuit by OR-ing over all the non-deterministic inputs. Williams used his fast ACC algorithm for sub-exponential size in his result.

There are two drawbacks of this direct approach. First, it requires a very large size. For instance, for sub-linear non-deterministic inputs, the size of the resultant deterministic circuit is $\cap_{\epsilon>0}\mathsf{SIZE}(2^{\epsilon n})$. Second, even if we have fast algorithms for these large circuits, the lower bounds don't transfer to NEXP. The fast algorithms can only give lower bounds for NEXP witnesses, or E^{NP}. This is because the NEXP EWL is not that fine-grained.

So we design an NEXP EWL for prSV non-deterministic circuits. Our EWL also works for the case of limited non-determinism. As far as we know, this is the first EWL that talks about any kind of non-deterministic circuits. We use our EWL and combine the non-deterministic and co-non-deterministic circuits for NEXP witnesses in a clever way, to yield $\mathsf{NEXP} \not\subset \cap_{\epsilon>0}\, \mathsf{prSV}^{n^\epsilon}\mathsf{ACC}$. Since our technique only requires fast algorithms for polynomial-size circuits, we also get lower bounds with sub-linear non-determinism, for E^{NP} against circuit classes lower than ACC.

Previous work: Note that, there was already an NEXP KLT for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$ which gives $\mathsf{NEXP} = \mathsf{AM}$ [95], and can be extended to $\mathsf{NEXP} = \mathsf{MA}^{\mathsf{NP}\cap\mathsf{Co\text{-}NP}}$ using results from [16]. This KLT implies that an NSUBEXP tautology algorithm would yield $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. But we use the EWL to derive lower bounds from much slower algorithms.

Extension of NEXP KLT and new gap theorems:

Our work: While deriving the EWL we also prove the converse of this KLT and extend it to the class $\mathsf{EXP}^{\mathsf{NP}}_{||}$, where the subscript '||' means that the algorithm is only allowed to make non-adaptive queires. This extension also applies to the NEXP KLT for $\mathsf{P}/poly$. The equivalence of non-uniform lower bounds for NEXP and $\mathsf{EXP}^{\mathsf{NP}}_{||}$ were already known (attributed to Buhrman in [29]). Our result proves an equivalence between uniform lower bounds, and thus results in a better gap theorem for MA than what was previous known [41]. We also get similar gap theorem for $\mathsf{MA}^{\mathsf{NP}\cap\mathsf{Co\text{-}NP}}$. As $\mathsf{MA} \subseteq \mathsf{MA}^{\mathsf{NP}\cap\mathsf{Co\text{-}NP}} \subseteq \mathsf{AM}$, this can be seen as an intermediate step for proving a gap theorem for AM.

Using the EWL we also get a gap/speed-up theorem for CAPP for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$-oracle circuits. We first extend Williams connection to show that, non-trivial CAPP algorithm for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$-oracle circuits imply $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. As this lower bound is equivalent to NSUBEXP CAPP algorithm (that works infinitely often, and uses sub-polynomial advice), we get that non-trivial savings in CAPP for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$-oracle circuits imply sub-exponential savings.

Our technique: In the extension of the KLT to $\mathsf{EXP}^{\mathsf{NP}}_{||}$, in some sense we derive an EWL for $\mathsf{EXP}^{\mathsf{NP}}_{||}$, where the witness circuit captures all the NP-oracle queries on all n-length inputs.

Unconditional fixed-polynomial size lower bounds:

Previous work: This work of fixed-polynomial size lower bounds was started by Kannan in 1982 [54]. He used the low-end KLT for NP, which collapses the polynomial heirarchy to $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ [55], to prove fix-polynomial lower bounds for $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$. Better lower bounds were proved using improved low-end KLTs [12, 15, 56, 62, 94], before Santhnam [79] gave the lower bound for MA/1 and prMA, using the high-end KLT for PSPACE from [9].

For non-deterministic and SV non-deterministic circuits, the best high-end KLT was given in [8], which collapses PSPACE to M(AM||Co-NP). In [87] this KLT was used to give fixed-polynomial lower bounds for prM(AM||Co-NP) against non-deterministic and SV non-deterministic circuits. The class M(AM||Co-NP) lies in the third-level of the polynomial hierarchy and contains AM and Σ_2^{P}.

Our work and technique: We establish fixed-polynomial lower bounds for prAM against prSV non-deterministic circuits. Note that, in this lower bound, the class has to have one language that beats all of the $(\mathtt{NTIME}(n^k) \cap \mathtt{Co\text{-}NTIME}(n^k))/n^k$ algorithms for each $k \geq 1$. The main technical difficulty in proving this lower bound was the lack of complete problems for $\mathtt{NTIME}(n^k) \cap \mathtt{Co\text{-}NTIME}(n^k)$. Prior to our result, the best known lower bounds for prAM were against fixed-polynomial deterministic circuits.

1.5 Organization of the paper

In the Section 2 we discuss and define all the technical definitions that we use. In the Section 3 we derive the EWL and KLT for UEXP and related classes. In the Section 4 we derive the connections between fast unambiguous circuit analysis algorithms and circuit lower bounds. In the Section 5 we derive connections between unique properties and UEXP lower bounds, derive fast unambiguous derandomization results from UEXP lower bounds. In the Section 6 we give all the results regarding the prSV non-deterministic circuits. Finally in the Section 7 we give concluding remarks and discuss some open problems.

2 PRELIMINARIES

Basic notations: Unless a new range is declared during the usage, we use t for time-constructible functions $n \leq t(n) \leq 2^{n^{O(1)}}$, a for advice functions $0 \leq a(n) \leq poly(n)$, s for circuit sizes (number of wires) $n \leq s(n) \leq 2^n$. For language L we use $L_n = \{x \mid x \in L \wedge |x| = n\}$ to denote the n^{th}-slice of L (or the characteristic function of L on n-length inputs). For circuit C, we use $tt(C)$ to denote its truth-table, and $|C|$ 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, MA, AM, $\Sigma_2^{\mathsf{P}}, \Pi_2^{\mathsf{P}}$, PH (see [7]) and their corresponding complexity measures, DTIME, NTIME, RTIME, UTIME, BPTIME, ZPTIME, MATIME, AMTIME, Σ_2TIME, Π_2TIME. For the special cases of $\Sigma_2^{\mathsf{P}}, \Pi_2^{\mathsf{P}}$, we omit the superscript P and simply write Σ_2, Π_2. For C = D, N, R, U, BP, ZP, MA, AM, Σ_2, Π_2: $\mathtt{CTIME}(t)$ denotes the class of languages accepted by CTIME machines that run in $O(t)$ time. CE, CEXP, CSUBE, CSUBEXP, denote the classes $\cup_{c\geq 0}\mathtt{CTIME}(2^{cn})$, $\cup_{c\geq 0}\mathtt{CTIME}(2^{n^c})$, $\cap_{c\geq 0}\mathtt{CTIME}(2^{cn})$, $\cap_{c\geq 0}\mathtt{CTIME}(2^{n^c})$ respectively. We assume familiarity with Ckt-SAT (circuit satisfiability), Ckt-TAUT (circuit tautology), k-SAT, k-TAUT, CNF-SAT, DNF-TAUT, Σ_2-SAT and Π_2-SAT.

Zero-error classes: We extend the concept of zero-error class to non-deterministic and unambiguous classes. We do this for the sake of clarity in certain arguments, and specially for distinguishing between certain non-uniform classes.

$L \in \mathtt{ZCTIME}(t)$ if there exists a Turing machine M, that for input (x, y) with $|x| = n$ and $|y| = c \cdot t(n)$ for some constant c, runs in time $c \cdot t(n)$ for $\forall n \in \mathbb{N}$, and whose output lies in $\{1, ?\}$ if $x \in L$, and in $\{0, ?\}$ if $x \notin L$. Additionally, the quantity $\Sigma_{y:M(x,y)\in\{0,1\}}1$ is equal to: 1 for C = U (*uniqueness*), $\geq \frac{1}{2} \times 2^{c \cdot t(n)}$ for C = R (*largeness*), ≥ 1 for C = N (*existence*). The verifier/predicate corresponding to M is called zero-error non-deterministic. For the special cases of C = U and C = R, its called zero-error unambiguous and zero-error randomized respectively.

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

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 $\{\mathsf{AC_0}, \mathsf{ACC_0}, \mathsf{TC_0}, \mathsf{NC_1}, \mathsf{NC}, \mathsf{P}/poly\}$. 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, i.e. the minimum size of any $\mathcal{C}$ circuit C whose truth-table (when concatenated to make the string

$C(00\ldots0)\ldots C(11\ldots1))$ is tt. In the case of unrestricted Boolean circuits, instead of $C(s)$ and $ckt_C(tt)$, we use $\mathtt{SIZE}(s)$ and $ckt(tt)$ respectively. $ckt^M(tt)$ denotes the minimum size of any M-oracle circuit whose truth-table is tt.

Non-uniform classes: $L \in \Gamma/a$ if there exists an advice-taking Γ Turing Machine M, and advice sequence $\{a_n\}_{n\in\mathbb{N}}$ satisfying $\forall n\ |a_n| = a(n)$, such that: $x \in L \iff M(x)/a_{|x|} = 1$. For semantic classes, the machine M only needs to satisfy the semantic promise on the correct advice sequence $\{a_n\}_{n\in\mathbb{N}}$ (and not on all advice sequences). Below we define an exception for $\mathtt{C} = \mathtt{N}, \mathtt{R}, \mathtt{U}$:

(1) $L \in (\mathtt{CTIME}(t) \cap \mathtt{Co\text{-}CTIME}(t))/a$: If there are $\mathtt{NTIME}(t)$ Turing Machines M and M', and advice sequence $\{a_n\}_{n\in\mathbb{N}}$ satisfying $\forall n\ |a_n| = a(n)$, such that: (i) $x \in L \iff M(x)/a_{|x|} = 1$; (ii) both M and M' satisfy the semantic promise on $\{a_n\}_{n\in\mathbb{N}}$ (for $\mathtt{C} = \mathtt{N}$ there is no promise); and (iii) both accept complement languages. For the other advice sequences, M and M' are not required to satisfy the semantic promise, but are required to accept complement languages.
(2) $L \in \mathtt{ZCTIME}(t)/a$: It's the same as (1), except that in the "other advice sequences" part, M and M' are not required to accept complement languages. That is, both M and M' are required to accept complement languages just for some correct advice sequence, and simultaneously satisfy the semantic promise. Equivalently, there is a $\mathtt{ZCTIME}(t)$ Turing Machine N, and advice sequence $\{a_n\}_{n\in\mathbb{N}}$ satisfying $\forall n\ |a_n| = a(n)$, such that: $x \in L \iff N(x)/a_{|x|} = 1$; and N satisfy the semantic promise. For other advice sequences N might: (i) fail to provide uniqueness/largeness/existence; (ii) for some input, output both 0 and 1 (on different non-deterministic branches); or (iii) for some input, output just '?' (on all non-deterministic branches).
(3) $L \in \mathtt{CTIME}(t)/a \cap \mathtt{Co\text{-}CTIME}(t)/a$: It's further relaxed than (2). We need two advice sequences $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$, satisfying $\forall n\ |b_n| = b(n)$ and $\forall n\ |b_n| = b(n)$ (these sequences need not be the same). M satisfy the semantic promise on $\{a_n\}_{n\in\mathbb{N}}$ and accept L. M' satisfy the semantic promise on $\{b_n\}_{n\in\mathbb{N}}$ and accept $\overline{L}$. There are no other conditions.

Remark: $L \in \mathtt{ZCTIME}(t)/a$ is equivalent to L having $\mathtt{CTIME}(t)/a$ and $\mathtt{Co\text{-}CTIME}(t)/a$ algorithms that both use the same advice. This shows:

$$(\mathtt{CTIME}(t) \cap \mathtt{Co\text{-}CTIME}(t))/a \subseteq \mathtt{ZCTIME}(t)/a \subseteq \mathtt{CTIME}(t)/a \cap \mathtt{Co\text{-}CTIME}(t)/a \subseteq \mathtt{ZCTIME}(t)/2a$$

So the difference between $\mathtt{ZCTIME}(t)/a$ and $\mathtt{CTIME}(t)/a \cap \mathtt{Co\text{-}CTIME}(t)/a$ only matters when the amount of advice is precise.

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

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

Promise classes: A promise problem $\Pi = (\Pi_Y, \Pi_N)$ is a pair of disjoint sets Π_Y and Π_N. In the special case where $\Pi_Y \cup \Pi_N = \{0,1\}^*$, Π is also a language. We say that a language L agrees with Π if, $x \in \Pi_Y$ implies $x \in L$, and $x \in \Pi_N$ implies $x \notin L$. $\mathtt{pr}\Gamma$ for any semantic class Γ, is the class of problems that have Γ algorithms for promise inputs (that may not satisfy the semantic promise on other inputs). Lower and upper bounds for promise classes are only defined on promise inputs.

Variety of witness complexities for $\mathtt{CTIME}$: We define different ways of measuring complexity of witnesses for non-deterministic verifiers, that has been used in the literature.

(1) **Witnesses:** A non-deterministic verifier V for L, has witnesses in s-size C circuits, if for every $x \in L$, there is an $s(|x|)$-size C circuit C_x, such that $V(x, tt(C_x)) = 1$. If V uses a amount of advice, then we say that V/a has witnesses in s-size C circuits, if for some correct advice sequence $\{a_n\}_{n\in\mathbb{N}}$ satisfying $\forall n\ |a_n| = a(n)$, for every $x \in L$, there is an $s(|x|)$-size C circuit C_x, such that $V(x, tt(C_x))/a_{|x|} = 1$.
(2) **Hitting-sets for witnesses (all witnesses in one):** A non-deterministic verifier V for L has l-size hitting-sets in s-size C circuits, if $\forall n \in \mathbb{N}$, there is an $s(n)$-size C circuit C_n such that $tt(C_n)$ when partitioned into l strings $\{str_1, \dots, str_l\}$ of equal lengths, satisfies $\forall(x : |x| = n \wedge x \in L)\ \exists(i \in [1, l])\ V(x, str_i) = 1$. The default value of l is 2^n. If V uses advice, hitting-sets are defined analogous to witnesses in the advice setting.
(3) **Oblivious witnesses (ordered hitting-sets for witnesses):** Let $y_1, \dots, y_{2^n}$ denote the n-length strings arranged in the lexicographical order. A non-deterministic verifier V for L has oblivious witnesses in s-size C circuits, if $\forall n \in \mathbb{N}$, there is an $s(n)$-size C circuit C_n such that $tt(C_n)$ when partitioned into 2^n strings $\{str_1, \dots, str_{2^n}\}$ 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. If V uses advice, oblivious witnesses are defined analogous to the witnesses in advice setting.

Variety of seed complexities for `ZCTIME`**:** Any language in `ZCTIME` has two, a `CTIME` algorithm and a `Co-CTIME` algorithm deciding it. So instead of witnesses, we define a stronger notion: seeds, which is nothing but a technical way of combining witnesses from the two algorithms.

(1) **Seeds:** A zero-error non-deterministic verifier V for L has seeds in s-size C circuits, if for every x, there is an $s(|x|)$-size C circuit C_x, such that $V(x, tt(C_x)) \in \{0, 1\}$. If V uses a amount of advice, then we say that V/a has seeds in s-size C circuits, if for some correct advice sequence $\{a_n\}_{n\in\mathbb{N}}$ satisfying $\forall n\ |a_n| = a(n)$, for every x, there is an $s(|x|)$-size C circuit C_x, such that $V(x, tt(C_x))/a_{|x|} \in \{0, 1\}$.
(2) **Hitting-sets for seeds (all seeds in one):** A zero-error non-deterministic verifier V for L has l-size hitting-sets in s-size C circuits, if $\forall n \in \mathbb{N}$, there is an $s(n)$-size C circuit C_n such that $tt(C_n)$ when partitioned into l strings $\{str_1, \dots, str_l\}$ of equal lengths, satisfies $\forall(x : |x| = n)\ \exists(i \in [1, l])\ V(x, str_i) \in \{0, 1\}$. The default value of l is 2^n. If V uses advice, hitting-sets are defined analogous to seeds in the advice setting.
(3) **Oblivious seeds (ordered hitting-sets for seeds):** Let $y_1, \dots, y_{2^n}$ denote the n-length strings arranged in the lexicographical order. A zero-error non-deterministic verifier V for L has oblivious seeds in s-size C circuits, if $\forall n \in \mathbb{N}$, there is an $s(n)$-size C circuit C_n such that $tt(C_n)$ when partitioned into 2^n strings $\{str_1, \dots, str_{2^n}\}$ of equal lengths, satisfies $\forall(i \in [1, 2^n])\ V(y_i, str_i) \in \{0, 1\}$. If V uses advice, oblivious seeds are defined analogous to seeds in the advice setting.

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

Definition 2.1 (Useful uniform properties). A uniform 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 of Boolean functions). $\mathcal{A}$ is said to be useful against s-size C circuits if it satisfies the second condition stated below.

(1) *Size restrictions:*
  (a) *Uniqueness for* $\mathsf{C} = \mathsf{U}$*:* $\forall n \in \mathbb{N}\ \Sigma_{x:|x|=2^n \wedge \mathcal{A}(x)=1} 1 = 1$
  (b) *Largeness for* $\mathsf{C} = \mathsf{R}$*:* $\forall n \in \mathbb{N}\ Pr_{x:|x|=2^n}[\mathcal{A}(x) = 1] \geq \frac{1}{2^n}$
  (c) *Existence for* $\mathsf{C} = \mathsf{N}$*:* $\forall n \in \mathbb{N}\ \Sigma_{x:|x|=2^n \wedge \mathcal{A}(x)=1} 1 \geq 1$

(2) *Usefulness:* for infinitely many $n \in \mathbb{N}$, $\forall(x : |x| = 2^n)$ $\mathcal{A}(x) = 1 \implies ckt_C(x) > s(n)$

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

Definition 2.2 (Useful properties that use advice). A Γ/a algorithm $\mathcal{A}$ is a Γ/a-C property if it satisfies the first condition of Definition 2.1 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 C circuits if it satisfies the second condition of Definition 2.1 on the same advice sequence $\{a_n\}_{n\in\mathbb{N}}$.

Lower bounds against $\mathsf{prSV}\Sigma_i$ circuits:
The promise $\mathsf{SV}\Sigma_i$ circuits need special care in regards of lower bounds, as we need to deal with each prSV algorithm separately. For any class Γ, we define the following upper bounds and lower bounds against prSV circuits:

- Let $\mathcal{A}$ be a prSV algorithm. For any 2^n-length truth-table tt, we use $ckt_{\mathsf{SV}(\mathcal{A})}(tt)$ to denote the minimum size $s(n)$ such that: $\mathcal{A}$ outputs C with $tt(C) = tt$ on an $s(n)$-size input. We use $ckt^f_{\mathsf{prSV}}(tt)$ to denote the minimum size $s(n)$ such that: any prSV algorithm with description length at most $f(n)$, outputs C with $tt(C) = tt$ on an $s(n)$-size input.
- $\Gamma \not\subset \mathsf{prSVSIZE}(s)$: There is an $L \in \Gamma$, such that for every prSV algorithm $\mathcal{A}$, there is an infinite subset $S \subset \mathbb{N}$ of input lengths, such that $n \in S \implies \forall x : |x| = s(n)\ tt(\mathcal{A}(x)) \neq L_n$.
- Γ has witnesses/seeds in $\mathsf{prSVSIZE}(s)$: For $L \in \Gamma$ and Γ verifier V for L, there is a prSV algorithm $\mathcal{A}$, there is an input sequence $\{x_{s(n)}\}_{n\in\mathbb{N}}$ such that $\forall n \in \mathbb{N}\ tt(\mathcal{A}(x_{s(n)}))$ is the 'oblivious witness / oblivious seed / hitting-set for witnesses / hitting-set for seeds' for V on n-length inputs. For the case of 'witnesses / seeds', $\forall n \in \mathbb{N}\ \forall y : |y| = n$ there is an input $x_{s(n)}$ such that $tt(\mathcal{A}(x_{s(n)})\}$ is the 'witness / seed' for V on input y.
- Γ doesn't have witnesses/seeds in $\mathsf{prSVSIZE}(s)$: There is an $L \in \Gamma$ and Γ verifier V for L, such that for every prSV algorithm $\mathcal{A}$, there is an infinite subset $S \subset \mathbb{N}$ of input lengths, such that $n \in S \implies \forall x : |x| = s(n)\ tt(\mathcal{A}(x))$ is not the 'oblivious witness / oblivious seed / hitting-set for witnesses / hitting-set for seeds' for V on n-length inputs. For the case of 'witnesses / seeds', $n \in S \implies \exists y : |y| = n\ \forall x : |x| = s(n)\ tt(\mathcal{A}(x))$ is not the ' witness / seed' for V on input y.

Hardness vs randomness: The process of using a function that is hard for a circuit class Λ (i.e. requires large size of Λ circuits) to yield a pseudo random generator (PRG) that fools Λ circuits (i.e. creates a sparse subset of inputs with roughly same fraction of inputs resulting in 1) is well known in the literature. A PRG G creates this sparse subset by mapping a small input length to the required larger output length (same as the input length of the circuit).

A PRG $G : s(n) \to n$ is computable in Γ means: the language $L_G = \{(s, i, b) \mid \text{the } i^{th}\text{-bit of } G(s) \text{ is } b\}$ is in Γ. Inputs to G are called seeds, and their size (here $s(n)$) is called the seed length of G.

A PRG G is fooling a circuit C means: the fraction of inputs from the $2^{s(n)}$ size image of G that C accepts, is same as the fraction of all the inputs that C accepts (within error $\pm 1/n$). We use the following theorem in all our derandomization results.

THEOREM 2.3. *[58, 71, 85, 91] There exists a universal constant g such that the following holds for any class O of oracles and oracle M, and any constants $\epsilon > 0$ and $d \geq 1$: if a Boolean function family $f = \{f_n\}_{n\in\mathbb{N}}$ computable in E^O that satisfies $\forall n \in \mathbb{N}\ ckt^M(f(n)) \geq n^{gd/\epsilon}$, then there exists a PRG family $G = \{G_n\}_{n\in\mathbb{N}}$ computable in E^O, such that $G_n : n^\epsilon \to n^d$ fools n^d-size B-oracle circuits. Moreover, if circuit lower bound holds infinitely often, then G fools circuits infinitely often.*

3 EWL AND KLT FOR UTIME AND RELATED CLASSES

We first give 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). Then we describe similar results for ZUTIME (Section 3.3) and $\widetilde{\texttt{Few}}$TIME (Section 3.4).

3.1 Search to decision reduction for UTIME

For $L \in$ NP and verifier V for L, there is a standard P^{NP} algorithm for the corresponding search problem. This algorithm implicitly decides the following language:

$$L_{ewl(V)} = \{(x,i) \mid \exists y\ [V(x,y) = 1 \wedge (i^{th}\text{-bit of } y \text{ is } 1) \wedge \forall (z <_{l.o.} y)\ V(x,z) = 0]\} \quad (1)$$

where $z <_{l.o.} y$ stands for "z is lexicographically smaller than y", and the subscript $ewl(V)$ in $L_{ewl(V)}$ stands for "easy-witness language for V".

So if P = NP, then $L_{ewl(V)} \in$ P. For $L \in$ NEXP and verifier V for L, such results are not known. In particular, it is not known whether NEXP = EXP yields an EXP algorithm for the corresponding search problem, let alone $L_{ewl(V)}$.

NEXP $\subset_w$ SIZE$(poly)$ yields EXP algorithms for the NEXP search problems, by a simple brute-force argument. It is known that NEXP $\subset_w$ SIZE$(poly)$ is equivalent to NEXP $\subset$ SIZE$(poly)$ [41, 100], and to NEXP = MA [41] (reverse implication was attributed to van Melkebeek). In [41] it was also shown that a weaker collapse, namely NEXP = AM, is sufficient to give EXP algorithms for NEXP search problems. This is the weakest collapse known so far.

From [46] we get: $\forall(L \in$ NEXP$)$ $\forall$ (NEXP verifier V for L) $L_{ewl(V)} \in$ EXP $\iff$ $\mathsf{EXP}^{\mathsf{NP}}$ = EXP.

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

Theorem 3.1. *For $L \in$ UTIME(t) and unambiguous verifier V for L, $L_{ewl(V)} \in$ UTIME$(t(n))$ (where n is the input size for L and not $L_{ewl(V)}$). Moreover if this statement is true for every $O(t)$-time non-deterministic verifier (ambiguous and unambiguous) for every* UTIME(t) *language, then* ZNTIME(t) = 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 $O(t(|x|))$.

The moreover part: For $L \in$ ZNTIME(t), let V_1 and V_0 be its NTIME(t) and Co-NTIME(t) verifiers respectively. Consider the UTIME(t) language $L' = \{0,1\}^*$.

Using V_1 and V_0 we first construct a verifier V' for L': if the first bit of the certificate is i, V' simulates V_i using the rest of the certificate.

Now using a UTIME$(t(n))$ algorithm $\mathcal{A}$ for $L'_{ewl(V')}$ we give a UTIME(t) algorithm for L: on input x, simulate $\mathcal{A}$ on $(x,1)$. Accept iff $\mathcal{A}$ accepts.

If $\mathcal{A}$ accepts then we know that $x \in L$ because there is no positive certificate for V' that starts with 0 (or in other words, no positive certificate for V_0). If $\mathcal{A}$ rejects, then $x \in L$ because there is a positive certificate for V' that starts with 0 (or in other words, a positive certificate for V_0).

Similarly, there is a UTIME(t) algorithm for $\overline{L}$, and thus $L \in$ ZUTIME(t). □

For the advice setting same proof goes through for the following adaptation of $L_{ewl(V)}$. For non-deterministic verifier V/a, that uses a amount of advice to decide a language L, for any correct advice sequence $\{a_n\}_{n\in\mathbb{N}}$:

$$L_{ewl(V/a)} = \{(x,i) \mid \exists y\ [V(x,y)/a_{|x|} = 1 \wedge (i^{th}\text{-bit of } y \text{ is } 1) \wedge \forall (z <_{l.o.} y)\ V(x,z)/a_{|x|} = 0]\} \quad (2)$$

Using this adaptation we get the following stronger corollary.

Corollary 3.2. *For $L \in \mathsf{UTIME}(t)/a$ and unambiguous verifier V/a for L, $L_{ewl(V/a)} \in \mathsf{UTIME}(t)/a$.*

3.2 EWL and KLT for UTIME

Using the search to decision reduction from Theorem 3.1 we derive EWL for unambiguous verifiers of languages in $\mathsf{UTIME}(t)$. 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 3.3. *For time-constructible $t \in 2^{O(n)}$, and constants c and k:*

(1) $\mathsf{UTIME}(t) \subset C(n^k) \implies \mathsf{UTIME}(t)$ *has oblivious witnesses in $C(n^k)$. Moreover if this statement is true for every $O(t)$-time non-deterministic verifier (ambiguous and unambiguous) for every $\mathsf{UTIME}(t)$ language, even just for witnesses (let alone oblivious-witnesses), then $\mathsf{ZNTIME}(t) \subseteq \mathsf{DTIME}(2^{n^{k+1}})$.*
(2) $\mathsf{UTIME}(t)/a \subseteq C(n^k) \implies \mathsf{UTIME}(t)/a$ *has oblivious witnesses in $C(n^k)$.*
(3) $\mathsf{UTIME}(2^{n^c})/a \subseteq C(n^k) \implies \mathsf{UTIME}(2^{n^c})/a$ *has oblivious witnesses in $C(n^{ck})$.*
(4) $\mathsf{UEXP}/a \subseteq \mathsf{SIZE}(poly) \implies \mathsf{UEXP}/a = \mathsf{MA}/a$.

Proof. *Proof of (1):* For $L \in \mathsf{UTIME}(t)$, 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 $\mathsf{UTIME}(t)$ algorithm for $L_{ewl(V)}$ from Theorem 3.1 puts it into $C(m^k)$ for input size m. The 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 $L \in \mathsf{ZNTIME}(t)$, construct the same verifier V' for the language $L' = \{0,1\}^*$ as in the proof of Theorem 3.1. As $L' \in \mathsf{UTIME}(t)$, V' will have witness in $C(n^k)$. Now a $\mathsf{DTIME}(2^{n^{k+1}})$ algorithm for L is: for n-length input x, go through all the circuits in $C(n^k \log n)$ 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 $x \notin L$.

Proofs of (2) & (3): They are analogous to the proof of (1), except that they use Corollary 3.2.

Proof of (4): Let $L \in \mathsf{UEXP}/a$, and V/a be an unambiguous (given the correct advice) verifier V for L that runs in time $O(2^{n^c})$ for some constant c. Since $\exists k\ L \in \mathsf{SIZE}(n^k)$, from the proof of part (3) we know that V/a has witnesses in $\mathsf{SIZE}(n^{ck})$ for some constant k.

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

Once we get $\mathsf{UEXP}/a = \mathsf{EXP}/a$, $\mathsf{EXP}/a \subseteq \mathsf{SIZE}(poly)$ gives $\mathsf{UEXP}/a = \mathsf{MA}/a$ [55]. □

3.3 EWL and KLT for ZUTIME

We extend the techniques from the previous section to give similar results for ZUTIME. The main difference in the proof of our search to decision reduction is that, we adapt our definition of $L_{ewl(V)}$ to capture seeds of zero-error non-deterministic verifiers. First let's define this adaptation. For zero-error non-deterministic verifier V for language L:

$$L_{ewl(V)} = \{(x, i) \mid \exists y\ [V(x, y) \in \{0, 1\} \wedge (i^{th}\text{-bit of } y \text{ is } 1) \wedge \forall (z <_{l.o.} y)\ V(x, z) = ?]\} \quad (3)$$

The difference is that $L_{ewl(V)}$ captures the lexicographically first certificate that gives the correct answer (doesn't matter whether the answer is 1 or 0). Once the search to decision reduction is established, the EWL and KLT follow from similar arguments as in the previous section. Here again we see why it might be difficult to extend these results to all ambiguous zero-error verifiers.

Theorem 3.4. *For time-constructible $t \in 2^{O(n)}$, and constants c and k:*

(1) *For $L \in \mathsf{ZUTIME}(t)$ and zero-error unambiguous verifier V for L, $L_{ewl(V)} \in \mathsf{ZUTIME}(t(n))$ (where n is the input size for L). Moreover if this statement is true for all $O(t)$-time zero-error non-deterministic verifiers (ambiguous and unambiguous) for every $\mathsf{ZUTIME}(t)$ language, then $\mathsf{ZNTIME}(t) = \mathsf{ZUTIME}(t)$.*
(2) *$\mathsf{ZUTIME}(t) \subset C(n^k) \implies \mathsf{ZUTIME}(t)$ has oblivious seeds in $C(n^k)$. Moreover if this statement is true for all $O(t)$-time zero-error non-deterministic verifiers (ambiguous and unambiguous) for every $\mathsf{ZUTIME}(t)$ language, even just for seeds (let alone oblivious-seeds), then $\mathsf{ZNTIME}(t) \subseteq \mathsf{DTIME}(2^{n^{k+1}})$.*
(3) *$\mathsf{ZUTIME}(2^{n^c}) \subset C(n^k) \implies \mathsf{ZUTIME}(2^{n^c})$ has oblivious seeds in $C(n^{ck})$.*
(4) *$\mathsf{ZUEXP} \subset \mathsf{SIZE}(poly) \implies \mathsf{ZUEXP} = \mathsf{MA}$.*

Proof. *Proof of (1): Algorithm for $L_{ewl(V)}$:* For input (x, i), guess a certificate y and simulate $V(x, y)$. Output '?' is V outputs '?'. Output 1 if V outputs in $\{0, 1\}$ and the i^{th}-bit of y is 1. Output 0 if V outputs in $\{0, 1\}$ and the i^{th}-bit of y is 0. This algorithm is correct and zero-error unambiguous as V is zero-error unambiguous. It runs in time $O(t(|x|))$.

The moreover part: For $L \in \mathsf{ZNTIME}(t)$, let V be its $\mathsf{ZNTIME}(t)$ verifier. Consider the $\mathsf{ZUTIME}(t)$ language $L' = \{0, 1\}^*$.

Using V we first construct a $\mathsf{ZNTIME}(t)$ verifier V' for L': V' ignores the first bit of the certificate and simulates V using the rest of the certificate. V' outputs '?' if V outputs '?', it outputs 1 if V outputs in $\{0, 1\}$ and its output matches the first bit of the certificate.

Now using a $\mathsf{ZUTIME}(t(n))$ algorithm $\mathcal{A}$ for $L'_{ewl(V')}$ we give a $\mathsf{ZUTIME}(t)$ algorithm for L: on input x, simulate $\mathcal{A}$ on $(x, 1)$. Output whatever $\mathcal{A}$ outputs.

Proofs of (2), (3) & (4): They are analogous to the proofs in Theorem 3.3. □

3.4 EWL and KLT for $\widetilde{\mathsf{FewTIME}}$

We use the following folklore result to translate our results for Boolean circuits to any typical circuit class.

Lemma 3.5. *If $\mathsf{P} \subset C$, then there exists a constant c such that: for large enough n, any s-size circuit has an equivalent s^c-size C circuit.*

Proof. $\mathtt{Ckt\text{-}Eval}$ 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 $\mathsf{P} \subset C$, then there is a constant c such that $\mathtt{Ckt\text{-}Eval}$ has $n^{c/2}$-size C circuits.

Let B be a $\mathsf{P/poly}$ circuit of size s. Let E be $(n + s \log s)^{c/2}$-size circuit corresponding to the $(n + s \log s)^{th}$-slice of $\mathtt{Ckt\text{-}Eval}$. Define $D(x) = E(B, x)$. It is easy to check that: (i) D is an s^c-size C circuit; and (ii) D is equivalent to B. □

Now we give the EWL and KLT for $\widetilde{\mathsf{FewE}}$.

Theorem 3.6. *For constant $k \geq 1$:*

(1) *$\widetilde{\mathsf{FewE}}/(a + n) \subset C(n^k) \implies \exists k'\ \widetilde{\mathsf{FewE}}/a$ has witnesses in $C(n^{k'})$*
(2) *$\widetilde{\mathsf{FewE}}/(a + n) \subset \mathsf{SIZE}(n^k) \implies \widetilde{\mathsf{FewE}}/a \subseteq \mathsf{MA}/a$*

Proof. *Proof of (1):* We prove the result for unrestricted Boolean circuits. The result for the circuit class C follows from the Lemma 3.5. The assumption implies $\mathsf{P} \subset C$, thus any $\mathsf{SIZE}(n^{k'})$ circuit has an equivalent $C(n^{ck'})$ circuit, for some constant c.

Contradiction: $\widetilde{\mathsf{FewE}}/(a + n) \subset \mathsf{SIZE}(n^k)$ implies $\mathsf{EXP} \subset \mathsf{SIZE}(poly)$ and thus $\mathsf{EXP} = \mathsf{MA}$ [9]. Now we show that, if $\forall k' \geq 1\ \widetilde{\mathsf{FewE}}/a$ doesn't have witnesses in $\mathsf{SIZE}(n^{k'})$, then $\mathsf{MA} \subset \text{io-}\widetilde{\mathsf{FewE}}/(a + n)$.

Combined with the above statement it leads to the contradiction $\mathsf{EXP} \subset$ io-$\mathsf{SIZE}(n^k)$ (since we can diagonalize against fixed-polynomial size circuits in EXP).

Hardness tester: $\forall k \geq 1$ $\widetilde{\mathsf{FewE}}/a$ doesn't have witnesses in $\mathsf{SIZE}(n^k)$ implies that for every $k \geq 1$, there is a $2^{\sqrt{n}}$-time $\widetilde{\mathsf{Few}}/a$ verifier V_k/a that has infinite set of inputs S_k that it accepts, and for $x \in S_k$ and $2^{\sqrt{|x|}}$-length certificate y such that $V_k(x, y)/a = 1$ (using the correct advice), the constraint $ckt(y) \geq n^k$ is true (where y is truth-table of a $\sqrt{n}$-input circuit). If V_k with its original a amount of advice, is also given elements of S_k as advice (one element per input length, and all 0s string for the input lengths for which S_k contains no element), V_k becomes a $\widetilde{\mathsf{FewTIME}}(2^{\sqrt{n}})/(a+n)$ hardness tester.

Derandomization: For $L \in \mathsf{MA}$, we derandomize the MA protocol for L using the above described $\widetilde{\mathsf{Few}}$ verifiers. After including the non-determinism of Merlin into Arthur's input, let the size of the circuit C that captures the BP computation of Arthur for L be bounded by n^l (for some constant l). We use the verifier V_k for $k = lg$, where g is the constant from Theorem 2.3. Our algorithm guesses a 2^n-bit string Y and simulates V_k on Y, using the $a + n$ amount of advice as described above. It rejects if V_k rejects, else it uses the certificate that V_k accepted. Note that, infinitely often the accepted certificates Y will satisfy $ckt(Y) \geq n^{lg}$. We use the certificates to construct a PRG $G : n \to n^l$ using the Theorem 2.3, that fools n^l-size circuits. We brute-force through the seeds of G to compute the acceptance probability of the circuit C in $2^{O(n)}$-time (within $\pm 1/n^l$ error). If the acceptance probability is greater than $1/2$, our algorithm accepts, else it rejects. The running time of our non-deterministic algorithm is bounded by 2^n, and the number of accepting branches is bounded by $2^{2^{(\log n)^2/4}} \times 2^{n^l}$, which is less that $2^{2^{(\log n)^2}}$ for large enough n.

Proof of (2): $\widetilde{\mathsf{FewE}}/(a+n) \subset \mathsf{SIZE}(n^k)$ combined with (1) gives $\widetilde{\mathsf{FewE}}/a$ has witnesses in $\mathsf{SIZE}(n^{k'})$ for some constant k'. This gives $\widetilde{\mathsf{FewE}}/a \subset \mathsf{EXP}/a$: by brute-forcing through the truth-tables of all $\mathsf{SIZE}(n^{k'})$ circuits to find accepting certificates (if there are any). Finally we get $\widetilde{\mathsf{FewE}}/a \subseteq \mathsf{MA}/a$ since $\mathsf{EXP}/a = \mathsf{MA}/a$ by [55]. □

4 UEXP LOWER BOUNDS FROM FAST UNAMBIGUOUS ALGORITHMS

First we show how to get UEXP lower bounds from fast unambiguous algorithms for canonization and tautology (Section 4.1). Then we show how to replace canonization and tautology by $\Pi_2\mathsf{SAT}$ and get more fine-grained results (Section 4.2). Then we show how to completely get rid of canonization for the case $\widetilde{\mathsf{FewE}}$ lower bounds (Section 4.3). Then we show how to simulate canonization using proper learning (Section 4.4). Finally, we show how to generalize certain lower bound frameworks for unrestricted Boolean circuits, to typical circuit classes, even when the algorithms are very slow (Section 4.5). We use the following hierarchy for semantic classes in our proofs.

Theorem 4.1 (Heirarchy for Semantic Classes [33]). *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 $\mathsf{UTIME}(t)/a \not\subset \mathsf{UTIME}(t^\epsilon)/(a+1)$ (resp. $\widetilde{\mathsf{FewTIME}}(t)/a \not\subset \widetilde{\mathsf{FewTIME}}(t^\epsilon)/(a+1)$).*

4.1 Lower bounds from unambiguous tautology and canonization algorithms

We use the following 'tight reductions to 3-USAT' in this section.

Theorem 4.2 (Efficient local reductions [31, 50, 89]). *Every language $L \in \mathsf{UTIME}(2^n)$ 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 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.*

We first formally define canonization and related notations.

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

$\mathsf{TAUT}_{(s,C)}$ (*resp.* $\mathsf{SAT}_{(s,C)}$) denotes the TAUT (*resp.* SAT) for s-size C circuits.

In these definitions we omit, the parameter s when it is $poly(n)$, and the circuit class when $C =$ Boolean.

The main idea is to guess the witness circuit unambiguoulsy using the canonization algorithm, and then combine the witness circuit with the reduction circuit in the same manner that Williams did [98]. The existence of the witness circuit follows from the UTIME EWL.

Theorem 4.3. *For $\delta \leq 1$, let a, c and ϵ be the parameters of Theorems 4.1 and 4.2 for the time bound $t = 2^{\delta n}$. Then for constant k and function $p(n) \geq n$, $\mathsf{UTIME}(2^{\delta n})/a \not\subset C(n^k)$ if:*

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

Proof. Using the assumptions (1 or 2), we will contradict the UTIME hierarchy (Theorem 4.1) by designing a $\mathsf{UTIME}(2^{\epsilon n})/(a+1)$ algorithm for arbitrary $L \in \mathsf{UTIME}(2^{\delta n})/a$.

Reduction circuit: For $L \in \mathsf{UTIME}(2^{\delta n})/a$ and input x, let F_x be the $2^n n^c$-size 3-USAT formula we get by reducing from x (Theorem 4.2). There is an n^c-size (P-uniform) C circuit D with $n + c\log n$ input wires, that outputs the i^{th}-clause of F when given the input $i \in [1, 2^n n^c]$.

Special verifier: Let V be the verifier for L that first reduces input x to the 3-USAT formula F_x, and then non-deterministically guesses a satisfying assignment for F_x.

Easy-witness circuit: From UTIME EWL (Theorem 3.3) and the assumption $\mathsf{UTIME}(2^{\delta n})/a \subset C(n^k)$ we know that V has witness circuits in $C(n^k)$. Let E be a witness circuit of this verifier for the input length $|x| = n$.

Final circuit C: Combining D and E we construct a circuit C that satisfies: "C is a tautology $\iff x \in L$". On input i, the output of D is $3n + 3c\log n + 3$ bits long. 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 C circuit E. Simulate the $\mathsf{CAN}_{(n^{k+1},C,p)}$ algorithm on E. This requires $\mathsf{UTIME}(2^{\epsilon n})/1$. Reject if the answer is negative. Continue if it's 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 $\mathsf{TAUT}_{(p(n^{k+1})n+n^c,C)}$ algorithm on C. This requires $\mathsf{UTIME}(2^{\epsilon n})$. 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 $\mathsf{UTIME}(2^{\delta n})/a$ algorithm for L. So we get a $\mathsf{UTIME}(2^{\epsilon n})/(a+1)$ algorithm.

Contradicting the second assumption: The algorithm is exactly the same, expect that the extra 1-bit of advice is used by the tautology algorithm, and not by the canonization algorithm. □

We get the following corollary that is cleaner in presentation.

Corollary 4.4. $\mathsf{UE}/O(n\log n) \not\subset C$*, if* $\mathsf{TAUT}_C \in \mathsf{USUBE}$ *and* $\mathsf{CAN}_{(C,p)} \in \mathsf{USUBE}$*, for* $p(n) \in poly(n)$.

4.2 Lower bounds from unambiguous $\Pi_2\mathsf{SAT}$ algorithms

Here the idea is to simulate canonization using a $\Pi_2\mathsf{SAT}$ algorithm.

Theorem 4.5. *For every constant k_1 there is a constant k_2, such that if $\Pi_2\mathsf{SAT}$ on n variables and n clauses can be solved in $\mathsf{UTIME}(2^{n/(\log n)^{k_2}})$, then $\mathsf{UE}/n \not\subset \mathsf{SIZE}(n(\log n)^{k_1})$.*

Proof. From the Theorem 4.1 we know that there is an $a \leq n$ such that $\mathtt{UTIME}(2^{n/(\log n)^2})/a \not\subset \mathtt{UTIME}(2^{n/(\log n)^3})/(a+1)$. So if $\mathtt{UE}/n \subset \mathtt{SIZE}(n(\log n)^{k_1})$, then $\mathtt{UTIME}(2^{n/(\log n)^2})/a \subset \mathtt{SIZE}(n(\log n)^{k_1})$. From the UTIME EWL we get $\mathtt{UTIME}(2^{n/(\log n)^2})/a \subset_w \mathtt{SIZE}(n(\log n)^{k_1})$.

Now the proof is similar to the proof of Theorem 4.3, except few changes. We prove that any $L \in \mathtt{UTIME}(2^{n/(\log n)^2})/a$ has an $\mathtt{UTIME}(2^{n/(\log n)^3})/a$ algorithm. After guessing a $\mathtt{SIZE}(n(\log n)^{k_1})$ witness circuit E, we use a fast $\Pi_2\mathsf{SAT}$ algorithm on it (from our assumption) to make sure that we move forward unambiguously. We check that, for all lexicographically small (in some fixed encoding scheme) circuits D, there is at least one input z, such that $E(z) \neq D(z)$. For the final circuit C, we use three copies of E and a reduction circuit that is (P-uniform) linear-size (the n^c-size circuit in the Theorem 6.9 can be made linear [50]). Finally we run a fast tautology algorithm (from our assumption) on the circuit C. □

4.3 Lower bounds from $\widetilde{\mathtt{Few}}$ tautology algorithms

The idea is that there can only be exponential many possibilities for the witness circuit, and thus the number of positive non-deterministic branches of the final algorithm remain within the limits of a $\widetilde{\mathtt{Few}}$ verifier.

Theorem 4.6. $\mathtt{TAUT}_C \in \widetilde{\mathtt{Few}}\mathtt{SUBE} \implies \forall k\ \widetilde{\mathtt{Few}}\mathtt{E}/O(n \log n) \not\subset \mathcal{C}(n^k)$.

Proof. For the sake of contradiction, assume that $\exists k\ \widetilde{\mathtt{Few}}\mathtt{E}/O(n\log n) \subset \mathcal{C}(n^k)$. From the $\widetilde{\mathtt{Few}}\mathtt{TIME}$ EWL (Theorem 3.6) we get that $\exists k'\ \widetilde{\mathtt{Few}}\mathtt{E}/a \subset_w \mathcal{C}(k')$, where a is the advice parameter of the Theorem 4.1 for time bound $t = 2^n$. Using the fast $\widetilde{\mathtt{Few}}$ algorithm from our assumption we contradict the Theorem 4.1 by showing that any arbitrary $L \in \mathtt{FewE}/a$ has an $\widetilde{\mathtt{Few}}\mathtt{SUBE}/(a+1)$ algorithm.

Now again our proof follows the structure of Theorem 4.3 with some modifications. We construct the final circuit C without using any canonization. There can only be $2^{O(n^{k'})}$ many witness circuits E. For designing a $\widetilde{\mathtt{Few}}\mathtt{TIME}(2^{n^\epsilon})/a$ algorithm for any $0 < \epsilon < 1$, we use a $\widetilde{\mathtt{Few}}\mathtt{TIME}(2^{n^\delta})$ tautology algorithm for some $\delta \leq \epsilon$. This makes the number of accepting paths for the final algorithm at most $2^{O(n^{k'})} \times 2^{(\delta n)^{\log(\delta n)}} \leq 2^{(\epsilon n)^{\log(\epsilon n)}}$. □

4.4 Lower bounds from unambiguous learning and tautology algorithms

We first show how exact (proper) learning along with tautology algorithm imply canonization. Then we plug this connection in the Theorem 4.3 to get lower bounds from learning and tautology algorithms. Before we give our result, we first formally define the UTIME exact learning algorithm that we use in our results.

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 $\mathsf{LRN}_{(s,\mathcal{C},p)}$, if for any s-size $\mathcal{C}$ circuit C, it outputs a circuit C' of size at most $p(s)$ in time at most $t(n)$ (where n is the number of input wires) 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 of the type: “What is the value of $C(x)$?”. An equivalence query is of the type: “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 polynomial-size $\mathcal{C}$ circuits, the algorithm is called $\mathsf{P}\text{-}\mathsf{LRN}_{(s,\mathcal{C},p)}$ (proper learning).

Here again, we omit the size parameter when $s(n) = poly(n)$, and the circuit class when $\mathcal{C} =$ Boolean. Here we omit $p(n)$ too, if it is $poly(n)$. Unlike in $\mathsf{CAN}_{(\mathcal{C},p)}$, in $\mathsf{LRN}_{(\mathcal{C},p)}$ p decides the size of the output (and not the input).

Theorem 4.7. *For any polynomial $p(n)$:*

(1) $\mathsf{P\text{-}LRN}_{(C,p)} \in \mathsf{UTIME}(t) \ \wedge\ \mathsf{TAUT}_C \in \mathsf{UTIME}(t') \implies \mathsf{CAN}_{(C,p)} \in \mathsf{UTIME}(t(t' + poly(n)))$

(2) $\mathsf{P\text{-}LRN}_{(C,p)} \in \mathsf{USUBE} \wedge \mathsf{TAUT}_C \in \mathsf{USUBE} \implies \mathsf{UE}/O(n \log n) \not\subset C$

Proof. *Proof of (1):* In an exact proper learning algorithm, if we have access to the circuit C that we are learning, then we can get a canonization algorithm for 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). As the final hypothesis will be of size at most $p(s)$ for s-size C circuits, we get a UTIME algorithm for $\mathsf{CAN}_{(C,p)}$. The membership queries can be handled directly since we have the circuit with us. For the equivalence queries, we non-deterministically guess the faith for the hypothesis H.

If we guess $H \equiv C$, we use the tautology algorithm to verify it. If our guess is wrong, we reject. If our guess is right, we accept if only if, H's description is same as C.

If we guess $H \not\equiv C$, we have to output an input z, such that $H(z) \neq C(z)$. We try to guess the lexicographically smallest such z to keep the whole process unambiguous. After guessing z, we check that $H(z) \neq C(z)$ and use our tautology algorithm to check that $\forall(z' <_{l.o.} z)\ H(z) = C(z)$, where l.o. stands for lexicographical ordering (the test $z' <_{l.o.} z$ can be encoded by any typical circuit of linear size). We return z if both the checks pass, else we reject.

Proof of (2): We get this directly from (1) and the Corollary 4.4. □

4.5 Generalization of lower bound frameworks

In the above sections we saw that fast UTIME algorithms for certain circuit analysis algorithms for C circuits were fed to certain frameworks to yield lower bounds for UTIME against C. Consider the scenario where: a framework is altogether different, or is a fine-grained version of one of the current ones (in terms of size and depth of the circuits and running time of the algorithms), and works for Boolean circuits, but not for some restriction C. Also consider that, the assumptions of these frameworks are satisfied for that 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 win-win type arguments analogous to the ones used in [72] for fast NTIME algorithms. We show that, either $\mathsf{P} \not\subset C$ (i.e., a stronger lower bound exists against C), or fast unambiguous algorithms for C circuits imply fast unambiguous algorithms for Boolean circuits (i.e., frameworks that only work for Boolean circuits can now be used). To prove our results, we use the Lemma 3.5.

Theorem 4.8. *Either $\mathsf{P} \not\subset C$, or $\exists c$, for $p(m) = m^k$ for any $k \geq 1$, and $t, t', t'' \leq 2^n$:*

(1) $\mathsf{CAN}_{(C,p)} \in \mathsf{UTIME}(t) \implies \mathsf{CAN}_{p^c} \in \mathsf{UTIME}(t)$

(2) $\mathsf{CAN}_{(C,p)} \in \mathsf{UTIME}(t) \ \wedge\ \mathsf{TAUT}_C \in \mathsf{UTIME}(t') \implies \mathsf{TAUT} \in \mathsf{UTIME}((t + t')poly(n))$

(3) $\mathsf{CAN}_{(C,p)} \in \mathsf{UTIME}(t) \ \wedge\ \mathsf{TAUT}_C \in \mathsf{UTIME}(t') \ \wedge\ \mathsf{SAT}_C \in \mathsf{UTIME}(t'')$
$\implies \mathsf{SAT} \in \mathsf{UTIME}((t + t')poly(n) + t'')$

Proof. If $\mathsf{P} \subset C$, from the Lemma 3.5 we know there exists a constant c such that: for each s-size Boolean circuit B, there is an equivalent s^c-size C circuit C (for large enough n).

Proof of (1): By a simple modification of an algorithm $\mathcal{A}$ for $\mathsf{CAN}_{(C,p)}$, we obtain an algorithm $\mathcal{A}'$ for CAN_{p^c}. On input B, the algorithm $\mathcal{A}'$ first checks whether B belongs to 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 $\mathsf{UTIME}(t)$ algorithm for $\mathsf{CAN}_{(C,p)}$, $\mathcal{A}'$ be a $\mathsf{UTIME}(t')$ algorithm for TAUT_C. Using $\mathcal{A}$ and $\mathcal{A}'$, we construct a UTIME algorithm $\mathcal{A}''$ for 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 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 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 equivalent C circuit C. Then, on this C we simulate a $\mathsf{UTIME}(t'')$ algorithm for SAT_C. □

For the case of $\widetilde{\mathsf{Few}}$ algorithms, we get the following theorem where we don't need canonization. The proof is same as of the above theorem, except that now we can skip all the canonization steps. This change will not increase the number of positive non-deterministic branches of the final algorithm by much, and thus the constraints of a $\widetilde{\mathsf{Few}}$ verifier are not violated.

Theorem 4.9. *Either* $\mathsf{P} \not\subset C$, *or:*

(1) $\mathsf{TAUT}_C \in \widetilde{\mathsf{FewTIME}}(2^n/n^{\omega(1)}) \implies \mathsf{TAUT} \in \widetilde{\mathsf{FewTIME}}(2^n/n^{\omega(1)})$

(2) $\mathsf{TAUT}_C \in \widetilde{\mathsf{FewTIME}}(2^n/n^{\omega(1)}) \wedge \mathsf{SAT}_C \in \widetilde{\mathsf{FewTIME}}(2^n/n^{\omega(1)}) \implies \mathsf{SAT} \in \widetilde{\mathsf{FewTIME}}(2^n/n^{\omega(1)})$

5 UNIQUE PROPERTIES VS. LOWER BOUNDS

In this section we establish relationships between unique properties and ZUTIME (Section 5.2) and UTIME (Section 5.3) lower bounds. In both these connections we use the equivalence between UP-U and P-U properties (Section 5.1). Then we use these connections to derive zero-error unambiguous derandomization under UEXP lower bounds (Section 5.4).

5.1 UP-U properties vs. P-U properties

The proof of this connection is along the same lines as the original connection [4, 72, 100]: an useful NP (*resp.* RP-natural) property yields an useful P (*resp.* P-natural) property.

Theorem 5.1. 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) $\mathcal{U}$ *is useful against* C $\implies$ $\mathcal{P}$ *is useful against* 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): 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/a property. 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 (2): If $\mathcal{U}$ is useful against C, then for each k there exists an infinite subset S_k such that for each $n \in S_k$, $\mathcal{U}(x) = 1 \implies ckt_C(x) > n^k$. For any x, let y be the unique certificate such that $V(x, y) = 1$. Since $ckt_C(x) > n^k \implies ckt_C(xy) > n^k \geq (cn)^{k-1}$ for each k, $\mathcal{P}$ is useful against n^{k-1}-size C circuits for each k, and hence is useful against C. □

5.2 ZUE lower bounds vs P-U properties

We extend arguments from [72] to give the following result. Unlike the $\mathsf{NE} \cap \mathsf{Co\text{-}NE}$ case, we get the equivalence too exploiting the fact that any unique property has a fix size.

Theorem 5.2. *The following statements are equivalent:*

(1) ZUE *doesn't have* C *circuits*
(2) ZUE *doesn't have oblivious seeds in* C
(3) ZUE *doesn't have hitting-sets for seeds in* C
(4) ZUE *doesn't have seeds in* C
(5) *There exists a* P *computable unique property against* C

Proof. (1) $\implies$ (5) Let $L \in \mathsf{ZUE} \setminus C$, and let V be $2^{O(n)}$-time zero-error 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 UP-U property $\mathcal{U}$ that is useful against C. Then the result follows from the Theorem 5.1.

For any input y of length 2^n, $\mathcal{U}$ simulates V on all the n-length strings, one by one. For each i, it matches the i^{th} bit of y, and the output of V on the i^{th} n-length string. $\mathcal{U}$ accepts if and only if it succeeds in all 2^n verifications.

Constructivity & uniqueness: For $n \in \mathbb{N}$, $\mathcal{U}$ unambiguously accepts the truth table of 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 (as V is ZUE).

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

(5) $\implies$ (4) Let $\mathcal{P}$ be a P-unique property useful against C. Using $\mathcal{P}$ we construct a zero-error unambiguous verifier V for the ZUE language $\{0, 1\}^*$ such that V doesn't have seeds in 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 C, the unique accepting witnesses of V are not in C.

(4) $\implies$ (3) This follows from the definitions.

(3) $\implies$ (2) This follows from the definitions.

(2) $\implies$ (1) The contrapositive follows from the ZUTIME EWL (Theorem 3.4). □

5.3 UE/n lower bounds vs P/log n-U properties

We extend arguments from [100] to give the following result. Here unlike the NE case, we need advice to establish relationship between lower bounds for UE witnesses and oblivious witnesses.

Theorem 5.3. *The following statements are equivalent for any constant* $k \geq 1$*:*

(1) UE/n^k *doesn't have* C *circuits*
(2) UE/n^k *doesn't have oblivious witnesses in* C
(3) UE/n^k *doesn't have hitting-sets for witnesses in* C
(4) UE/n^k *doesn't have witnesses in* C
(5) *There exists a* P/$(\log n)^k$ *computable unique property against* C

Proof. (1) $\implies$ (5) Let $L \in \mathsf{UE}/n^k \setminus C$, 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 UP/$\log^k m$-U property $\mathcal{U}$ that is useful against C. Then the result follows from the Theorem 5.1.

For odd m, among all the inputs of length 2^m, $\mathcal{U}$ only accepts the all 0s string. For $m = 2n$ for some n, for any 2^m-length input yz with $|y| = 2^n$ and $|z| = 2^{2n} - 2^n$, $\mathcal{U}$ 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 in the lexicographical order (to verify its inclusion in L). The first n^k bits of advice is the advice required for the simulation of V. The rest of the $(\log 2^{2n})^k - n^k = (2n)^k - n^k \geq n$ bits of advise encodes the number of n-length inputs that V accepts. $\mathcal{U}$ accepts if and only if: (i) it succeeds in all 2^n verifications; (ii) the hamming weight of y is equal to the number encoded by the last $(2n)^k - n^k$ bits of advise; and (iii) z is an all 0s string.

Constructivity & uniqueness: For $n \in \mathbb{N}$, $\mathcal{U}$ unambiguously accepts the truth table of function f_n (followed by an all 0s string of length $2^{2n} - 2^n$), and rejects all the other strings. As it runs for $2^{O(n)}$-time for 2^n-length inputs with n^k-size advice, it is $\mathsf{UP}/(\log n)^k$-U (as V is UE).

Usefulness: As $L \notin C$, for each l, there are infinitely many input lengths n, such that f_n doesn't have n^{l+1}-size C circuits. Corresponding to each such n, for the inputs of length 2^{2n}, $\mathcal{U}$ accepts strings y that doesn't have $(2n)^l$-size C circuits because: any $(2n)^l$-size circuit C with $tt(C) = y$, decides L_n after we fix the first half of its input wires to 1s, and $(2n)^l \leq n^{l+1}$.

(5) $\implies$ (4) Let $\mathcal{P}$ be a $\mathsf{P}/(\log n)^k$-U property useful against C. We construct an unambiguous verifier V for the UE/n^k language $\{0,1\}^*$, that doesn't have witnesses in 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 C, V doesn't have witnesses in C. As $\mathcal{P}$ is unique, V is UE/n^k.

(4) $\implies$ (3) This follows from the definitions.

(3) $\implies$ (2) This follows from the definitions.

(2) $\implies$ (1) The contrapositive follows from the UTIME EWL (Theorem 3.3). □

5.4 Derandomization using unique properties

Here we extend the lower-bounds to derandomization connection to UEXP and ZUEXP lower bounds. We use the two connections from the above two sections. The idea is to obtain unique properties from UEXP and ZUEXP lower bounds, and then use these properties to unambiguously obtain hard functions, which then yield the desired derandomization.

Theorem 5.4. *[Unambiguous derandomization from* UEXP *and* ZUEXP *lower bounds]*

(1) $\mathsf{ZUEXP} \neq \mathsf{EXP} \implies \mathsf{BPP} \subset \cap_{\epsilon>0}\ io\text{-}\mathsf{ZUTIME}(2^{n^\epsilon})$
(2) $\mathsf{ZUEXP} \not\subset \mathsf{SIZE}(poly) \implies \mathsf{BPP} \subset \cap_{\epsilon>0}\ io\text{-}\mathsf{ZUTIME}(2^{n^\epsilon})$
(3) $\mathsf{ZUEXP} \neq \mathsf{BPP} \implies \mathsf{BPP} \subset \cap_{\epsilon>0}\ io\text{-}Heur\text{-}\mathsf{ZUTIME}(2^{n^\epsilon})$
(4) $\mathsf{UEXP} \neq \mathsf{EXP} \implies \mathsf{BPP} \cap_{\epsilon>0} \subset io\text{-}\mathsf{ZUTIME}(2^{n^\epsilon})/n^\epsilon$
(5) $\mathsf{UEXP} \not\subset \mathsf{SIZE}(poly) \implies \mathsf{BPP} \subset \cap_{\epsilon>0}\ io\text{-}\mathsf{ZUTIME}(2^{n^\epsilon})/n^\epsilon$
(6) $\mathsf{UEXP} \neq \mathsf{BPP} \implies \mathsf{BPP} \subset \cap_{\epsilon>0}\ io\text{-}Heur\text{-}\mathsf{ZUTIME}(2^{n^\epsilon})/n^\epsilon$

Proof. *Proof of (1):* Let's assume that $\mathsf{ZUEXP} \neq \mathsf{EXP}$. Then ZUE can't have seeds in $\mathsf{SIZE}(poly)$, because brute-forcing through the seeds will prove $\mathsf{ZUEXP} = \mathsf{EXP}$. Thus, there exists a P-U property $\mathcal{P}$ useful against $\mathsf{SIZE}(poly)$ (from the Theorem 5.2). For each c, let S_c be the infinite set of input lengths where $\mathcal{P}$ only accept strings str satisfying $ckt(str) \geq n^c$. These strings are truth-tables of hard functions, and can be computed in UE using the constructivity of $\mathcal{P}$.

For $k, \epsilon > 0, \epsilon > \epsilon'/2$ and $L \in \mathsf{BPTIME}(n^{k/2})$, set $c = gk/\epsilon'$ (where g is the constant from Theorem 2.3). We give a $\mathsf{ZCTIME}(2^{n^\epsilon})$ 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 $\mathsf{SIZE}(n^k)$ circuit capturing the BP computation of L.

Non-deterministically guess a string Y of length $m = 2^{n^{\epsilon'}}$. Output '?' if $\mathcal{P}(Y) = 0$. Once we have access to Y with $\mathcal{P}(Y) = 1$ (or $ckt(Y) \geq n^k$), we can construct PRG $G : n^\epsilon \to n^k$ from Y (using the Theorem 2.3) that is computable in E. We brute-force through all the $n^{\epsilon'}$-length seeds, and on each of the output strings of length n^k, compute the circuit C_x to calculate its acceptance probability in time 2^{n^ϵ} (within $\pm 1/n^k$ error). Output 1 if this value is 1/2 or more, else output 0. Since $\mathcal{P}(Y) = 1$ holds for unique Y, the whole process is unambiguous.

Proofs of (2): We prove the contrapositive. Assume that $\exists\epsilon > 0$ such that $\mathsf{BPP} \not\subset$ io-$\mathsf{ZUTIME}(2^{n^\epsilon})$. This gives us $\mathsf{EXP} \subset \mathsf{SIZE}(poly)$ [9, 70, 71], and $\mathsf{ZUEXP} = \mathsf{EXP}$ from (1). Thus, $\mathsf{ZUEXP} \subset \mathsf{SIZE}(poly)$.

Proof of (3): Either $\mathsf{ZUEXP} \neq \mathsf{EXP}$ or $\mathsf{EXP} \neq \mathsf{BPP}$. The former gives better derandomization from (1), and the latter gives the desired derandomization from [48].

Proof of (4): It's analogous to the proof of (1), except that the property we get is $\mathsf{P}/\log n$ and not P constructive (from the Theorem 5.3). The $\log n$-bit advice for this property is precisely the n^{ϵ}-bit advice for the $\mathsf{ZUTIME}(2^{n^{\epsilon}})$ algorithm we get.

Proofs of (5) & (6): They are analogous to the proofs of (2) and (3). The advice from the proof of (4) travels to them as well. □

6 LOWER BOUNDS AGAINST prSV NON-DETERMINISTIC CIRCUITS

Here we discuss all the results regarding lower bounds against prSV non-deterministic circuits. First we derive the EWL for the case of NEXP and $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$ (Section 6.1) and use it to derive the connection between non-trivial $\mathsf{GAP\text{-}SAT}$ algorithm and the lower bound $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$ (Section 6.2). Then we derive new gap theorems for MA and CAPP (Section 6.3). Then we derive connections between fast algorithms and NE and E^{NP} lower bounds against circuits that use limited amount of prSV non-determinism (Section 6.4) and use that connection to derive unconditional lower bounds (Section 6.5). Finally we show unconditional lower bounds against fixed-polynomial prSV non-deterministic circuits (Section 6.6).

6.1 NEXP vs $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$

Here we give NEXP EWL and KLT for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, and the converses. We also extend the results to $\mathsf{E}^{\mathsf{NP}}_{||}$. These results work even if replace $\mathsf{NP} \cap \mathsf{Co\text{-}NP}$ with P (in circuit classes and as oracles).

Theorem 6.1. *The following statements are equivalent:*

(1) $\mathsf{NE} \not\subset \mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$
(2) $\mathsf{E}^{\mathsf{NP}}_{||} \not\subset \mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$
(3) $\mathsf{E}^{\mathsf{NP}}_{||} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$
(4) $\mathsf{NE} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$
(5) NE *doesn't have witnesses in* $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$
(6) NE *doesn't have hitting-sets for witnesses in* $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$
(7) NE *doesn't have oblivious witnesses in* $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$
(8) $\mathsf{prMA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}} \subset \cap_{\epsilon>0}$ *io*-$\mathsf{NTIME}(2^{n^{\epsilon}})/n^{\epsilon}$

Proof. (1) $\implies$ (2) This is trivial.

(2) $\implies$ (3) For the sake of contradiction, assume that $\mathsf{E}^{\mathsf{NP}}_{||} \not\subset \mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$ and $\mathsf{E}^{\mathsf{NP}}_{||} \subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. The latter implies $\mathsf{EXP} = \mathsf{AM} = \mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$. Thus the former implies $\mathsf{E}^{\mathsf{NP}}_{||} \not\subset \mathsf{EXP}$ and $\exists k$ $\mathsf{NE}/O(n) \subset \mathsf{NSIZE}(n^k)$ (using a linear time NE-complete language). $\mathsf{AM} \subset$ io-$\mathsf{NE}/O(n)$, and these implications, gives us the contradiction $\exists k$ $\mathsf{EXP} \subset$ io-$\mathsf{NSIZE}(n^k)$.

We show $\mathsf{AM} \subset$ io-$\mathsf{NE}/O(n)$ by using a language $L \in \mathsf{E}^{\mathsf{NP}}_{||}$, such that $L \notin \mathsf{EXP}$ (which again follows from the assumptions). For any $\mathsf{E}^{\mathsf{NP}}_{||}$ algorithm $\mathcal{A}$ deciding L, for any constant k, for infinitely many n, there can't be any NP-oracle n^k-size circuits encoding the witnesses for all the positive oracle queries that $\mathcal{A}$ makes on all n-length inputs. This is because, brute-forcing through such circuits will give an EXP algorithm for L. Now using $\mathcal{A}$ we get an $\mathsf{NE}/O(n)$ algorithm $\mathcal{B}$, that for each k, for infinitely many n, produces $2^{O(n)}$-length strings tt with $ckt^{\mathsf{NP}}(tt) > n^k$. The advice encodes the number of positive oracle queries that $\mathcal{A}$ makes on that input length. For any n, $\mathcal{B}$ simulates $\mathcal{A}$ on all n-length inputs and using advice guesses that many queries to be positive. It verifiers its guesses by non-deterministically guessing certificates for the positive oracle queries. After all the verification steps, it outputs the concatenation of all its non-deterministic certificates.

This concatenated string can't have NP-oracle n^k-size circuits, for any n where its sub-strings that represent the positive oracle queries of $\mathcal{A}$ doesn't have NP-oracle n^{k+1}-size circuits (because a circuit for the whole string, can be projected down to get a circuit for any sub-string). Now $\mathcal{B}$ gives us $\mathsf{AM} \subset$ io-$\mathsf{NE}/O(n)$, using the hardness to derandomization connection from [58].

(3) $\implies$ (4) This follows from the result $\mathsf{E}^{\mathsf{NP}}_{||} \subset \mathsf{NE}/O(n)$, where the advice gives the count of the number of positive oracle queries on all n-length inputs.

(4) $\implies$ (5) Let $L \in \mathsf{NE} \setminus (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. We construct a NE verifier V for the language $\{0,1\}^*$ that doesn't have witnesses in $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. V accepts any n-length string only on the 2^n-length witness that represents the characteristic function of L_n. Since $L \notin (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, witness of V are also not in $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$.

(5) $\implies$ (6) This follows from the definitions.

(6) $\implies$ (7) This follows from the definitions.

(7) $\implies$ (8) The NE verifier V that doesn't have oblivious-witnesses in $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, yields a function sequence computable in $\mathsf{NE}/O(n)$ that, for any constant k, for any $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$-oracle A, infinitely often, doesn't have A-oracle circuits of size n^k. The advice is used to encode the number of inputs that the NE verifier accepts, and output sequence is just the oblivious-witnesses of V. Moreover, by a simple padding argument, for any $\epsilon > 0$, the function sequence can be computed in $\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$.

Any language $L \in \mathsf{prMA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$, has MA protocols where Arthur uses some $A \in \mathsf{NP} \cap \mathsf{Co\text{-}NP}$ as oracle. After including the non-determinism of Merlin into the input, Arthur's computation can be converted into an A-oracle n^d-size circuit C for some constant d. This conversion only takes NP. Now for any $\epsilon > 0$, we derandomize these circuits for infinitely many input lengths n, in $\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$. This will establish $\mathsf{prMA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}} \subset \cap_{\epsilon>0}$ io-$\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$.

For any input length n, we first compute the function from the function sequence that doesn't have $n^{gd/\epsilon}$-size A-oracle circuits, and then use that function and the Theorem 2.3, to construct a PRG $G : n^\epsilon \to n^d$. This PRG fools n^d-size A-oracle circuits, and thus brute-forcing through its inputs, we can estimate the acceptance probability of C, and output accordingly.

(8) $\implies$ (1) If $\mathsf{NE} \subset \mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$ and $\mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}} \subset \cap_{\epsilon>0}$ io-$\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$, then we get $\mathsf{EXP} = \mathsf{NEXP} \subset \cap_{\epsilon>0}$ io-$\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$. This gives us $\mathsf{EXP} \subset \cap_{\epsilon>0}$ io-$\mathsf{TIME}(2^{n^c})/n$ for some constant c. This is false due to the diagonalization result given in [41]. □

6.2 Improving exhaustive search implies $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$

In this section we show that super-polynomial savings in non-deterministic algorithms for GAP-SAT for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$-oracle circuits, imply $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. We first state the following PCP verifier for NEXP, and hierarchy theorem for NTIME, that we will need in our result.

Theorem 6.2 (see [11, 98]). *For any $L \in \mathsf{NTIME}(2^n)$, there exists a PCP verifier $V(x,y,r)$ with soundness $1/2$, perfect completeness, randomness complexity $n + c\log n$, query complexity n^c, and verification time n^c, for some constant c. That means:*

- *V has random access to x and y, uses at most $|r| = n + c\log n$ random bits in any execution, makes n^c queries to the candidate proof y, and runs in at most n^c steps.*
- *if $x \in L$, $\exists y : |y| = n^c$ $Pr_r[V(x,y,r) = 1] = 1$.*
- *if $x \notin L$, $\forall y : |y| = n^c$ $Pr_r[V(x,y,r) = 1] \leq 1/2$.*

Theorem 6.3 (NTIME Hierarchy [51]). *Let t_1 and t_2 be time constructible functions that satisfy $t_1(n+1) \in o(t_2(n))$. There is a unary language in $\mathsf{NTIME}(t_2(n))$ that is not in $\mathsf{NTIME}(t_1(n))$.*

Now we prove our result. Recall that, a CAPP or tautology algorithm can also solve GAP-SAT.

Theorem 6.4. *For any super-polynomial function sp, an* $\mathsf{NTIME}(2^n/sp(n))$ $\mathsf{GAP\text{-}SAT}$ *algorithm for n-input* $poly(n)$*-size A-oracle circuits, for every* $A \in (\mathsf{NP} \cap \mathsf{Co\text{-}NP})$*, implies* $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$.

Proof. For $L \in \mathsf{NTIME}(2^n)$ we design an $\mathsf{NTIME}(2^n/sp(n))$ algorithm, under the assumption $\mathsf{NEXP} \subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. This will contradict the NTIME hierarchy from Theorem 6.3.

Reduction circuit: Let V be a PCP verifier for L from the Theorem 6.2. On any input x, $V(x, y, r)$ receives $|r| = n + c \log n$ random bits, makes oracle queries to the proof y of size $2^n n^c$, and runs for n^c-time. Let C_x be an oracle circuit capturing this computation. For the oracle gates, we will use copies of the following described easy-witness circuit B_x for a special verifier V'.

Special NE *verifier:* On input x and certificate y, $V'(x, y)$ computes $V(x, y, r)$ on each value of r and outputs 1 if and only if $\forall r \; V(x, y, r) = 1$.

Easy-witness circuit: Since $\mathsf{NEXP} \subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly \implies \mathsf{NEXP} = \mathsf{AM}$, from [41] we get that the search problem for V' is in EXP. Thus, there is an algorithm $\mathcal{A}$ that: on any input $x \in L$ outputs y such that $V'(x, y) = 1$; on any input $x \notin L$ outputs an all zeros string. Now define a new language $L' = \{(x, i) \mid i^{th}$ output bit of $\mathcal{A}$ on input x is $1\}$. $L' \in \mathsf{EXP}$ and thus $L' \in \mathsf{P}^A/poly$ for some $A \in \mathsf{NP} \cap \mathsf{Co\text{-}NP}$. Let B_x be the A-oracle circuit whose truth-table is the witness for V' on input x that is produced by $\mathcal{A}$.

Final circuit F_x*:* $(n + c \log n)$-bits long input r is given to C_x. The oracle gates are replaced by the circuit B_x. The final output is the output of C_x.

Final algorithm: On input x, we get C_x, non-deterministically guess B_x, construct F_x and run the fast $\mathsf{GAP\text{-}SAT}$ algorithm on F_x.

Correctness: The $\mathsf{GAP\text{-}SAT}$ algorithm on F_x checks if the non-deterministic guess B_x satisfies $Pr_r \; [V(x, tt(B_x), r) = 1] \geq 1/2$, or equivalently $V'(x, tt(B_x)) = 1$. If $x \notin L$, this is not possible for any B_x due to the definition of V. If $x \in L$, as argued above, this is true for a poly-size A-oracle circuit B_x that captures witnesses for V'. □

6.3 New gap theorems for CAPP and MA

Results from the previous two sections also gives us gap theorems for CAPP and MA. First we saw that $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$ is equivalent to the derandomization of CAPP for $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$-oracle circuits in $\mathsf{NSUBEXP}$ (infinitely often, with sub-polynomial advice). Then we saw that a non-trivial derandomization is sufficient to prove $\mathsf{NEXP} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. So we get the following gap theorem for CAPP.

Theorem 6.5 (Gap theorem for CAPP on (NP ∩ Co-NP)-oracle circuits). *Let sp be any super-polynomial function, then an* $\mathsf{NTIME}(2^n/sp(n))$ CAPP *algorithm for n-input* $poly(n)$*-size oracle circuits, for every* $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$*-oracle, implies a* $\cap_{\epsilon>0}$ $io\text{-}\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$ *algorithm for n-input* $poly(n)$*-size oracle circuits, for every* $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})$*-oracle.*

In [41] they used NEXP KLT and its converse to establish a gap theorem for MA: either MA is as powerful as NEXP, or can be derandomized in $\mathsf{NSUBEXP}$ (infinitely often, with sub-polynomial advice). From the arguments in Section 6.1 we can get an improved gap theorem where $\mathsf{MA} = \mathsf{EXP}^{\mathsf{NP}}_{||}$ in the first case.

Theorem 6.6 (Gap theorem for MA). *Exactly one of the following statements is true:*

(1) $\mathsf{MA} = \mathsf{EXP}^{\mathsf{NP}}_{||}$

(2) $\mathsf{MA} \subset \cap_{\epsilon>0}$ $io\text{-}\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$

We also get a similar gap theorem for $\mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$: either $\mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$ is as powerful as $\mathsf{EXP}^{\mathsf{NP}}_{||}$, or can be derandomized in $\mathsf{NSUBEXP}$ (infinitely often, with sub-polynomial advice).

Theorem 6.7 (Gap theorem for $\mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$). *Exactly one of the following statements is true:*

(1) $\mathsf{MA}^{\mathsf{NP}\cap\mathsf{Co\text{-}NP}} = \mathsf{EXP}^{\mathsf{NP}}_{||}$
(2) $\mathsf{MA}^{\mathsf{NP}\cap\mathsf{Co\text{-}NP}} \subset \cap_{\epsilon>0}$ io-$\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon$

6.4 Fast algorithms imply lower bounds against circuits with limited prSV non-determinism

Here we show how fast tautology algorithms imply lower bounds for NE and E^{NP}, against circuits that use limited amount of prSV non-determinism.

Theorem 6.8. *For $s(n) \in O(n)$:*

(1) $\mathsf{NE} \subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C} \implies \mathsf{NE}$ *has oblivious witnesses in* $\mathsf{prSV}^{s(n)^{O(1)}}\text{-}\mathcal{C}$
(2) $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C} \implies \mathsf{NE}$ *has oblivious witnesses in* $\mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$

Proof. Let $L \in \mathsf{NE}$, and V be an NE verifier for L.

Proof of (1): Since $\mathsf{NEXP} \subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly \implies \mathsf{NEXP} = \mathsf{AM}$, from [41] we get that the search problem for V is in EXP. Thus, there is an algorithm $\mathcal{A}$ that: on any input $x \in L$ outputs y such that $V(x, y) = 1$; on any input $x \notin L$ outputs an all zeros string. Now define a new language $L' = \{(x, i) \mid i^{th}$ output bit of $\mathcal{A}$ on input x is $1\}$. $L' \in \mathsf{EXP}$ and thus $L' \in \mathsf{prSV}^{s(n)^{O(1)}}\text{-}\mathcal{C}$. The circuit sequence for L' captures oblivious-witnesses for V.

Proof of (2): Let $\mathcal{A}$ that is defined above, output the lexicographically smallest witnesses for V. Then its already known that the corresponding L' language is in E^{NP} (the algorithm does a binary search over all the witnesses, and use the NP-oracle to check if there is any positive witness smaller than the current witness). □

Before giving our main result, we state the local reductions that we will use in our proof.

Theorem 6.9 (Efficient local reductions [31, 50, 89]). *Every language $L \in \mathsf{NTIME}(2^n)$ can be reduced to* 3-SAT *instances of $2^n n^d$-size, for some constant c. Moreover, given an instance of L there is an* P*-uniform deterministic circuit C that, on an integer $i \in [2^n n^d]$ in binary as input, output the i^{th}-clause of the resulting* 3-SAT *formula. Each output bit of C depends on at most d input bits.*

Now we prove our main result.

Theorem 6.10. *For super-polynomial function sp and $s(n) \leq O(n)$:*

(1) *an* $\mathsf{NTIME}(2^{n-s(n)^c}/sp(n))$ $\mathcal{C}$*-tautology algorithm for every $c > 0$ implies* $\mathsf{NE} \not\subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$
(2) *an* $\mathsf{NTIME}(2^{n-3s(n)}/sp(n))$ $\mathcal{C}$*-tautology algorithm implies* $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$

Proof. Assume that $\mathsf{NE} \subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$ or $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$. We contradict the NTIME hierarchy by giving an $\mathsf{NTIME}(2^n/sp(n))$ algorithm for arbitrary $L \in \mathsf{NE}$.

Reduction circuit: From the Theorem 6.9 we get: any input x for L uniformly reduces to a 3-SAT instance ϕ_x, where the number of variables and clauses in ϕ_x are bounded by $n^d 2^n$ for some constant d. Moreover the reduction is local in the sense that: it can be uniformly converted to a deterministic circuit C_x that on $(n + d \log n)$-bits input i outputs the three variables x_{i1}, x_{i2}, x_{i3} ($3n + 3d \log n$ bits) from the i^{th}-clause of ϕ_x, along with three extra bits z_1, z_2, z_3 that indicate for each of these three variables, whether it appears as a positive literal or a negative literal.

Special verifier: Let V be a non-deterministic verifier for L, that first reduces L to 3-SAT, and then non-deterministically guesses a satisfying assignment for the 3-SAT formula.

Witness Circuits B_x: We construct two witness circuits (after guessing the advice of the $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$ algorithm $\mathcal{A}$ that has V's oblivious-witnesses): one non-deterministic B^1_x, and one co-non-deterministic B^2_x.

Final Circuit F_x: Take the reduction circuit C_x. C_x outputs three literals. Plug any positive literal into a copy of the co-non-deterministic circuit B^2_x, and any negative literal into a copy of the co-non-deterministic circuit $\overline{B_x^1}$. Output is the logical-or of the three copies used. To make the circuit deterministic, include the non-deterministic inputs of the copies of B_x^2 and $\neg B_x^1$ into the actual input.

Final algorithm: Get C_x. Non-deterministically guess the advice for $\mathcal{A}$, and get B_x^1 and B_x^2 (that are guaranteed to have complementary truth-tables). Construct the deterministic circuit F_x as described above. Run the the fast TAUT algorithm on F_x.

Correctness: $x \in L \iff F_x$ is a tautology. The tautology algorithm on F_x checks if the pair (B_x^1, B_x^2) satisfy, $V(x, tt(B_x^1)) = 1$. If $x \notin L$, this is not possible for any B_x^1 and B_x^2. If $x \in L$, this is true for the witness circuits that exists due the easy-witness lemma proved in the above theorem. While constructing F_x, we use the fact that tautology of a co-non-deterministic circuit, is same as the tautology of the deterministic circuit we get after including the non-deterministic inputs into the actual input.

Final contradiction: The final input size is increased by $s(n)^c + O(\log n)$ if we use the EWL from assumption $\mathsf{NE} \not\subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$, and is increased by $3s(n)+O(\log n)$ if we use the EWL from assumption $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{prSV}^{s(n)}\text{-}\mathcal{C}$. So algorithms from our assumptions are fast enough to contradict the NTIME hierarchy. □

6.5 Uncodntional lower bounds against restricted prSV non-deterministic circuits

Using the Theorem 6.10 from previous section we get uncondtional lower bounds against restricted circuits that use limited prSV non-determinism. The following theorem follows from $\mathsf{TIME}(2^{n-n^{\epsilon}})$ tautology algorithm for ACC circuits [99], where the constant ϵ depends on the depth and the modulus function used by the circuits.

Theorem 6.11. $\mathsf{NE} \not\subset \cap_{\epsilon>0}\mathsf{prSV}^{n^{\epsilon}}\text{-}\mathsf{ACC}$

The following theorem follows from the $\mathsf{ZPTIME}(2^{n(1-1/(\log n)^{\epsilon})})$ tautology algorithm for AC^0 circuits [42], where the constant ϵ increases as the size or depth of the circuits increases.

Theorem 6.12. $\mathsf{E}^{\mathsf{NP}} \not\subset \cap_{\epsilon>0}\mathsf{prSV}^{n/(\log n)^{\epsilon}}\text{-}\mathsf{AC}^0$

In the proof of the Theorem 6.10, the final circuit is constructed by giving the output bits of the reduction circuit as input to the witness circuit. Each output bit of the reduction circuit of Theorem 6.9 only depends on constant number of inputs, so can be represented by a set of constant-width clauses or terms, and thus can be plugged without increasing the depth. Thus the depth of the final circuit is only increased by the top OR-gate. For the case of E^{NP}, the final circuit also preserves the size of the witness circuit upto a constant factor. So we get the following result using fast AC^0 algorithms for different size and depth ranges [42].

Theorem 6.13. $\mathsf{E}^{\mathsf{NP}} \not\subset \cap_{\epsilon>0}\mathsf{prSV}^{\epsilon n/(\log n)^2}\text{-}k\text{-}\mathsf{CNF}$ *and* $\mathsf{E}^{\mathsf{NP}} \not\subset \cap_{\epsilon>0}\mathsf{prSV}^{\epsilon n}\text{-}\mathsf{AC}^0(n)$

Note that, $O(n)$ amount of prSV non-determinism in any of the above two lower bounds, will give super-linear lower bounds for E^{NP}.

6.6 Unconditional lower bounds against unrestricted fixed-polynomial prSV non-determinsitic circuits

Here we give unconditional lower bounds for prAM against fixed-polynomial size prSV non-deterministic circuits. We use the following PSPACE-complete language of Santhanam [79], which was also a crucial technical step in his celebrated MA lower-bound.

Lemma 6.14. *There is a* PSPACE*-complete language L^S and probabilistic polynomial-time oracle Turing machines M and M' such that the following holds for any n-length input x:*

(1) *M and M' only query their oracle on strings of length n.*
(2) *If M (resp. M') is given L^S as its oracle and $x \in L^S$ (resp. $x \notin L^S$), then M (resp. M') accepts with probability 1.*
(3) *If $x \notin L^S$ (resp. $x \in L^S$), then irrespective of the oracle, M (resp. M') rejects with probability at least 2/3.*

Like Santhanam's proof, our proof is also split into two cases: (i) The easier case where $\mathsf{PSPACE} \subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, we use the KLT from [16]. (ii) The difficult case where PSPACE doesn't have poly-size prSV circuits, we design AM protocol for a padded version of L^S that doesn't have fixed-polynomial prSV circuits. We first prove an auxiliary lemma that we use for the second case.

Lemma 6.15. *For $k \geq 1$ and super-constant function sc, using L^S from Lemma 6.14 we define:*

$$L^k = \{x1^y \mid x \in L^S \wedge \exists(z \in \mathbb{N})\ y = 2^z \geq |x| > 0, (2y + |x|)^{k+1} \geq ckt^{sc}_{\mathsf{prSV}}(L^S_{|x|}) > (y + |x|)^{k+1}\}$$

If $\mathsf{PSPACE} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, then $L^k \notin \mathsf{prSVNSIZE}(n^k)$ for every $k \geq 1$.

Proof. For the sake of contradiction, let's assume that $L^k \in \mathsf{prSVN}(n^k)$. That means, there is a prSVN algorithm $\mathcal{A}$, that produces an n^k-size SV non-deterministic circuit sequence, that decides L^k. We modify this sequence to yield a sequence for L^S (used in the definition of L^k). Any input length n can be broken into unique m and $y = 2^z$ such that $y \geq m$ and $m + y = n$. If y satisfies $(2y + m)^{k+1} \geq ckt^{sc}_{\mathsf{prSV}}(L^S_m) \geq (y + m)^{k+1}$, then a circuit for the n^{th}-slice of L^k can be used to yield a circuit for the m^{th}-slice of L^S (by fixing the last y input bits to 1s). Moreover for any m, there is a unique y that satisfies $(2y + m)^{k+1} \geq ckt^{sc}_{\mathsf{prSV}}(L^S_m) \geq (y + m)^{k+1}$ (since y is a power of 2).

For any input length m, the size of the circuit from this sequence will be $(m + y)^k$ for the unique y that is paired with m. This leads to the contradiction $(m + y)^k \geq ckt_{\mathsf{prSV}(\mathcal{A})}(L^S_m) \geq ckt^{sc}_{\mathsf{prSV}}(L^S_m) > (m+y)^{k+1}$ on input lengths m where L^S requires more that m^{k+1} size (due to $L^S \notin (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$) and thus a positive y exits. The first inequality follows from the fact that the circuit sequence is produced by $\mathcal{A}$. The second inequality uses the fact that the measure ckt^{sc}_{prSV}, beats the measure $ckt_{\mathsf{prSV}(\mathcal{A})}$ for any prSVN algorithm $\mathcal{A}$, after a certain input length (because $\mathcal{A}$'s description is only of constant length, i.e. less than $sc(m)$). The third inequality follows from the definition of L^k. □

Now we prove one of the two main results of this section.

Theorem 6.16. *For any super-constant function sc, $\forall k\ \mathsf{AM}/sc(n) \not\subset \mathsf{prSVNSIZE}(n^k)$.*

Proof. If $\mathsf{PSPACE} \subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$, then $\mathsf{PSPACE} = \mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$. As in PSPACE we can diagonalize against any fixed-polynomial size circuit class, we get the desired fixed-polynomial circuit lower-bound for $\mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$ (without any advice). $\mathsf{MA}^{\mathsf{NP} \cap \mathsf{Co\text{-}NP}}$ is contained $\mathsf{MAM} = \mathsf{AM}$ (Idea: after Arthur guesses its random bits, it sends them to Merlin, who then computes all the $\mathsf{NP} \cap \mathsf{Co\text{-}NP}$ queries Arthur will make, and sends Arthur the replies along with the certificates for the queries and their compliments).

If $\mathsf{PSPACE} \not\subset (\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$. From the Lemma 6.15 we get languages (L^k for $k \geq 1$) with the desired lower bounds. We design $\mathsf{AM}/sc(n)$ protocols for these languages. Arthur rejects everything if the first advice bit is 0. The first advice bit is 1 exactly for the input lengths n that split into valid (m, y) pairs (see the proof of the Lemma 6.15 for the notion of valid pairs), when L^k is defined using the measure $ckt^{sc-1}_{\mathsf{prSV}}$. Arthur rejects if the input is not in the format $x1^y$. Else, it simulates the machine M from the Lemma 6.14 to check if $x \in L^S$ or not. It accepts if and only if $x \in L^S$. It uses the circuit C, that it computes from Merlin's reply and the rest of the $sc(n) - 1$ bits of advice, as an oracle to M (from the Lemma 6.14).

The last $sc(n) - 1$ bits of advice encodes a prSVN algorithm $\mathcal{A}$. Correct advice encodes the most efficient one of the most efficient prSVN algorithms for that input length, i.e. an algorithm

$\mathcal{A}$ such that $ckt_{\mathsf{SV}(\mathcal{A})}(L^S_{|x|}) = ckt_{\mathsf{prSV}}(L^S_{|x|})$. For n-length input $x1^y$ with $|x| = m$, Merlin sends an $(2y+m)^{k+1}$-length input w for $\mathcal{A}$. Arthur produces the circuit $C = \mathcal{A}(w)$ to use as an oracle for M. Arthur then guesses random bits for the simulation of M and sends them to Merlin. Merlin in return sends the certificates that sets the flag bit of C to 1, on all the queries that M makes to C. Arthur uses these certificates, to compute the value bits of C, and thus successfully simulates M on x (using C as oracle).

Completeness follows easily. If $x \in L^S$, Merlin can send the input on which the algorithm $\mathcal{A}$ outputs the correct SV circuit for L^S. If $x \notin L^S$, soundness follows from the fact that the algorithm $\mathcal{A}$ always generates SV circuits, and thus the oracle used by M is consistent (to some language), and thus M rejects with probability at least 2/3. □

For each input length, assigning multiple input lengths corresponding to each possible advice, and making the input lengths with the correct advice as the promise input lengths, we get the following theorem.

Theorem 6.17. $\forall k$ $\mathsf{prAM} \not\subset \mathsf{prSVNSIZE}(n^k)$.

7 CONCLUSIONS AND OPEN PROBLEMS

The main open problem is whether there are any connections between fast algorithms and non-uniform lower bounds possible within deterministic classes such as 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 weakened. In what other ways could we get such connections for smaller classes by restricting the use of non-determinism? The circuit model combines two features: time and non-uniformity. Can we get a fine-grained version of easy-witness lemma by distinguishing these two parameters?

Next obvious question in this line is whether we can get lower bounds for UEXP and related classes using our connections. Designing fast algorithms is one direct strategy. One other, seemingly easier strategy is to design tight hierarchy theorems for these semantic classes, possibly under the assumption that they have small circuits.

Our results also show that, if we are using unrestricted non-determinism to simulate non-uniformity, we can extract more out of it. That is, the guessed circuit is also allowed to use non-determinism that is promise-single-valued. In what other ways can we extend this allowance? Can we remove the promise condition? Specifically, can we prove NEXP easy-witness lemmas and Karp-Lipton theorems for circuit classes above $(\mathsf{NP} \cap \mathsf{Co\text{-}NP})/poly$?

We also show unconditional NEXP lower bounds where sub-polynomial and sub-linear amounts of promise-single-valued non-determinism is allowed. Can we increase the amount of non-determinism allowed, to polynomial or linear? Designing fast algorithms is one direct strategy. Can we do it without changing the satisfiability upper bounds? This would lead to super-linear and super-polynomial lower bounds against unrestricted Boolean circuits. Or can we get lower bounds against TC^0 circuits by simulating threshold gates, by the use of less expressive gates and limited non-determinism?

ACKNOWLEDGMENTS

This work is supported by the Simons Foundation and NSF grant CCF-1909634. We want to thank Sasank Mouli, Vaibhav Krishan, Marco Carmosino and Sam McGuire for useful discussions, and for comments and corrections on the manuscript.

REFERENCES

[1] Scott Aaronson and Avi Wigderson. 2009. Algebrization: A New Barrier in Complexity Theory. *ACM Trans. Comput. Theory* 1, 1 (2009), 2:1–2:54. https://doi.org/10.1145/1490270.1490272

[2] Miklós Ajtai. 1983. $\sum^1{}_1$-Formulae on finite structures. *Ann. Pure Appl. Log.* 24, 1 (1983), 1–48. https://doi.org/10.1016/0168-0072(83)90038-6

[3] Eric Allender. 1986. The Complexity of Sparse Sets in P. In *Structure in Complexity Theory, Proceedings of the Conference hold at the University of California, Berkeley, California, USA, June 2-5, 1986.* 1–11. https://doi.org/10.1007/3-540-16486-3_85

[4] Eric Allender. 2001. When Worlds Collide: Derandomization, Lower Bounds, and Kolmogorov Complexity. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science*, Ramesh Hariharan, V. Vinay, and Madhavan Mukund (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–15.

[5] N Alon and RB Boppana. 1987. The monotone circuit complexity of boolean functions. *Combinatorica* 7, 1 (1987), 1–22.

[6] Dana Angluin. 1987. Queries and Concept Learning. *Machine Learning* 2, 4 (1987), 319–342. https://doi.org/10.1007/BF00116828

[7] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach* (1st ed.). Cambridge University Press, New York, NY, USA.

[8] Baris Aydinlioglu and Dieter van Melkebeek. 2017. Nondeterministic circuit lower bounds from mildly derandomizing Arthur-Merlin games. *Computational Complexity* 26, 1 (2017), 79–118. https://doi.org/10.1007/s00037-014-0095-y

[9] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. 1993. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity* 3 (1993), 307–318. https://doi.org/10.1007/BF01275486

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

[11] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. 2005. Short PCPs Verifiable in Polylogarithmic Time. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*. IEEE Computer Society, 120–134. https://doi.org/10.1109/CCC.2005.27

[12] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. 1996. Oracles and Queries That Are Sufficient for Exact Learning. *J. Comput. Syst. Sci.* 52, 3 (1996), 421–433. https://doi.org/10.1006/jcss.1996.0032

[13] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. 1998. Nonrelativizing Separations. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity, Buffalo, New York, USA, June 15-18, 1998.* IEEE Computer Society, 8–12. https://doi.org/10.1109/CCC.1998.694585

[14] Harry Buhrman and Steven Homer. 1992. Superpolynomial Circuits, Almost Sparse Oracles and the Exponential Hierarchy. In *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings (Lecture Notes in Computer Science, Vol. 652)*, R. K. Shyamasundar (Ed.). Springer, 116–127. https://doi.org/10.1007/3-540-56287-7_99

[15] Jin-yi Cai. 2001. $S^P{}_2$ subseteq ZPP^{NP}. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 620–629. https://doi.org/10.1109/SFCS.2001.959938

[16] Jin-Yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. 2005. Competing provers yield improved Karp–Lipton collapse results. *Information and Computation* 198, 1 (2005), 1 – 23. https://doi.org/10.1016/j.ic.2005.01.002

[17] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. 2006. A Duality between Clause Width and Clause Density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic.* IEEE Computer Society, 252–260. https://doi.org/10.1109/CCC.2006.6

[18] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. 2009. The Complexity of Satisfiability of Small Depth Circuits. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 5917)*, Jianer Chen and Fedor V. Fomin (Eds.). Springer, 75–85. https://doi.org/10.1007/978-3-642-11269-0_6

[19] Marco Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. 2015. 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 (LIPIcs, Vol. 40)*, Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 645–658. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2015.645

[20] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. 2016. 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*, Madhu Sudan (Ed.). ACM, 261–270. https://doi.org/10.1145/2840728.2840746

[21] Brynmor Chapman and Ryan Williams. 2015. The Circuit-Input Game, Natural Proofs, and Testing Circuits With Data. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science* (Rehovot, Israel) *(ITCS '15)*. Association for Computing Machinery, New York, NY, USA, 263–270. https://doi.org/10.1145/2688073.2688115

[22] Lijie Chen, Ron Rothblum, Roei Tell, and Eylon Yogev. 2019. On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds. *Electron. Colloquium Comput. Complex.* 26 (2019), 169. https://eccc.weizmann.ac.il/report/2019/169

[23] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. 2015. Mining Circuit Lower Bound Proofs for Meta-Algorithms. *Comput. Complex.* 24, 2 (2015), 333–392. https://doi.org/10.1007/s00037-015-0100-0

[24] Timothy Y. Chow. 2011. Almost-natural proofs. *J. Comput. System Sci.* 77, 4 (2011), 728 – 737. https://doi.org/10.1016/j.jcss.2010.06.017 JCSS IEEE AINA 2009.

[25] Stephen A. Cook. 1971. 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*. 151–158. https://doi.org/10.1145/800157.805047

[26] Stephen A. Cook. 1972. A Hierarchy for Nondeterministic Time Complexity. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg (Eds.). ACM, 187–192. https://doi.org/10.1145/800152.804913

[27] Holger Dell, Valentine Kabanets, Dieter van Melkebeek, and Osamu Watanabe. 2013. Is Valiant-Vazirani's isolation probability improvable? *Computational Complexity* 22, 2 (2013), 345–383. https://doi.org/10.1007/s00037-013-0059-7

[28] Joan Feigenbaum and Lance Fortnow. 1993. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.* 22, 5 (1993), 994–1005. https://doi.org/10.1137/0222061

[29] Lance Fortnow and Adam R. Klivans. 2005. NP with Small Advice. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*. IEEE Computer Society, 228–234. https://doi.org/10.1109/CCC.2005.15

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

[31] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. 2005. Time-space Lower Bounds for Satisfiability. *J. ACM* 52, 6 (Nov. 2005), 835–865. https://doi.org/10.1145/1101821.1101822

[32] Lance Fortnow and Rahul Santhanam. 2011. Robust Simulations and Significant Separations. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 6755)*, Luca Aceto, Monika Henzinger, and Jirí Sgall (Eds.). Springer, 569–580. https://doi.org/10.1007/978-3-642-22006-7_48

[33] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. 2005. Hierarchies for Semantic Classes. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing* (Baltimore, MD, USA) *(STOC '05)*. ACM, New York, NY, USA, 348–355. https://doi.org/10.1145/1060590.1060642

[34] Merrick L. Furst, James B. Saxe, and Michael Sipser. 1984. Parity, Circuits, and the Polynomial-Time Hierarchy. *Math. Syst. Theory* 17, 1 (1984), 13–27. https://doi.org/10.1007/BF01744431

[35] Ryan C. Harkins and John M. Hitchcock. 2013. Exact Learning Algorithms, Betting Games, and Circuit Lower Bounds. *TOCT* 5, 4 (2013), 18:1–18:11. https://doi.org/10.1145/2539126.2539130

[36] J. Hartmanis and R. E. Stearns. 1965. On the Computational Complexity of Algorithms. *Trans. Amer. Math. Soc.* 117 (1965), 285–306. http://www.jstor.org/stable/1994208

[37] Johan Håstad. 1986. Almost Optimal Lower Bounds for Small Depth Circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, Juris Hartmanis (Ed.). ACM, 6–20. https://doi.org/10.1145/12130.12132

[38] F. C. Hennie and Richard Edwin Stearns. 1966. Two-Tape Simulation of Multitape Turing Machines. *J. ACM* 13, 4 (1966), 533–546. https://doi.org/10.1145/321356.321362

[39] Timon Hertli. 2011. 3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS '11)*. IEEE Computer Society, USA, 277–284. https://doi.org/10.1109/FOCS.2011.22

[40] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. 2018. The Power of Natural Properties as Oracles. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA (LIPIcs, Vol. 102)*, Rocco A. Servedio (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:20. https://doi.org/10.4230/LIPIcs.CCC.2018.7

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

[42] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. 2012. A satisfiability algorithm for AC^0. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, Yuval Rabani (Ed.). SIAM, 961–972. https://doi.org/10.1137/1.9781611973099.77

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

[44] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. 2013. A Satisfiability Algorithm for Sparse Depth Two Threshold Circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. IEEE Computer Society, 479–488. https://doi.org/10.1109/FOCS.2013.58

[45] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 1998. 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*. 653–663. https://doi.org/10.1109/SFCS.1998.743516

[46] Russell Impagliazzo and Gábor Tardos. 1989. Decision Versus Search Problems in Super-Polynomial Time. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*. IEEE Computer Society, 222–227. https://doi.org/10.1109/SFCS.1989.63482

[47] Russell Impagliazzo and Avi Wigderson. 1997. $P = BPP$ 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*. 220–229. https://doi.org/10.1145/258533.258590

[48] Russell Impagliazzo and Avi Wigderson. 2001. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.* 63, 4 (2001), 672–688. https://doi.org/10.1006/jcss.2001.1780

[49] Kazuo Iwama and Hiroki Morizumi. 2002. An Explicit Lower Bound of 5n - o(n) for Boolean Circuits. In *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2420)*, Krzysztof Diks and Wojciech Rytter (Eds.). Springer, 353–364. https://doi.org/10.1007/3-540-45687-2_29

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

[51] Stanislav Žák. 1983. A Turing machine time hierarchy. *Theoretical Computer Science* 26, 3 (1983), 327 – 333. https://doi.org/10.1016/0304-3975(83)90015-4

[52] Valentine Kabanets. 2001. Easiness Assumptions and Hardness Tests: Trading Time for Zero Error. *J. Comput. Syst. Sci.* 63, 2 (2001), 236–252. https://doi.org/10.1006/jcss.2001.1763

[53] Valentine Kabanets and Russell Impagliazzo. 2004. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity* 13, 1-2 (2004), 1–46. https://doi.org/10.1007/s00037-004-0182-6

[54] Ravi Kannan. 1982. Circuit-Size Lower Bounds and Non-Reducibility to Sparse Sets. *Information and Control* 55, 1-3 (1982), 40–56. https://doi.org/10.1016/S0019-9958(82)90382-5

[55] Richard M. Karp and Richard J. Lipton. 1980. Some Connections Between Nonuniform and Uniform Complexity Classes. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing* (Los Angeles, California, USA) *(STOC '80)*. ACM, New York, NY, USA, 302–309. https://doi.org/10.1145/800141.804678

[56] Johannes Köbler and Osamu Watanabe. 1998. New Collapse Consequences of NP Having Small Circuits. *SIAM J. Comput.* 28, 1 (1998), 311–324. https://doi.org/10.1137/S0097539795296206 arXiv:https://doi.org/10.1137/S0097539795296206

[57] Adam R. Klivans, Pravesh Kothari, and Igor Carboni Oliveira. 2013. 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*. 86–97. https://doi.org/10.1109/CCC.2013.18

[58] Adam R. Klivans and Dieter van Melkebeek. 2002. Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses. *SIAM J. Comput.* 31, 5 (2002), 1501–1526. https://doi.org/10.1137/S0097539700389652

[59] Matthias Krause and Stefan Lucks. 2002. Pseudorandom Functions in TC0 and Cryptographic Limitations to Proving Lower Bounds. *Comput. Complex.* 10, 4 (May 2002), 297–313. https://doi.org/10.1007/s000370100002

[60] Oded Lachish and Ran Raz. 2001. Explicit Lower Bound of 4.5n - o(n) for Boolena Circuits. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing* (Hersonissos, Greece) *(STOC '01)*. Association for Computing Machinery, New York, NY, USA, 399–408. https://doi.org/10.1145/380752.380832

[61] Leonid A. Levin. 1973. Universal sorting problems. *Problems of Information Transmission* 9 (1973), 265–266.

[62] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. 1992. Algebraic Methods for Interactive Proof Systems. *J. ACM* 39, 4 (1992), 859–868. https://doi.org/10.1145/146585.146605

[63] Kazuhisa Makino, Suguru Tamaki, and Masaki Yamamoto. 2013. Derandomizing the HSSW Algorithm for 3-SAT. *Algorithmica* 67, 2 (2013), 112–124. https://doi.org/10.1007/s00453-012-9741-4

[64] Eric Miles and Emanuele Viola. 2012. Substitution-Permutation Networks, Pseudorandom Functions, and Natural Proofs. In *Advances in Cryptology – CRYPTO 2012*, Reihaneh Safavi-Naini and Ran Canetti (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 68–85.

[65] Burkhard Monien and Ewald Speckenmeyer. 1985. Solving satisfiability in less than 2^n steps. *Discret. Appl. Math.* 10, 3 (1985), 287–295. https://doi.org/10.1016/0166-218X(85)90050-2

[66] Hiroki Morizumi. 2015. Lower Bounds for the Size of Nondeterministic Circuits. In *Computing and Combinatorics*, Dachuan Xu, Donglei Du, and Dingzhu Du (Eds.). Springer International Publishing, Cham, 289–296.

[67] Robin A. Moser and Dominik Scheder. 2011. A full derandomization of schöning's k-SAT algorithm. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, Lance Fortnow and Salil P. Vadhan (Eds.). ACM, 245–252. https://doi.org/10.1145/1993636.1993670

[68] Cody Murray and R. Ryan Williams. 2018. 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.* 890–901. https://doi.org/10.1145/3188745.3188910

[69] Moni Naor and Omer Reingold. 2004. Number-theoretic Constructions of Efficient Pseudo-random Functions. *J. ACM* 51, 2 (March 2004), 231–262. https://doi.org/10.1145/972639.972643

[70] Noam Nisan. 1991. Pseudorandom bits for constant depth circuits. *Combinatorica* 11, 1 (1991), 63–70. https://doi.org/10.1007/BF01375474

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

[72] Igor Carboni Oliveira. 2013. Algorithms versus Circuit Lower Bounds. *CoRR* abs/1309.0249 (2013). arXiv:1309.0249 http://arxiv.org/abs/1309.0249

[73] Igor Carboni Oliveira and Rahul Santhanam. 2017. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia.* 18:1–18:49. https://doi.org/10.4230/LIPIcs.CCC.2017.18

[74] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. 2005. An improved exponential-time algorithm for k-SAT. *J. ACM* 52, 3 (2005), 337–364. https://doi.org/10.1145/1066100.1066101

[75] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. 1999. Satisfiability Coding Lemma. *Chicago J. Theor. Comput. Sci.* 1999 (1999). http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html

[76] A. A. RAZBOROV. 1985. Lower bounds for the monotone complexity of some Boolean functions. *Soviet Math. Dokl.* 31 (1985), 354–357. https://ci.nii.ac.jp/naid/10003041149/en/

[77] Alexander A. Razborov. 1989. On the Method of Approximations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, David S. Johnson (Ed.). ACM, 167–176. https://doi.org/10.1145/73007.73023

[78] Alexander A Razborov and Steven Rudich. 1997. Natural Proofs. *J. Comput. System Sci.* 55, 1 (1997), 24 – 35. https://doi.org/10.1006/jcss.1997.1494

[79] Rahul Santhanam. 2009. Circuit Lower Bounds for Merlin–Arthur Classes. *SIAM J. Comput.* 39, 3 (2009), 1038–1061. https://doi.org/10.1137/070702680

[80] Rahul Santhanam. 2010. Fighting Perebor: New and Improved Algorithms for Formula and QBF Satisfiability. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA.* IEEE Computer Society, 183–192. https://doi.org/10.1109/FOCS.2010.25

[81] Uwe Schöning. 1999. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*. IEEE Computer Society, 410–414. https://doi.org/10.1109/SFFCS.1999.814612

[82] Rainer Schuler. 2005. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms* 54, 1 (2005), 40–44. https://doi.org/10.1016/j.jalgor.2004.04.012

[83] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. 1978. Separating Nondeterministic Time Complexity Classes. *J. ACM* 25, 1 (1978), 146–167. https://doi.org/10.1145/322047.322061

[84] Kazuhisa Seto and Suguru Tamaki. 2012. A Satisfiability Algorithm and Average-Case Hardness for Formulas over the Full Binary Basis. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012.* IEEE Computer Society, 107–116. https://doi.org/10.1109/CCC.2012.29

[85] Ronen Shaltiel and Christopher Umans. 2006. Pseudorandomness for Approximate Counting and Sampling. *Computational Complexity* 15, 4 (2006), 298–341. https://doi.org/10.1007/s00037-007-0218-9

[86] Roman Smolensky. 1987. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, Alfred V. Aho (Ed.). ACM, 77–82. https://doi.org/10.1145/28395.28404

[87] Donald M. Stull. 2017. Some Results on Circuit Lower Bounds and Derandomization of Arthur-Merlin Problems. *CoRR* abs/1701.04428 (2017). arXiv:1701.04428 http://arxiv.org/abs/1701.04428

[88] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. 2001. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.* 62, 2 (2001), 236–266. https://doi.org/10.1006/jcss.2000.1730

[89] Iannis Tourlakis. 2001. Time–Space Tradeoffs for SAT on Nonuniform Machines. *J. Comput. System Sci.* 63, 2 (2001), 268 – 287. https://doi.org/10.1006/jcss.2001.1767

[90] G. S. Tseitin. 1983. *On the Complexity of Derivation in Propositional Calculus*. Springer Berlin Heidelberg, Berlin, Heidelberg, 466–483. https://doi.org/10.1007/978-3-642-81955-1_28

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

[92] Leslie G. Valiant. 1984. A Theory of the Learnable. *Commun. ACM* 27, 11 (1984), 1134–1142. https://doi.org/10.1145/1968.1972

[93] Leslie G. Valiant and Vijay V. Vazirani. 1986. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.* 47, 3 (1986), 85–93. https://doi.org/10.1016/0304-3975(86)90135-0

[94] N.V. Vinodchandran. 2005. A note on the circuit complexity of PP. *Theoretical Computer Science* 347, 1 (2005), 415 – 418. https://doi.org/10.1016/j.tcs.2005.07.032

[95] N. V. Vinodchandran. 2004. AM_{exp}[nsube](NP[cap]coNP)/poly. *Inf. Process. Lett.* 89, 1 (2004), 43–47. https://doi.org/10.1016/j.ipl.2003.09.011

[96] Nikhil Vyas and Ryan Williams. 2019. On Super Strong ETH. In *Theory and Applications of Satisfiability Testing – SAT 2019*, Mikoláš Janota and Inês Lynce (Eds.). Springer International Publishing, Cham, 406–423.

[97] Nikhil Vyas and R. Ryan Williams. 2020. Lower Bounds Against Sparse Symmetric Functions of ACC Circuits: Expanding the Reach of #SAT Algorithms. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France (LIPIcs, Vol. 154)*, Christophe Paul and Markus Bläser (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 59:1–59:17. https://doi.org/10.4230/LIPIcs.STACS.2020.59

[98] Ryan Williams. 2013. Improving Exhaustive Search Implies Superpolynomial Lower Bounds. *SIAM J. Comput.* 42, 3 (2013), 1218–1244. https://doi.org/10.1137/10080703X

[99] Ryan Williams. 2014. Nonuniform ACC Circuit Lower Bounds. *J. ACM* 61, 1 (2014), 2:1–2:32. https://doi.org/10.1145/2559903

[100] R. Ryan Williams. 2016. Natural Proofs versus Derandomization. *SIAM J. Comput.* 45, 2 (2016), 497–529. https://doi.org/10.1137/130938219

[101] R. Ryan Williams. 2018. New Algorithms and Lower Bounds for Circuits With Linear Threshold Gates. *Theory of Computing* 14, 1 (2018), 1–25. https://doi.org/10.4086/toc.2018.v014a017

[102] Andrew Chi-Chih Yao. 1985. Separating the Polynomial-Time Hierarchy by Oracles (Preliminary Version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*. IEEE Computer Society, 1–10. https://doi.org/10.1109/SFCS.1985.49

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il