Electronic Colloquium on Computational Complexity, Report No. 188 (2017)

Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP

Cody D. Murray
MIT

R. Ryan Williams[*]
MIT

Abstract

We prove that if every problem in NP has n^k-size circuits for a fixed constant k, then for every NP-verifier and every yes-instance x of length n for that verifier, the verifier's search space has an $n^{O(k^3)}$-size *witness circuit*: a witness for x that can be encoded with a circuit of only $n^{O(k^3)}$ size. An analogous statement is proved for nondeterministic quasi-polynomial time, i.e., $\mathsf{NQP} = \mathsf{NTIME}[n^{\log^{O(1)} n}]$. This significantly extends the Easy Witness Lemma of Impagliazzo, Kabanets, and Wigderson [JCSS'02] which only held for larger nondeterministic classes such as NEXP.

As a consequence, the connections between circuit-analysis algorithms and circuit lower bounds can be considerably sharpened: algorithms for approximately counting satisfying assignments to given circuits which improve over exhaustive search can imply circuit lower bounds for functions in NQP, or even NP. To illustrate, applying known algorithms for satisfiability of $\mathsf{ACC} \circ \mathsf{THR}$ circuits [R. Williams, STOC 2014] we conclude that for every fixed k, NQP does not have $n^{\log^k n}$-size $\mathsf{ACC} \circ \mathsf{THR}$ circuits.

[*]Supported by NSF CAREER award CCF-1552651.

ISSN 1433-8092

1 Introduction

Proving non-uniform circuit lower bounds for functions in nondeterministic complexity classes such as NP or $\mathsf{NEXP} = \mathsf{NTIME}[2^{n^{O(1)}}]$ is a well-known grand challenge. A natural approach is to assume that a given complexity class has small circuits, and to derive unlikely consequences, eventually absurd enough to derive a contradiction with other results. Over the years, interesting consequences have been derived from assuming (for example) NP has $O(n)$-size circuits, or NEXP has polynomial-size circuits; for a sample, see [KL82, Kan82, Lip94, KW98, IKW02, FSW09, Din15, Wil16].

One of the most striking results in this line of work is the *Easy Witness Lemma* of Impagliazzo, Kabanets, and Wigderson [IKW02]. Informally, they show that if NEXP has small circuits, then every yes-instance of every NEXP problem has a highly *compressible* "easy" witness:

Lemma 1.1 (Easy Witness Lemma [IKW02] (Informal)) *If $\mathsf{NTIME}[2^n]$ has circuits of size $s(n)$, then there is a k such that every yes-instance for every verifier for every problem in $\mathsf{NTIME}[2^n]$ has a witness string w (of $O(2^n)$ length) which can be encoded by a circuit of size at most $s(s(n^k)^k)^k$. In particular, viewing w as the truth table of a function $f_w : \{0,1\}^{n+O(1)} \to \{0,1\}$, this f_w has $s(s(n^k)^k)^k$-size circuits.*

For $s(n)$ being polynomial or quasipolynomial (i.e., $s(n) = 2^{\log^{O(1)} n}$), Lemma 1.1 says that small circuits for NEXP imply small circuits encoding *solutions* to NEXP problems. Informally, we say the consequence of Lemma 1.1 means $\mathsf{NTIME}[2^n]$ has *small witness circuits*. Note an NEXP search problem has $2^{2^{\mathrm{poly}(n)}}$ possible solutions; having small witness circuits means that the existence of some witness in this huge space implies an "easy" witness with a short description, in a much smaller search space. Furthemore, if $\mathsf{NTIME}[2^n]$ has poly-size witness circuits, then $\mathsf{NEXP} = \mathsf{EXP}$, by enumerating over all $2^{\mathrm{poly}(n)}$ small circuits and checking if any encode a witness for the given verifier. Thus Lemma 1.1 shows $\mathsf{NEXP} \subset \mathsf{P}/\mathrm{poly}$ implies $\mathsf{NEXP} = \mathsf{EXP}$. Impagliazzo, Kabanets, and Wigderson found many other applications, including a surprising equivalence between $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ and non-trivial derandomizations.

Application: NEXP Lower Bounds. The second author [Wil11, Wil14] used the Easy Witness Lemma (along with other ideas) to prove NEXP does not have $n^{\log^{O(1)} n}$-size ACC circuits (with the polylog factors tightened in [CP16]), and NEXP does not have $n^{\log^{O(1)} n}$-size $\mathsf{ACC} \circ \mathsf{THR}$ circuits (ACC composed with a layer of linear threshold gates at the bottom). These results were proved by designing faster-than-2^n algorithms for the SAT problem for ACC circuits, and using the algorithms in an involved argument with the Easy Witness Lemma which (eventually) applies the nondeterministic time hierarchy [Wil10]. In general, improving the time complexity of CIRCUIT SAT (or even estimating the fraction of satisfying assignments to within a small constant, which is widely believed to be in P) for a "typical" complexity class $\mathscr{C}$ (such as ACC, TC^0, NC^1, NC, P/poly) from $2^n \cdot s^{O(1)}$ time to (for example) $2^n \cdot s^{O(1)}/n^{\mathrm{poly}(\log n)}$ time would imply NEXP does not have $\mathscr{C}$-circuits of $n^{\log n}$ size [Wil10, SW13, BSV14]. This algorithmic approach to proving circuit lower bounds via weak circuit satisfiability algorithms has led to some optimism that lower bounds like $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ may be approachable, and provable.

The Quest for Lower-Complexity Functions. While this program for proving lower bounds has seen some successes, a major deficiency is that it can only yield circuit lower bounds for functions in $\mathsf{NTIME}[2^{n^{\varepsilon}}]$ and higher, for $\varepsilon > 0$. As NEXP is presumably an enormous class, results such as $\mathsf{NEXP} \not\subset \mathsf{ACC}$ are far less interesting than proving that a more explicit function, such as SAT, is not in ACC. A key bottleneck in the arguments is the Easy Witness Lemma: Lemma 1.1 is crucial in the connection between SAT algorithms and lower bounds, but its proof only works for nondeterministic *exponential-time* classes.[1] Given an Easy Witness Lemma for NP (i.e., *if* NP *has* n^k*-size circuits, then every yes-instance of every* NP *problem has* $n^{f(k)}$*-size witness circuits*), or for $\mathsf{NQP} = \mathsf{NTIME}[n^{\log^{O(1)} n}]$, circuit lower bounds for these smaller classes could be obtained as well, provided that the necessary (weak) CIRCUIT SAT algorithms are fast enough.[2] Prior work tried to avoid this problem by working with the larger classes $\mathsf{QP}^{\mathsf{NP}}$ [Wil11] or P^{NP} [FSW09], but these are less natural and less obviously weaker (it is open whether $\mathsf{NEXP} = \mathsf{P}^{\mathsf{NP}}$!).

1.1 Easy Witness Lemmas for NP and NQP and Applications.

In this paper, we prove an easy witness lemma that works for both NP and NQP. These results readily imply (using the existing framework) stronger circuit lower bounds: for example, NQP does not have $\mathsf{ACC} \circ \mathsf{THR}$ circuits of $n^{\log^k n}$ size, for all fixed k.

Lemma 1.2 (Easy Witnesses for NP) *There is a $c \geq 1$ such that for all $k \geq 1$, if $\mathsf{NP} \subset \mathsf{SIZE}[n^k]$ then every $L \in \mathsf{NP}$ has witness circuits of size at most n^{ck^3}.*

Lemma 1.3 (Easy Witnesses for NQP) *There is a $c \geq 1$ such that for all $k \geq 1$, if $\mathsf{NQP} \subset \mathsf{SIZE}[2^{\log^k n}]$ then every $L \in \mathsf{NQP}$ has witness circuits of size $2^{c \log^{k^3} n}$.*

The full formal statement encapsulating both lemmas can be found in Lemma 4.1. Our primary motivation for proving these lemmas is to improve the known generic connections between circuit-SAT algorithms and circuit lower bounds. We only need the underlying class $\mathscr{C}$ of Boolean circuits to satisfy a few simple properties. Informally, a circuit class $\mathscr{C}$ is *evaluatable* if given a description of a C from $\mathscr{C}$ and an assignment x to some of the inputs, it is easy to evaluate $C(x)$; $\mathscr{C}$ is *closed under negation* if it is easy to compute the negation of a given circuit from $\mathscr{C}$; $\mathscr{C}$ is *closed under conjunctions* if it is easy to take a collection of $\mathscr{C}$-circuits, each with n inputs, and output one n-input $\mathscr{C}$-circuit equivalent to the conjunction of the given circuits. Classes with all three properties are called *typical*. (Definitions are in the Preliminaries.)

For any circuit class $\mathscr{C}$, let $\mathscr{C}$-SAT be the satisfiability problem for circuits from $\mathscr{C}$. We state our results in terms of a presumably much easier problem, the GAP $\mathscr{C}$ UNSAT problem, in which we are promised that the given $\mathscr{C}$-circuit on n inputs is either *unsatisfiable*, or *has at least* $2^n/4$ *satisfying assignments*. Unlike $\mathscr{C}$-SAT, the GAP $\mathscr{C}$ UNSAT problem is trivially solvable with randomness. (In prior work [SW13] the problem is called DERANDOMIZE-$\mathscr{C}$; another version is called CAPP [IKW02].) We say a *nondeterministic algorithm solves* GAP $\mathscr{C}$ UNSAT if:

- it outputs *yes*, *no*, or *don't know* on every computation path,
- if its input circuit has at least $2^n/4$ satisfying assignments, then it outputs *no* on some path,
- if its input circuit is unsatisfiable, then it outputs *yes* on some path, and
- for every input circuit, it never outputs *yes* on some path and *no* on another path.

It is widely believed that GAP $\mathscr{C}$ UNSAT is solvable in *deterministic polynomial time*; this would follow from the existence of pseudorandom generators sufficient for $\mathsf{P} = \mathsf{BPP}$. With this notation, our new algorithms-to-lower bounds connections can be stated:

[1]There are simple easy witness lemmas for deterministic classes such as P and EXP, as one can see from [Lip94, FSW09]. But it is not known how to extend the *connection* from SAT algorithms to circuit lower bounds to prove non-uniform lower bounds for functions in deterministic classes such as P and EXP; a crucial aspect of those arguments is that the non-uniform circuit is *nondeterministically guessed* in the eventual proof-by-contradiction.

[2]It's worth noting that proving NP doesn't have circuits of size n^k would have other major implications, such as $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ and non-trivial derandomizations of MA [IKW02].

Theorem 1.1 (NP Lower Bounds from $2^{(1-\varepsilon)n}$-time Circuit-SAT) *Let $\mathscr{C}$ be typical, and let $\varepsilon \in (0,1)$. Suppose* GAP $\mathscr{C}$ UNSAT *on n-input, $2^{\varepsilon n}$-size circuits is solvable in nondeterministic $O(2^{(1-\varepsilon)n})$ time. Then there is a $c \geq 1$ such that for all k, $\mathsf{NTIME}[n^{ck^4/\varepsilon}]$ does not have n^k-size $\mathscr{C}$-circuits.*[3]

Theorem 1.2 (NQP Lower Bounds from $2^{n-n^{\varepsilon}}$-time Circuit-SAT) *Let $\mathscr{C}$ be typical, and let $\varepsilon \in (0,1)$. Suppose* GAP $\mathscr{C}$ UNSAT *on n-input, $2^{n^{\varepsilon}}$-size circuits is in nondeterministic $O(2^{n-n^{\varepsilon}})$ time. Then for all k, there is a $c \geq 1$ such that $\mathsf{NTIME}[2^{\log^{ck^4/\varepsilon} n}]$ does not have $2^{\log^k n}$-size $\mathscr{C}$-circuits.*

It is important to note that for $2^{\varepsilon n}$ size and n inputs, CIRCUIT SAT is solvable in polynomial time. Similarly, CIRCUIT SAT is in quasi-polynomial time for circuits of size $2^{n^{\varepsilon}}$. Thus Theorems 1.1 and 1.2 are concerned with improving the running time of problems in P and QP respectively, possibly using nondeterministic algorithms, and the problem GAP $\mathscr{C}$ SAT is believed to be in P even for poly(n) circuit sizes.

These new generic algorithm-to-lower-bound connections immediately imply much tighter lower bounds for ACC and $\mathsf{ACC} \circ \mathsf{THR}$ circuits, namely the separation $\mathsf{NQP} \not\subset \mathsf{ACC} \circ \mathsf{THR}$:

Theorem 1.3 *For every constant k, d, and $m \geq 2$, there is an $e \geq 1$ and a problem in $\mathsf{NTIME}[n^{\log^e n}]$ which does not have depth-d $n^{\log^k n}$-size $\mathsf{AC}[m]$ circuits, even with $n^{\log^k n}$ linear threshold gates at the bottom layer.*

1.2 Intuition

Our proof of the easy witness lemma for NP and NQP has some similarities to the original Impagliazzo-Kabanets-Wigderson argument [IKW02], but makes some important modifications. Let us review their proof. For simplicity, we restrict ourselves here to looking at poly-size circuits, but the full proof is much more general, allowing for quasi-polynomials and even third-exponentials.

IKW's Easy Witness Lemma. At a high level, the proof of Lemma 1.1 works as follows. Assume

(A) $\mathsf{NTIME}[2^n]$ has circuits of size n^k, and

(B) $\mathsf{NTIME}[2^n]$ *does not* have witness circuits of size $n^{O(k^2)}$.

We wish to obtain a contradiction. We show that (A) and (B) contradict the theorem that there is a language $L_{hard} \in \mathsf{TIME}[2^{n^{k+1}}]$ such that for all but finitely many n, L_{hard} does not have n-input circuits of size $O(n)^k$ (note this is provable by direct diagonalization; see Theorem 2.3 in the Preliminaries, which shows we can put L_{hard} in $\mathsf{SPACE}[n^{k+1}]$).

Assumption (A) implies that $L_{hard} \in \mathsf{TIME}[2^{n^{k+1}}]$ has $n^{O(k^2)}$-size circuits, by padding/translation. Building on the proof that $\mathsf{EXP} \subset \mathsf{P/poly}$ implies $\mathsf{EXP} = \mathsf{MA}$ ([BFNW93]), that in turn implies (as noted by [MNW99]) that L_{hard} has a Merlin-Arthur protocol P running in $n^{O(k^2)}$ time.

We can use (B) to "derandomize" the Arthur part of the Merlin-Arthur protocol P for L_{hard}, as follows. Assumption (B) implies that for all $c \geq 1$, there is a "bad" $O(2^n)$-time verifier V_c that takes x of length n and y of length $O(2^n)$, such that for infinitely many n_i, there is a "bad" input x_i of length n_i such that $V_c(x_i, y_i)$ accepts for some y_i, but *every* y_i accepted by $V_c(x_i, \cdot)$ has circuit complexity at least n^{ck^2}. Thus, if we give x_i as n bits of *advice*, nondeterministically *guess* a string y_i of length $O(2^n)$, and verify $V_c(x_i, y_i)$ accepts, we are guaranteed y_i encodes the truth table of a function with circuit complexity at least n^{ck^2}.

Applying standard hardness-to-randomness connections, the hard function y_i can be used to power a pseudorandom generator G with $O(k^2 c \log n)$-length seeds that runs in $2^{O(n)}$ time and fools circuits of size $n^{ck^2/g}$, for some universal $g \geq 1$. For large enough c, we have a generator G that can "fool" the Arthur part of

[3]In fact, the GAP $\mathscr{C}$ UNSAT algorithm could even use $2^{o(n)}$ *bits of advice*; see Remark 1.

the protocol P for L_{hard}, given an input and Merlin's message. So by guessing y_i (of $O(2^n)$ length), guessing Merlin's message, and running G on all seeds, the Merlin-Arthur protocol P can be faithfully simulated on infinitely many input lengths n_i by a nondeterministic algorithm N, running in $2^{O(n)}$ time, with n bits of advice. Finally, applying (A) once more to N, we conclude that L_{hard} has circuits of size $O(n)^k$ on infinitely many input lengths, contradicting the definition of L_{hard}.

How to Improve IKW? A Minor Modification. Suppose we replace $\mathsf{NTIME}[2^n]$ with NP or NQP in assumptions (A) and (B) in the above. What goes wrong? For starters, we no longer have $\mathsf{EXP} = \mathsf{MA}$ or $\mathsf{PSPACE} = \mathsf{MA}$ as a consequence of our assumptions, so we can no longer easily infer from our assumptions that a hard language L_{hard} has efficient Merlin-Arthur protocols! Thus it is not at all clear what lower bound we may hope to contradict. All is not lost though. Since [IKW02] appeared, other circuit lower bounds have been proved, such as the celebrated result of Santhanam [San07] that for all k, $\mathsf{MA}/1 \not\subset \mathsf{SIZE}[n^k]$. In particular, Santhanam ([San07], Theorem 4.3) gives an explicit language L_k computable by a Merlin-Arthur protocol P_k in $n^{O(k^2)}$ time (with 1 bit of advice) which does not have n^k-size circuits. Letting $d \gg k^2$, and assuming $\mathsf{NTIME}[n^d]$ does not have witness circuits of $n^{O(k^2)}$ size, we can derandomize P_k just as in the IKW argument, by guessing-and-verifying a witness of length $O(n^d)$ with circuit complexity at least $n^{\Omega(k^2)}$. We conclude that L_k is solvable in nondeterministic $n^{O(d)}$ time, with $n+1$ bits of advice, for infinitely many input lengths. If we further assume that $\mathsf{NTIME}[n^{O(d)}]$ has n^k-size circuits, we could then infer that L_k has $O(n)^k$-size circuits *for infinitely many input lengths*. Unfortunately, Santhanam's $\mathsf{MA}/1$ lower bound for L_k is not known to hold in the infinitely-often setting; his proof only rules out a n^k-size circuit family computing L_k on all but finitely many input lengths. However, this new argument does show that, if NP has n^k-size circuits, then NP also has $n^{O(k^2)}$-size witness circuits for *infinitely many input lengths n* (because if NP did not have such witness circuits almost everywhere, then we could simulate P_k nondeterministically *almost everywhere* with small advice, and get the desired contradiction). Call this the **"infinitely-often witness" argument**.

As far as we can tell, having witness circuits "infinitely often" is not enough for the lower bound applications. (For one, we would need an "almost-everywhere" nondeterministic time hierarchy theorem to get the final contradiction, but it is open whether NEXP is contained in *io*-NP!) We need another approach to getting a contradiction; perhaps another type of circuit lower bound for MA would work?

Our Approach: "Almost" an Almost-Everywhere Circuit Lower Bound for MA. Again, here we stick to polynomial-size circuits, to keep the exposition simple. By allowing a little more advice to a Merlin-Arthur protocol, we still do not get a lower bound for all but finitely many input lengths, but we can guarantee that $\mathsf{MA}/O(\log n)$ has n^k-size lower bounds on input lengths that are only polynomially far apart, for *all but finitely many n*. In particular, we prove:

Theorem 1.4 (Informal) *For all k, there is an explicit language L'_k computable by a Merlin-Arthur protocol P_k in $n^{O(k^2)}$ time (with $O(\log n)$ bits of advice) such that, for all but finitely many input lengths n, either*

- *L'_k does not have an n^k-size circuit on inputs of length n, or*
- *L'_k does not have an $n^{O(k^2)}$ size circuit on inputs of length $n^{O(k)}$.*

The full formal statement can be found in Theorem 3.1. Our L'_k is related to Santhanam's aforementioned $\mathsf{MA}/1$ language, as they both attempt to compute a paddable, self-correctable, and downward self-reducible PSPACE-complete language L_{PSPACE} on padded inputs (due to Trevisan and Vadhan [TV02], with modifications by Santhanam [San07]). Santhanam breaks his analysis into two cases, whether or not PSPACE has polynomial-size circuits, and shows his language does not have small circuits in either case.

Our advice and analysis are very different. At a high level, our Merlin-Arthur protocol P_k has two parts. Its advice on input length n encodes the largest ℓ_n such that L_{PSPACE} has circuits of size n^{ak^2} on length-ℓ_n inputs for a sufficiently large $a \geq 1$, which will indicate which part to run on an input y of length n:

Part 1: If $\ell_n \geq n^{ck}$, then P_k guesses-and-randomly-checks a circuit of n^{ak^2} size with n^{ck} inputs for L_{PSPACE} (via Theorem 2.2), and uses this circuit to decide if y is in a language $L_{diag} \in \mathsf{SPACE}[n^{O(k)}]$ (Theorem 2.3) which provably *does not have* n^k-size circuits for any infinite set of input lengths. (Here, we use the fact that L_{PSPACE} is PSPACE-complete.)

Part 2: If $\ell_n < n^{ck}$, then P_k parses y as a padded string 1^q0x. If $|x|+1 > \ell_n$ then P_k rejects, else P_k tries to compute L_{PSPACE} on $1^{\ell_n-|x|-1}0x$, by guessing-and-checking a circuit with ℓ_n inputs and n^{ak^2} size.

(Here, $c \leq a$ is a large enough constant.) Let L'_k be the language accepted by P_k. By our choice of ℓ_n, P'_k only guesses circuits for L_{PSPACE} when valid ones exist, and it uses the circuit as an oracle in Santhanam's checker for L_{PSPACE}. Together, these ensure that P'_k satisfies the MA promise condition on all inputs.

To prove the lower bound for L'_k, suppose there is an infinite set $S \subseteq \mathbb{N}$ such that for all $n \in S$,

(A) L'_k has n^k-size circuits for length-n inputs, and

(B) L'_k has n^{ck^2}-size circuits for length-n^{ck} inputs.

We wish to derive a contradiction. We consider two cases:

Case 1. Suppose there is an infinite subset $S' \subseteq S$ such that for all $n \in S'$, $\ell_{n^{ck}} \geq n^{c^2k^2}$. Then for all $n \in S'$, P_k runs the first part of its protocol on n^{ck}-length inputs. In particular, for all $n \in S'$, P_k on y of length n^{ck} guesses a circuit of $n^{O(k^3)}$ size with $n^{O(k^2)}$ inputs for L_{PSPACE}, verifies the circuit using randomness, then uses the circuit to simulate L_{diag}. By definition of L_{diag}, for almost all $n \in S'$, L'_k does **not** have n^{ck^2}-size circuits for n^{ck}-length inputs. However, note this is in direct contradiction with (B).

Case 2. Suppose for all but finitely many $n \in S$, $\ell_{n^{ck}} < n^{c^2k^2}$. Then for almost all $n \in S$, the protocol P_k runs the second part of its protocol on n^{ck}-length inputs: P_k simulates L_{PSPACE} on $1^{\ell_n-|x|-1}0x$, provided $|x|+1 \leq \ell_n$. We show how the second part of P_k, together with assumptions (A) and (B), can be used to infer that L_{PSPACE} itself has an n^{ck^2}-size circuit on inputs of length n^{ck}, using a "bootstrapping" kind of argument.

In more detail, assume for some m that L'_k has an m^k-size circuit on length-m input, and that $\ell_m < m^{ck}$, so that the second part of P_k is run on length-m inputs. We start by observing $\ell_m \geq 1$: P_k can (unconditionally) guess-and-check $O(1)$-size circuits for L_{PSPACE} on inputs of length 1. If P_k on m-bit inputs can be used to compute L_{PSPACE} on ℓ-bit inputs (i.e., $m^{ck} > \ell_m \geq \ell$), then (using the assumed L'_k circuit) there is an m^k-size circuit for L_{PSPACE} on ℓ-bit inputs. By the downward self-reducibility of L_{PSPACE}, there is also an m^{dk}-size circuit C computing L_{PSPACE} on $(\ell+1)$-bit inputs, for some (fixed) d. For $d \leq ak$, this circuit C implies that $\ell_m \geq \ell+1$. Thus the protocol P_k has enough time to guess and verify this C, and the second part of P_k on m-bit inputs can also compute L_{PSPACE} on $(\ell+1)$-bit inputs. Since L'_k has a m^k-size circuit on length-m inputs, the m^{dk}-size circuit for L_{PSPACE} on $(\ell+1)$-bit inputs can then be replaced by an m^k-size circuit(!). We can repeat this argument for lengths $\ell = 1, \ldots, m-1$, implying that L_{PSPACE} has m^k-size circuits for input length m.

Therefore, in Case 2, for almost all $n \in S$, n^{ck^2}-size circuits for L'_k on length-n^{ck} inputs (assumption (B)) implies that L_{PSPACE} has an n^{ck^2}-size circuit for length n^{ck}. Since $c \leq a$, by definition of ℓ_n it follows that for almost all $n \in S$ we have $\ell_n \geq n^{ck}$. Thus the first part of P_k is run on these $n \in S$, so P_k is simulating L_{diag} on almost all inputs of length $n \in S$. By definition of L_{diag}, for almost all $n \in S$, L'_k does not have n^k-size circuits for length-n inputs. But this is a direct contradiction with (A). This completes the intuition for Theorem 1.4.

From an MA Circuit Lower Bound to Easy Witnesses. Let us briefly discuss how this Merlin-Arthur lower bound can be used for our new Easy Witness Lemma. We have to show the "almost" almost-everywhere lower bound is enough for obtaining a contradiction with the aforementioned "infinitely-often witness" argument. Assume NP has n^k-size circuits, and assume there is an NP-verifier V that does not have n^{ck^3}-size witness circuits (rather than n^{k^2}, as in IKW) for large c. Then there are infinitely many "bad" inputs x_i such that $V_c(x_i, y_i)$ accepts for some y_i, but every y_i accepted by $V_c(x_i, \cdot)$ has circuit complexity at least n^{ck^3}. Then, just as in the IKW argument and infinitely-often witness argument, we can use y_i in a pseudorandom generator that fools circuits of size $n^{ck^3/g}$. Take the "hard" Merlin-Arthur protocol P_k. On infinitely many pairs of input lengths $(n, n^{\Theta(k)})$ corresponding to the bad inputs x_i, we want to derive an n^k-size circuit for length-n inputs *and* an $n^{\Theta(k^2)}$-size circuit on length-$n^{\Theta(k)}$ inputs, to contradict the "almost" almost-everywhere lower bound. Because the circuit constructed from the Merlin-Arthur protocol has size $n^{O(k^3)}$ even on inputs of length $n^{\Theta(k)}$, this pseudorandom generator works not only for input length n but *also* for length n^k. Thus, if witness circuits of size n^{ck^3} did not exist, we would be able to simulate P_k on infinitely many pairs of input lengths $(n, n^{\Theta(k)})$ in NP, using $n + O(\log n)$ bits of advice. Assuming NP has n^k-size circuits, this contradicts the earlier lower bound proved for P_k.

Outline of the Rest. The rest of the paper is structured as follows. In Section 2 we give definitions and review related material from the literature. In Section 3 we prove the "almost" almost-everywhere lower bound for MA. In Section 4 we prove the new Easy Witness Lemma in its full form. In Section 5 we improve connections between circuit-analysis and circuit lower bounds using the Lemma. In Section 6 we conclude with some open questions.

2 Preliminaries

Basics. All languages/functions are over $\{0,1\}$, and all logarithms are base-2 with ceilings as appropriate. We assume knowledge of basic complexity theory [AB09] such as circuit families computing a language, and complexity classes such as EXP, NEXP, ACC, etc. We use $\mathsf{SIZE}[s(n)]$ to denote the class of problems computed by a (non-uniform) $s(n)$-size circuit family. When we refer to a "typical" circuit class (AC^0, ACC, TC^0, NC^1, NC, or P/poly), we assume the underlying circuit families are *non-uniform* and of *polynomial-size*, unless otherwise specified.

We will often take as given that any $O(t(n))$-time algorithm can be converted (in $O(t(n)^2)$ time) into an equivalent circuit family of size at most $t(n)^2$, for all but finitely many n. (Different circuit conversions would only affect the constant factors in our arguments.)

We use *advice classes*: for a deterministic or nondeterministic class $\mathscr{C}$ and a function $a(n)$, $\mathscr{C}/a(n)$ is the class of languages L such that there is an $L' \in \mathscr{C}$ and an arbitrary function $f : \mathbb{N} \to \{0,1\}^\star$ with $|f(n)| \le a(n)$ for all x, such that $L = \{x \mid (x, f(|x|)) \in L'\}$. That is, the arbitrary advice string $f(n)$ can be used to solve all n-bit instances within class $\mathscr{C}$.

In our MA lower bound, we need to show that some protocol "satisfies the MA promise" on all inputs:

Definition 2.1 *We say that a Merlin-Arthur protocol P* satisfies the MA promise on length ℓ *if for all $x \in \{0,1\}^\ell$, either there is a Merlin message such that Arthur accepts x with probability at least $2/3$, or for all Merlin messages, Arthur rejects x with probability at least $2/3$. In other words, P satisfies the MA promise on length ℓ if it obeys the usual semantics of an MA language on all inputs of that length.*

Typical Circuit Classes. In general, a Boolean circuit class $\mathscr{C}$ is a collection of *descriptions* of Boolean functions. $\mathscr{C}$ is *evaluatable* if given a description of a C from $\mathscr{C}$ and given an 0/1 assignment a to some (but not necessarily all) of the inputs, the sub-circuit $C(a)$ can be determined in polynomial time. Say that a class of Boolean circuits $\mathscr{C}$ is *closed under negation* if there is a poly(n)-time algorithm A such that, given any n-bit description of a circuit C from $\mathscr{C}$, A outputs a description of $\neg C$. $\mathscr{C}$ is *closed under conjunctions* if there is a poly(n)-time algorithm A such that, given the descriptions of n circuits $C_1, C_2, \ldots, C_n$, each on n inputs from $\mathscr{C}$, algorithm A outputs a circuit $C'(x_1, \ldots, x_n) \equiv C_1(x_1, \ldots, x_n) \wedge \cdots \wedge C_n(x_1, \ldots, x_n)$. Any $\mathscr{C}$ satisfying all three properties is called *typical*.

Circuit Complexity of Strings and Pseudorandom Generators. Let $x_1, \ldots, x_{2^\ell}$ be the ℓ-bit strings in lexicographical order. For a circuit C on ℓ inputs, define $tt(C) := C(x_1) \cdots C(x_{2^\ell})$, i.e., $tt(C) \in \{0,1\}^{2^\ell}$ is the truth table of C. For every string y, let 2^ℓ be the smallest power of 2 such that $2^\ell \geq |y|+1$. We define the *circuit complexity of* y, or $CC(y)$, to be the circuit complexity of the ℓ-input function defined by the truth table $y10^{2^\ell-|y|-1}$. We will use the following strong construction of pseudorandom generators from hard functions:

Theorem 2.1 (Umans [Uma03]) *There is a universal constant g and a function $G : \{0,1\}^\star \times \{0,1\}^\star \to \{0,1\}^\star$ such that, for all s and Y satisfying $CC(Y) \geq s^g$, and for all circuits C of size s,*

$$\left| \Pr_{x \in \{0,1\}^{g \log |Y|}} [C(G(Y,x)) = 1] - \Pr_{x \in \{0,1\}^s} [C(x) = 1] \right| < 1/s.$$

Furthermore, G is computable in poly$(|Y|)$ time.

Witness Circuits. We formally define the terminology behind "witness circuits" (also found in [Wil10, Wil16, Wil11]).

Definition 2.2 *Let $L \in \mathsf{NTIME}[t(n)]$ where $t(n) \geq n$ is constructible. An algorithm $V(x,y)$ is a* **verifier for** *L if V runs in time $O(|y| + t(|x|))$ and for all strings x,*

$$x \in L \iff \textit{there is a } y \textit{ of length } O(t(n)) \textit{ (a witness for } x\textit{) such that } V(x,y) \textit{ accepts.}$$

We denote $L(V)$ to be the $\mathsf{NTIME}[t(n)]$ language verified by V.
We say V **has witness circuits of size** *$w(n)$ if for all strings x, if $x \in L(V)$ then there is a y_x of length $O(t(n))$ such that $V(x, y_x)$ accepts and y_x has circuit complexity at most $w(n)$.*
We say L **has witness circuits of size** *$w(n)$ if every verifier for L has witnesses of size $w(n)$.*
$\mathsf{NTIME}[t(n)]$ has witness circuits of size $w(n)$ *if every $L \in \mathsf{NTIME}[t(n)]$ has witness circuits of size $w(n)$.*

Thinking of $w(n) \ll t(n)$, if a nondeterministic time-$t(n)$ language L has size-$w(n)$ witness circuits, then for *every* $O(t(n))$-time verifier V for L, all $x \in L$ have a highly *compressible* witness y_x with respect to V: y_x is a string of length $t(n)$, but as a truth table, it has a circuit of size at most $w(n)$.

A Structured PSPACE-Complete Problem. A fundamental result used often in the theory of pseudorandomness (modifying a construction of Trevisan and Vadhan [TV02, San07]) is that there is a PSPACE-complete language with strong reducibility properties. First, we define the properties.

Definition 2.3 *A language $L \subseteq \{0,1\}^\star$ is* downward self-reducible *if there is a (deterministic) polynomial-time oracle algorithm $A^?$ such that for all n and all $x \in \{0,1\}^n$, $A^{L^{n-1}}(x) = L(x)$.*

Theorem 2.2 ([San07]) *There is a PSPACE-complete language L_{PSPACE} which is paddable, downward self-reducible, and furthermore is "checkable" in that there is a probabilistic polynomial-time oracle Turing machine M so that, for any input x,*

- *M asks its oracle queries only of length $|x|$.*

- *if M is given L_{PSPACE} as oracle and $x \in L$, then M accepts with probability 1.*
- *if $x \notin L_{\mathsf{PSPACE}}$, then irrespective of the oracle given to M, M rejects with probability at least 2/3.*

Note that Santhanam proves the existence of the probabilistic polynomial-time "checker" M, and that L_{PSPACE} is PSPACE-complete; we also note in the below proof that his language is paddable and downward self-reducible. All four of these properties of L_{PSPACE} will be used in our argument.

Proof. We simply verify that the language given by Santhanam [San07] has the additional desired properties. Let $L_{\mathsf{PSPACE}} = \{1^i \mid i \geq 0\} \cup \{1^i 0x \mid x \in S \text{ and } i \geq 0\}$, where S is the PSPACE-complete language of Trevisan and Vadhan [TV02] which is downward self-reducible and "self-correctable" (which we do not define here). By definition, it is clear that for any amount of padding m, $1^m x \in L_{\mathsf{PSPACE}} \iff x \in L_{\mathsf{PSPACE}}$. To construct a downward self-reduction A for L_{PSPACE}, we can make use of the downward self-reduction A_0 for the language S:

```
  $A$: On input $y$, parse $y = 1^m 0x$. If $m > 0$, *accept* iff $1^{m-1}0x \in L_{\mathsf{PSPACE}}$.
If $m = 0$, run the downward self-reduction $A_0$ on $x$.
For every query $x'$ for $S$ made by $A_0$, query $0x' \in L_{\mathsf{PSPACE}}$ instead.
*Accept* if and only if $A_0$ accepts on $x$.
```

Observe A is a downward self-reduction for L_{PSPACE}: Either y has some amount of 1-padding at the beginning, or it has no 1-padding. In the first case, L_{PSPACE} can be quickly solved with one oracle query (remove a 1). In the second case, the downward self-reduction for S can be used. Since x has length $|y|-1$, all of the queries x' made by A_0 are of length $|y|-2$, so $0x'$ is a query to L_{PSPACE} of length $|y|-1$. □

We also need a standard almost-everywhere circuit lower bound, provable by direct diagonalization:

Theorem 2.3 *Let $s(n) < 2^n/(2n)$ be space-constructible. There is a language $L_{diag} \in \mathsf{SPACE}[s(n)^2]$ that does not have $s(n)$-size circuits for all but finitely many input lengths n.*

Proof. Folklore. On an input of length n, using $O(s(n)^2)$ space one can exhaustively try all Boolean functions $f : \{0,1\}^{2\log(s(n))} \to \{0,1\}$ and circuits of size $s(n)$, until one finds the lexicographically first function f_n, represented as a bit string of length $2^{2\log(s(n))} \leq O(s(n)^2)$, that is not computable by any circuit of size $s(n)$. Then the padded language

$$L_{diag} = \bigcup_{n\in\mathbb{N}} \{1^{n-1-2\log(s(n))}0x \mid |x| = 2\log(s(n)) \wedge f_n(x) = 1\}$$

does not have $s(n)$-size circuits for all but finitely many n, but is computable in space $O(s(n)^2)$. □

Significant Separations. For the case of poly-size circuits, the circuit lower bound of Theorem 3.1 implies a "significant separation", as defined by Fortnow and Santhanam [FS17]. They say a class $\mathscr{C}$ has a significant separation from $\mathscr{D}$, if there is an $f \in \mathscr{C}$ such that for all $g \in \mathscr{D}$ there is a k so that for every m, the function f_n (restricted to n-bit inputs) differs from g_n on some input length n in $[m, m^k]$. The lower bound of Theorem 3.1 implies a significant separation of MA (with advice) from small circuits. However, our lower bound seems a bit stronger in that, rather than proving an lower bound for some input length in all polynomially-long intervals, we show an lower bound for every pair of input lengths which are polynomially far apart.

3 "Almost" an Almost-Everywhere Circuit Lower Bound for MA

In this section, we prove the main lower bound needed for our new Easy Witness Lemmas. For a language $L \subseteq \{0,1\}^\star$ and $n \in \mathbb{N}$, let L^n be the language L restricted to n-bit inputs. Say that $s : \mathbb{N} \to \mathbb{N}$ is a *circuit-size function* if it is increasing, time constructible, and $s(n) < 2^n/(2n)$ holds for all sufficiently large n.

Theorem 3.1 ("Almost" A.E. Circuit Lower Bound for MA with Advice) *There are $d_1, d_2, d_3 \geq 1$ such that for all circuit-size functions s and time-constructible $s_1, s_2 : \mathbb{N} \to \mathbb{N}$ satisfying*

(i) $s_2(n) \geq s(n)^{2d_2}$,
(ii) $s_1(n) \geq s(s_2(n))$, *and*
(iii) $s_1(n) \geq s(n)^{2d_1+1}$,

there is a language L_1 computable by a Merlin-Arthur protocol in $O(s_1(n)^2 \cdot s_2(n)^{d_3})$ time with $2\log s_2(n)$ advice, such that for all sufficiently large $n \in \mathbb{N}$, either

- *the circuit complexity of L_1^n is greater than $s(n)$, or*
- *the circuit complexity of $L_1^{s_2(n)}$ is greater than $s(s_2(n))$.*

That is, we "almost" prove an almost-everywhere size-$s(n)$ circuit lower bound for MA with $2\log s(n)$ advice: for almost all n, the MA language L_1 either has high complexity on length n, or it has high complexity on length $s_2(n)$. Later, we will use this "almost" almost-everywhere lower bound to obtain a stronger easy witness lemma.

The rest of this section is devoted to proving Theorem 3.1. The language L_1 is formed from two different Merlin-Arthur protocols:

1. One protocol attempts to solve the problem L_{diag} that does not have size-$s(n)$ circuits almost everywhere (from Theorem 2.3), by guessing and checking circuits for the complete language L_{PSPACE}, and applying a reduction from L_{diag} to L_{PSPACE}.
2. The other protocol solves L_{PSPACE} (from Theorem 2.2) on inputs with sufficiently long padding.

The $2\log s_2(n)$ bits of advice are used to determine which of these two languages can be successfully computed by our MA protocol, on a given input length.

Set $d_1, d_2, d_3 \geq 1$ such that:

- the downward self-reduction A for L_{PSPACE} (as seen in Theorem 2.2) runs in time $O(n^{d_1})$,
- every reduction from a language $L \in \mathsf{SPACE}[s(n)]$ to L_{PSPACE} runs in time at most $O(s(n)^{d_2})$ and produces an L_{PSPACE} instance of length $s(n)^{d_2}$. To ensure this condition, set d_2 such that the language $L' = \{(M, x, 1^t) \mid M \text{ accepts } x \text{ using space at most } t\}$ can be reduced to L_{PSPACE} in $O(n^{d_2-1})$ time. Then, every $\mathsf{SPACE}[s(n)]$ language L has an $O(s(n))$-time reduction to L', and therefore L has an $O(s(n)^{d_2})$-time reduction to L_{PSPACE}.
- the probabilistic polytime oracle machine M for L_{PSPACE} (as seen in Theorem 2.2) runs in time $O(n^{d_3})$.

Since $s(n) < 2^n/(2n)$, there is a language L_{diag} without $s(n)$-size circuits almost everywhere (Theorem 2.3). Using $L_{diag} \in \mathsf{SPACE}[s(n)^2]$ and the reduction from L_{diag} to L_{PSPACE}, we have:

Claim 1 *There is an $O(s(n)^{2d_2})$-time reduction R such that for all strings y, $|R(y)| = s(|y|)^{2d_2}$ and $y \in L_{diag} \iff R(y) \in L_{\mathsf{PSPACE}}$.*

We are now in position to define a Merlin-Arthur protocol M_1 recognizing a hard language L_1. For input length n, define

$$\ell(n) := \text{the largest integer } \ell \text{ such that } L_{\mathsf{PSPACE}}^{\ell} \text{ has a circuit of size at most } s_1(n). \tag{1}$$

Note that a circuit always exists when $s_1(n) \geq 2^\ell$, so we have $\ell(n) \geq \log(s_1(n))$. Also note that since L_{PSPACE} is paddable, a circuit for L^m_{PSPACE} can solve L^ℓ_{PSPACE} for any $\ell < m$ by setting the first $m - \ell$ bits to 1. Our advice α_n for inputs of length n will be the integer

$$\alpha_n := \min\{s_2(n), \ell(n)\}. \tag{2}$$

Note that α_n can be encoded in at most $2 \log s_2(n)$ bits. (For example, the first bit of the encoding could denote which of $s_2(n), \ell(n)$ is the minimum; if it is $s_2(n)$, then no further information is required, as that can be computed by the machine. If the minimum is $\ell(n)$, then we can encode $\ell(n)$ as an integer in $\{0, 1, \dots, s_2(n) - 1\}$ using less than $2 \log s_2(n)$ bits.)

Let R be the reduction from L_{diag} to L_{PSPACE} of Claim 1, with running time $O(s(n)^{2d_2})$. The Merlin-Arthur protocol M_1 defining a language L_1 is described in pseudocode as follows:

```
Advice $\alpha_n := \min\{s_2(n)$, largest $\ell \in \mathbb{N}$ s.t. $L^\ell_{\mathsf{PSPACE}}$ has an $s_1(n)$-size circuit$\}$
 $M_1(y)_{/\alpha_n}$ := Let $n = |y|$.
        If $\alpha_n = s_2(n)$:
            // simulate $L_{diag}$ on $y$
            Let $z = R(y)$; note $|z| \leq s_2(n)$ (by constraint (i)).
            Guess an $s_2(n)$-input circuit $C$ of size $s_1(n)$.
            Output $M^C(1^{s_2(n)-|z|}z)$ (Theorem 2.2).
        Otherwise ($\alpha_n < s_2(n)$):
            // simulate $L_{PSPACE}$ on a padded input
            Parse $y = 1^a0x$ for some $a \geq 0$. If $|x| + 1 > \alpha_n$ then reject.
            Guess an $\alpha_n$-input circuit $C$ of size $s_1(n)$.
            Output $M^C(1^{\alpha_n-|x|-1}0x)$ (Theorem 2.2).
```

MA Promise and Running Time Analysis. First, we verify that the advice α_n ensures that M_1 satisfies the MA promise on all input lengths — in particular, α_n ensures that the desired circuits C being guessed always exist in both branches of M_1 — and that M_1 runs in $O(s_1(n)^2 \cdot s_2(n)^{d_3})$ time. Intuitively, Merlin provides a circuit C and Arthur runs M^C.

- If $\alpha_n = s_2(n)$, then by definition (2) (and padding) $L^{s_2(n)}_{\mathsf{PSPACE}}$ has a $s_1(n)$-size circuit.
  If $y \in L_{diag}$, then $1^{s_2(n)-|z|}z \in L_{\mathsf{PSPACE}}$, which means there is a (Merlin) circuit C of size $s_1(n)$ such that $M^C(z)$ (Arthur) accepts with probability 1 (following Theorem 2.2). In particular, the reduction from L_{diag} to L_{PSPACE} runs in time $O(s(n)^{2d_2})$ and produces an instance z of length $s(n)^{2d_2}$. By constraint (i) ($s_2(n) \geq s(n)^{2d_2}$), we can pad z and M^C will accept it with probability 1.
  If $y \notin L_{diag}$, then for every possible (Merlin) circuit C, the probabilistic polynomial time M^C (Arthur) will reject with probability at least $2/3$ (again following Theorem 2.2). So M_1 satisfies the MA promise in this case. Simulating M^C can be done in time $O(s_1(n)^2 \cdot s_2(n)^{d_3})$, where the extra $s_1(n)^2$ factor comes from evaluating the circuit C on each oracle query.

- If $\alpha_n < s_2(n)$, then M_1 guesses a $s_1(n)$-size circuit C for $L_{\mathsf{PSPACE}}^{\alpha_n}$, then runs $M^C(1^{\alpha_n-|x|-1}1x)$.
  Suppose $y = 1^a0x \in L_{\mathsf{PSPACE}}$. By the paddability of L_{PSPACE}, it follows that $1^{\alpha_n-|x|-1}0x \in L_{\mathsf{PSPACE}}$ as well. Moreover by definition (2), $L_{\mathsf{PSPACE}}^{\alpha_n}$ has a circuit of size $s_1(n)$ when $\alpha_n < s(n)$. Therefore there is a circuit C of size-$s_1(n)$ and α_n inputs such that $M^C(1^{\alpha_n-|x|-1}0x)$ accepts with probability 1 (following Theorem 2.2).
  On the other hand, if $y = 1^a0x \notin L_{\mathsf{PSPACE}}$, then we also have $1^{\alpha_n-|x|-1}0x \notin L_{\mathsf{PSPACE}}$. By the properties of M from Theorem 2.2, it follows that for every possible circuit C, $M^C(1^{\alpha_n-|x|-1}0x)$ rejects with probability at least $2/3$, so M_1 satisfies the MA promise in this case as well.
  Here, the running time of M_1 is the time needed to run $M^C(1^{\alpha_n-|x|-1}0x)$ by directly simulating C, which is $O(s_1(n)^2 \cdot \alpha_n^{d_3}) \leq O(s_1(n)^2 \cdot s_2(n)^{d_3})$.

Therefore M_1 with advice $\{\alpha_n\}$ satisfies the MA promise on all inputs and runs in $O(s_1(n)^2 \cdot s_2(n)^{d_3})$ time.

"Almost" Almost-Everywhere Hardness. Let L_1 be the language recognized by M_1. Next we show for all sufficiently large $n \in \mathbb{N}$, either the circuit complexity of L_1^n is greater than $s(n)$, or the circuit complexity of $L_1^{s_2(n)}$ is greater than $s(s_2(n))$. We do this by showing that if the circuit complexity was low in both cases, even infinitely often, then the language L_{diag} would also have size-$s(n)$ circuits infinitely often, contradicting Theorem 2.3.

For the sake of contradiction, assume there is an infinite sequence $\{n_i\}$ such that

(A) $L_1^{n_i}$ has a circuit of size $s(n_i)$ and

(B) $L_1^{s_2(n_i)}$ has a circuit of size $s(s_2(n_i))$.

Let $I = \{n_i \mid i \in \mathbb{N}\} \cup \{s_2(n_i) \mid i \in \mathbb{N}\}$. We begin by showing that for all but finitely many $m \in I$, if L_1 has $s(m)$-size circuits on length m inputs, then L_1 cannot be simulating L_{diag} on length m (therefore it must be simulating the padded version of L_{PSPACE}, instead). We use the abbreviation "a.e." to mean "almost every", a.k.a. "all but finitely many" in the following.

Proposition 1 *For a.e. $m \in I$, if L_1^m has a $s(m)$-size circuit, then $\ell(m) < s_2(m)$.*

Proof. Suppose there were infinitely many $m \in I$ such that L_1^m has a circuit of size $s(m)$ and $\ell(m) \geq s_2(m)$. Then $\alpha_m = s_2(n)$ for those m, and by the definition of L_1, $L_1^m = L_{diag}^m$ for those m. Thus L_{diag} would have circuits of size $s(m)$ for infinitely many m, which is a contradiction to Theorem 2.3. □

Applying assumption (A) to Proposition 1, we find that for all sufficiently large $m \in I$,

$$\alpha_m < s_2(m). \tag{3}$$

Thus for all sufficiently large $m \in I$, a circuit for L_1^m computes a padded version of L_{PSPACE}, under our assumptions. Our next step is to show that sufficiently small circuits for L_1 for a given input length m actually implies analogously small circuits for L_{PSPACE} on input length m:

Lemma 3.1 *For a.e. $m \in I$, if L_1^m has a size-$s(m)$ circuit, then L_{PSPACE}^m has a size-$s(m)$ circuit.*

Assuming Lemma 3.1 holds, we can finish the proof of Theorem 3.1, and show the desired lower bound on L_1. In particular, applying assumption (B) to Lemma 3.1 yields that $L_{\mathsf{PSPACE}}^{s_2(n_i)}$ has a circuit of size $s(s_2(n_i))$, for all sufficiently large i. By constraint (ii) ($s_1(n) \geq s(s_2(n))$), $L_{\mathsf{PSPACE}}^{s_2(n_i)}$ has a circuit of size at most $s_1(n_i)$ for all sufficiently large i, so by definition of the ℓ-function, $\ell(n_i) \geq s_2(n_i)$. But this implies $\alpha_{n_i} = s_2(n_i)$ for all sufficiently large i, which contradicts (3).

We now turn to showing how to prove Lemma 3.1. Intuitively, our strategy is to use $s(m)$-size circuits for L_1^m to obtain $s(m)$-size circuits for L_{PSPACE} on progressively larger and larger input lengths, using the downward self-reducibility of L_{PSPACE} and the hypothesis that L_1^m has small circuits.

Proof of Lemma 3.1. Let $m \in I$ be sufficiently large, and assume L_1^m has a circuit C_m of size $s(m)$. We want to prove that L_{PSPACE}^m has a circuit of size $s(m)$. Since L_{PSPACE}^1 has an $O(1)$-size size circuit (and $s(m)$ is increasing), it's clear that L_{PSPACE}^1 has a circuit of size $s(m)$ for all but finitely many input lengths m. We make the following inductive claim:

Amplification Claim: Under the hypothesis, if there is an $\ell < m$ such that $L_{\mathsf{PSPACE}}^{\ell}$ has a size-$s(m)$ circuit, then $L_{\mathsf{PSPACE}}^{\ell+1}$ also has a size-$s(m)$ circuit.

Assuming the claim with $\ell := 1$, we conclude that $L_{\mathsf{PSPACE}}^{\ell+1} = L_{\mathsf{PSPACE}}^2$ also has a circuit of size $s(m)$. Repeatedly applying the claim for $\ell = 1$ up to $\ell = m-1$, we conclude that L_{PSPACE}^m has a circuit of size $s(m)$. That would complete the proof of the lemma.

To prove the Amplification Claim, suppose there is an $\ell < m$ such that $L_{\mathsf{PSPACE}}^{\ell}$ has a size $s(m)$ circuit D_ℓ. Using the downward self-reduction for L_{PSPACE}, we can also obtain a circuit $D'_{\ell+1}$ for $L_{\mathsf{PSPACE}}^{\ell+1}$, of size at most $O(s(m)^{2d_1+1})$. In particular, we can construct an $L_{\mathsf{PSPACE}}^{\ell}$-oracle circuit for $L_{\mathsf{PSPACE}}^{\ell+1}$ using the $O(n^{d_1})$-time downward self-reduction. This oracle circuit has size at most $(\ell+1)^{2d_1}$; if we replace each $L_{\mathsf{PSPACE}}^{\ell}$ oracle query with the size-$s(m)$ circuit D_ℓ, this yields a circuit $D'_{\ell+1}$ of size at most $(\ell+1)^{2d_1} \cdot s(m) \leq s(m)^{2d_1+1}$. By constraint (iii), we have that $D'_{\ell+1}$ has size at most $s_1(m)$.

We want to use the assumed size-$s(m)$ circuit C_m for L_1^m to reduce the circuit complexity of $L_{\mathsf{PSPACE}}^{\ell+1}$ down to $s(m)$. By (3), we have $\alpha_m < s_2(m)$ for sufficiently large $m \in I$, so (by the properties of the protocol M_1) the circuit C_m implements L_{PSPACE} for inputs of up to length α_m.

We claim that by feeding the string $1^{m-\ell-1}$ to the first $m-\ell-1$ inputs of C_m, the remaining subcircuit of size at most $s(m)$ can compute $L_{\mathsf{PSPACE}}^{\ell+1}$ on all x of length $\ell+1$. (Note that since we assume $\ell \leq m-1$, the padding amount $m-\ell-1 \geq 0$.) Since the circuit $D'_{\ell+1}$ has size at most $s_1(m)$, $L_{\mathsf{PSPACE}}^{\ell+1}$ has a circuit of size at most $s_1(m)$. Therefore by the definition of α_n (2) and $\ell(n)$ (1), for our choice of m we have $\alpha_m \geq \ell+1$.

Since C_m implements L_{PSPACE} for inputs of up to length α_m, and $\alpha_m \geq \ell+1$, the circuit

$$C'_{\ell+1}(x) := C_m(1^{m-\ell-1}x)$$

is of size $s(m)$ and computes $L_{\mathsf{PSPACE}}^{\ell+1}$ correctly (by the paddability of L_{PSPACE}). This completes the proof of the Amplification Claim, and also the proof of Lemma 3.1. □

This completes the proof of Theorem 3.1.

4 Small Circuits Imply Small Witnesses

In this section we prove our new Easy Witness Lemma, using the new Merlin-Arthur circuit lower bound (Theorem 3.1).

Lemma 4.1 (Easy Witnesses for Low Nondeterministic Time) *There are $e, g \geq 1$ such that for every increasing time-constructible $s(n)$, $s_2(n) := s(e \cdot n)^e$, and $t(n)$, if $\mathsf{NTIME}[O(t(n)^e)] \subset \mathsf{SIZE}[s(n)]$ then every $L \in \mathsf{NTIME}[t(n)]$ has witness circuits of size $s_2(s_2(s_2(n)))^{2g}$, provided that*

(a) $s(n) < 2^{n/e}/n$ and

(b) $t(n) \geq s_2(s_2(s_2(n)))^d$ for a sufficiently large constant d.

For $s(n) = n^k$, if $\mathsf{NP} \subset \mathsf{SIZE}[n^k]$, then NP has witness circuits of size $n^{O(k^3)}$ (Theorem 1.2). For $s(n) = 2^{(\log n))^k}$, $\mathsf{NQP} \subset \mathsf{SIZE}[2^{(\log n)^k}]$ implies that NQP has witness circuits of size $2^{O((\log n)^{k^3})}$ (Theorem 1.3).

Proof. Let $w(n) := s_2(s_2(s_2(n)))^{2g}$. Suppose

$$\mathsf{NTIME}[O(t(n)^e)] \subseteq \mathsf{SIZE}[s(n)], \tag{4}$$

and assume $\mathsf{NTIME}[t(n)]$ does not have $w(n)$-size witness circuits, i.e.,

$$\begin{aligned} &\text{there is some linear-time verifier } V \text{ such that there are infinitely many “bad” inputs } x, \\ &\text{where there is a } y_x \text{ of length } t(|x|) \text{ such that } V(x, y_x) \text{ accepts,} \\ &\text{yet all such } y_x \text{ have circuit complexity at least } w(|x|). \end{aligned} \tag{5}$$

We wish to establish a contradiction. Take $e \geq 1$ to be a sufficiently large parameter (all our constraints on e will be constant lower bounds on it). Using our assumptions on $s(n)$ and $s_2(n)$, we claim that the hard MA language L_1 of Theorem 3.1 exists for

$$s'(n) := s(e \cdot n), s_2'(n) := s'(n)^{d'}, \text{ and } s_1'(n) := s'(s_2'(n)),$$

assuming d' is chosen such that $max(2d_2, 2d_1 + 1, d_3 + 2) \leq d' \leq e$.

Let us check that these functions s', s_1', and s_2' satisfy all the constraints of Theorem 3.1:

- First, since $s(n) < 2^{n/e}/n$ (by constraint (a)), we know that $s'(n) = s(e \cdot n) < 2^{e \cdot n/e}/(e \cdot n) < 2^n/(2n)$, for all sufficiently large n and $e \geq 2$. Thus s' is a circuit-size function.
- For constraint (i), $d' \geq 2d_2$ implies $s_2'(n) = s'(n)^{d'} \geq s'(n)^{2d_2}$.
- For constraint (ii), $d' \geq 2d_1 + 1$ implies $s_1'(n) = s'(s_2'(n)) \geq s_2'(n) = s'(n)^{d'} \geq s'(n)^{2d_1+1}$.
- Constraint (iii) $s_1'(n) \geq s'(s_2'(n))$ is trivially satisfied.

Thus by Theorem 3.1, L_1 is computable by an MA protocol in $O(s_1'(n)^2 \cdot s_2'(n)^{d_3})$ time with advice of length

$$2 \log s_2'(n) = 2d' \log s'(n) < 2d' \cdot \log(2^{e \cdot n/e}) = 2d' \cdot n.$$

For $d' \geq d_3 + 2$, we also have

$$s_1'(n)^2 \cdot s_2'(n)^{d_3} = s'(s_2'(n))^2 \cdot s_2'(n)^{d_3} \leq s'(s_2'(n))^{2+d_3} \leq s_2'(s_2'(n)) \leq s_2(s_2(n))$$

since $s_2'(n) = s(e \cdot n)^{d'} \leq s(e \cdot n)^e = s_2(n)$ for $d' \leq e$. So L_1 is computable by an MA protocol in $O(s_2(s_2(n)))$ time with $2d' \cdot n$ bits of advice, and has circuit lower bounds for size $s'(n) = s(e \cdot n)$. That is, for all sufficiently large $n \in \mathbb{N}$, either

- the circuit complexity of L_1^n is greater than $s(e \cdot n)$, or
- the circuit complexity of $L_1^{s_2'(n)}$ is greater than $s(e \cdot s_2'(n))$.

By (5), there are infinitely many input lengths $\{n_i\}$ such that, by encoding a bad input x_i of length n_i as advice and guessing a valid y_i such that $V(x_i, y_i)$ accepts, we can nondeterministically guess and verify the truth table of a function y_i with circuit complexity at least $w(|x_i|)$.

Our next step is to use these bad x_i's to derandomize the MA protocol defining L_1: we give a nondeterministic algorithm N such that for all i and all inputs of length n_i, N uses $(2d' + 1)n_i$ bits of advice, runs in $O(t(n_i)^e)$ time, and correctly decides L_1 on all n_i-bit inputs. On inputs of length n_i, the advice to N is:

- a bad input x_{hard} of length n_i, and
- the advice α_{n_i} used by the Merlin-Arthur protocol M_1 of Theorem 3.1, on inputs of length n_i.

On inputs of length $s_2'(n_i)$, the advice to N is x_{hard} as well as the advice $\alpha_{s_2'(n_i)}$ used by M_1.

On inputs x of length n_i **and** inputs x of length $s_2'(n_i)$, N first guesses a string y_{hard} of length $O(t(n_i))$ and runs $V(x_{hard}, y_{hard})$. If V rejects, then N *rejects*. Otherwise, N will use y_{hard} as a hard function in the pseudorandom generator $G(\cdot,\cdot)$ of Theorem 2.1, for fooling circuits of size $w(n)^{1/g} = s_2(s_2(s_2(n)))^2$.

Fix a constant $a \geq 1$ such that $G(y_{hard},\cdot)$ runs in $O(|y_{hard}|^a)$ time. Let $\ell_i \in \{n_i, s_2'(n_i)\}$, depending on the current input length. Our N constructs a circuit $C_x(\cdot,\cdot)$ simulating the deterministic $O(s_2(s_2(\ell_i)))$-time part of $M_1(x)$. In particular, the circuit C_x takes a "nondeterministic" input z of length at most $s_2(s_2(\ell_i))^2$, as well as "randomness" input r of length at most $s_2(s_2(\ell_i))^2$, and C has size at most $s_2(s_2(\ell_i))^2 \leq s_2(s_2(s_2(n_i)))^2$. Then N nondeterministically guesses an input z, and enumerates all $s = |y_{hard}|^g$ seeds to the generator $G(y_{hard},\cdot)$, obtaining output strings $r_1,\ldots,r_s$ in $O(|y_{hard}|^{a+g}) = O(t(n_i)^{a+g})$ time. (We emphasize that this is done for both $\ell_i = n_i$ and $\ell_i = s_2'(n_i)$.) Finally, N computes the probability that $C_x(y_{hard}, r_j)$ accepts over all $j = 1,\ldots,s$. If this probability is greater than $1/2$, then N *accepts* else it *rejects*.

In total, N takes time $O(s_2(s_2(n_i))^2 + t(n_i)^{a+g} + t(n_i)^g \cdot s_2(s_2(n_i))^2) = O(t(n_i)^{a+g})$ on inputs of length n_i. Note the y_{hard} for an input x_{hard} of length n_i has circuit complexity at least $s_2(s_2(s_2(n_i)))^{2g}$. Since the circuit $C_x(\cdot,\cdot)$ for inputs x of length $\ell_i = s_2'(n_i)$ has size $s_2(s_2(s_2'(n_i)))^2 \leq s_2(s_2(s_2(n_i)))^2$, the same advice x_{hard} can *also* be used with the generator $G(\cdot,\cdot)$ to derandomize M_1 on inputs of length $s_2'(n_i)$ as well. Thus on inputs of length $s_2'(n_i)$, N takes $n_i + 2d' \cdot s_2'(n_i)$ advice and runs in time

$$O(s_2(s_2(s_2(n_i)))^2 + t(n_i)^{a+g} + t(n_i)^g \cdot s_2(s_2(s_2(n_i)))^2) \leq O(t(n_i)^{a+g}) \leq O(t(s_2'(n_i))^{a+g}),$$

where in the next-to-last inequality, we have applied constraint (b).

If we set $e \geq a+g$, then by (4), the nondeterministic $O(t(n)^e)$-time algorithm N (using $(2d'+1)n$ bits of advice) described in the previous paragraph has circuits of size $s((2d'+2)n)$. In particular, for almost every input length in the set $\{n_i\}$, the language L_1 has circuits of size $s((2d'+2)n_i)$ on n_i-bit inputs, **and** has circuits of size $s((2d'+1)\cdot s_2'(n_i) + n_i) \leq s((2d'+2)\cdot s_2'(n_i))$ on $s_2'(n_i)$-bit inputs.

For $e \geq 2d'+2$, this contradicts the circuit lower bound of Theorem 3.1. $\square$

5 Applications

Now we turn to showing how the new Easy Witness Lemmas imply stronger algorithm-to-lower bound connections. Recall a nondeterministic algorithm for GAP $\mathscr{C}$ UNSAT outputs *yes*, *no*, or *don't know* on every computation path, it outputs *no* on some computation path if its input circuit has at least $1/4$ of its assignments satisfying, outputs *yes* on some path if its input circuit is unsatisfiable, and for every input circuit, the algorithm never outputs *yes* on some path and *no* on another path.

Reminder of Theorem 1.1 *Let $\mathscr{C}$ be typical, and let $\varepsilon \in (0,1)$. Suppose* GAP $\mathscr{C}$ UNSAT *on n-input, $2^{\varepsilon n}$-size circuits is solvable in nondeterministic $O(2^{(1-\varepsilon)n})$ time. Then there is a $c \geq 1$ such that for all k, $\mathsf{NTIME}[n^{ck^4/\varepsilon}]$ does not have n^k-size $\mathscr{C}$-circuits.*

Proof. The proof is similar to earlier arguments [Wil10, Wil11, SW13]. Let $\varepsilon \in (0,1)$ be given. Assume

(A) for all $c \geq 1$, there is a $k \geq 1$ such that $\mathsf{NTIME}[n^{ck^4/\varepsilon}]$ has n^k-size $\mathscr{C}$-circuits, and

(B) GAP $\mathscr{C}$ UNSAT on n-input $2^{\varepsilon n}$-size circuits is in $O(2^{(1-\varepsilon)n})$ time.

Let c be sufficiently large in the following, and let

$$t(n) = n^{ck^4/\varepsilon}$$

for notational convenience. Since $\mathscr{C}$ is evaluatable (given a circuit from the class, we can evaluate it on an input in polynomial time), from (A) it follows that $\mathsf{NTIME}[t(n)]$ has $n^{O(k)}$-size *unrestricted* circuits (e.g., over the basis AND/OR/NOT of fan-in two). By the Easy Witness Lemma for NP (Lemma 1.2), $\mathsf{NTIME}[t(n)]$ has $n^{O(k^3)}$-size *witness* circuits for every NP verifier. We choose the PCP verifier of Ben-Sasson and Viola [BSV14], which yields proofs which are quasi-linear for any running time $T(n)$. In particular, for every verifier algorithm $V(x,y)$ running in time $T(n) \geq n$, they give an algorithm A running in $\text{poly}(n, \log T(n))$ time which, given $x \in \{0,1\}^n$, outputs an *oracle* circuit C_x^O with the following properties:[4]

- C_x^O has $\ell = \log_2(T(n)) + O(\log\log T(n))$ inputs, $\text{poly}(n, \log T(n))$ size, and each copy of the oracle gate O in C_x^O has ℓ inputs as well.
- If there is a y of length $T(n)$ such that $V(x,y)$ accepts, then there is an oracle $O : \{0,1\}^\ell \to \{0,1\}$ such that C_x^O is unsatisfiable.
- If $V(x,y)$ rejects on all y of length $T(n)$, then for every oracle $O : \{0,1\}^\ell \to \{0,1\}$, C_x^O has at least $2^\ell \cdot (1 - 1/n)$ satisfying assignments.

Note the oracle O acts as a *witness* for the PCP verifier which runs the above transformation on x, and checks whether C_x^O is unsatisfiable or not by guessing random inputs to C_x^O.

Take a language $L \in \mathsf{NTIME}[t(n)] - \mathsf{NTIME}[t(n)^{1-\varepsilon/2}]$ such that $L \subseteq \{1^n \mid n \geq 0\}$ [SFM78, Žák83], and let V be a $O(t(n))$-time verifier for L. On an input 1^n, let N be a nondeterministic algorithm which:

1. runs the algorithm A in $\text{poly}(n, \log t(n))$ time to produce a circuit C_n^O,
2. guesses a witness circuit W_n of size $n^{O(k^3)}$ encoding the oracle O for the PCP verifier,
3. plugs the circuit W_n in place of the oracle O in the circuit C_n^O, obtaining a circuit of size $n^{O(k^3)} \cdot \text{poly}(n, \log t(n)) \leq n^{O(k^3)}$,
4. checks satisfiability of $C_n^{W_n}$ by exhaustive search, in time $2^\ell \cdot n^{O(k^3)} \leq t(n) \cdot \text{poly}(\log t(n)) \cdot n^{O(k^3)} \leq n^{ck^4/\varepsilon + O(k^3)}$.

We will show how to simulate N in nondeterministic time $t(n)^{1-\varepsilon} \cdot \text{poly}(\log t(n))$, implying a contradiction. If we had an unrestricted CIRCUIT SAT algorithm or GAP CIRCUIT UNSAT algorithm running in $O(2^{\ell(1-\varepsilon)}) \leq t(n)^{1-\varepsilon+o(1)}$ time, this would be easy: we could just use the nondeterministic GAP CIRCUIT UNSAT algorithm to speed up item 4 in the description above. Because we only have a GAP $\mathscr{C}$ UNSAT algorithm, and $\mathscr{C}$ might be weaker than unrestricted circuits (as far as we know), the argument becomes more complicated.

Consider the task:

> EVAL-GATE: *Given a circuit C^O of n size, a circuit of size at most n encoding an oracle O, an input x of length ℓ, and an integer $i = 1, \ldots, n$, what is the output of the i-th gate of C^O evaluated on x?*

Clearly EVAL-GATE is computable in polynomial time. By assumption (A), EVAL-GATE has $\mathscr{C}$-circuits of size $n^{O(k)}$. In lieu of item 4 in the description of N, our new nondeterministic algorithm N' guesses a $\mathscr{C}$-circuit E computing EVAL-GATE which can take C_n^O and W_n as inputs; since both have size $n^{O(k^3)}$, a $\mathscr{C}$-circuit of $n^{O(k^4)}$ suffices. For every $i = 1, \ldots, s$ of $C_n^{W_n}$ (where $s = n^{O(k^3)}$), $E(C_n^O, W_n, x, i)$ outputs the value of the i-th gate of $C_n^{W_n}(x)$; by convention, we let the s-th gate be the output of the circuit. Assuming E is a circuit correctly computing EVAL-GATE on the appropriate input length, it is easy to see that

$$E(C_n^O, W_n, x, s) = 1 \iff C_n^{W_n}(x) = 1.$$

[4] This is not formally stated in their article, but it is immediate from their Theorem 1.1.

Without loss of generality, assume all gates of $C_n^{W_n}$ have a fixed gate type (e.g., NAND). For each gate i of $C_n^{W_n}$, let $i_1, i_2 < i$ be the indices of the two gates of $C_n^{W_n}$ whose outputs are the two inputs to gate i. Let $F_i(g_i, g_{i_1}, g_{i_2})$ be the canonical 3-CNF on three variables (where g_j corresponds to the output of gate j) which is true if and only if the outputs of gates i_1 and i_2 are consistent with the output of gate i. Now consider the circuit

$$D(x) := \neg \left(\bigwedge_{i=1}^{s} F_i(E(C_n^O, W_n, x, g_i), E(C_n^O, W_n, x, g_{i_1}), E(C_n^O, W_n, x, g_{i_2})) \right).$$

It is easy to see that $D(x) = 0$ if and only if E is *consistent* for all gates of $C_n^{W_n}$ on the input x; that is, E claims correct outputs for every gate of the circuit $C_n^{W_n}$ on the input x. From this, we have the consistency condition:

$$\text{for all } x,\ D(x) = 0 \text{ implies that } E(C_n^O, W_n, x, s) = C_n^{W_n}(x). \tag{6}$$

Furthermore, since $\mathscr{C}$ is closed under conjunctions and projections, given E, C_n^O, and W_n, the circuit D can be computed in time $n^{O(k^4)}$, where D is a circuit from $\mathscr{C}$. By our choice of $t(n)$ (with sufficiently large c),

$$n^{O(k^4)} \le 2^{\varepsilon \ell} \le t(n)^{\varepsilon} \cdot \operatorname{poly}(\log t(n)).$$

So the circuit D has ℓ inputs and size $2^{\varepsilon \ell}$, so our nondeterministic algorithm N' can run the nondeterministic GAP $\mathscr{C}$ UNSAT algorithm on D, assumed by (B). N' *rejects* if the nondeterministic GAP-UNSAT algorithm outputs *no* or *don't know* on D. Otherwise, N' *accepts* if and only if the nondeterministic GAP-UNSAT algorithm outputs *yes* on $E(C_n^O, W_n, \cdot, s)$ (with ℓ free inputs).

The running time of N' is

$$n^{O(k^4)} + 2^{(1-\varepsilon)\ell} \le t(n)^{1-\varepsilon} \cdot \operatorname{poly}(\log t(n)).$$

By the properties of the PCP verifier and our GAP-UNSAT algorithm, we have:

- If $1^n \in L$, then there is a W_n and an E such that D is unsatisifiable. Then, N' accepts 1^n on the correct guesses of W_n and E. In particular, the GAP-UNSAT algorithm will report *yes* on D (since D is unsatisfiable) on some path, and report *yes* on $E(C_n^O, W_n, \cdot, s)$ on some path as well.
- If $1^n \notin L$, for all choices of W_n, the circuit $C_n^{W_n}$ has at least $2^{\ell} \cdot (1 - 1/n)$ satisfying assignments. The algorithm N' guesses a circuit E for EVAL-GATE and constructs the circuit D. If our GAP-UNSAT algorithm returns *no* or *don't know* on D, then N' rejects by definition. If GAP-UNSAT returns *yes* on some path, then it must be that D has *less than* $2^{\ell}/4$ satisfying assignments (if it had more, the GAP-UNSAT algorithm would return *no* on some path, and therefore never output *yes* on any path, by definition). So $D(x) = 0$ on at least $3/4$ of the inputs x. By (6), $E(C_n^O, W_n, x, s) = C_n^{W_n}(x)$ for at least $3/4$ of the inputs x. Since $C_n^{W_n}$ has at least $2^{\ell} \cdot (1 - 1/n)$ satisfying assignments, it must be that the circuit $E(C_n^O, W_n, \cdot, s)$ of ℓ inputs has at least $2^{\ell} \cdot (1 - 1/n - 1/4) \ge 2^{\ell}/4$ satisfying assignments. Therefore the GAP-UNSAT algorithm on the circuit $E(C_n^O, W_n, \cdot, s)$ will **not** return *yes* on any computation path (because it must return *no* on some path, and it never outputs contradictory answers on the same circuit). Thus N' rejects 1^n on all computation paths.

In summary, N' correctly decides the language L. This is a contradiction, as N' runs in $t(n)^{1-\varepsilon} \cdot \operatorname{poly}(\log t(n))$ time. □

Remark 1 *Fortnow and Santhanam [FS16] proved that for all polynomials $t(n)$, there is an $\alpha > 0$ and an $L \in \mathsf{NTIME}[t(n)]$ that is not in $\mathsf{NTIME}[t(n)^{1-\varepsilon/2}]/n^{\alpha}$. If we use this L in the above proof, then the nondeterministic* GAP $\mathscr{C}$ SAT *algorithm could even use $2^{\beta n}$* bits of advice *on n-input circuits of size $2^{\varepsilon n}$ (for* *small enough $\beta > 0$ depending on ε and k), and still yield the desired lower bound.*

Reminder of Theorem 1.2 *Let $\mathscr{C}$ be typical, and let $\varepsilon \in (0,1)$. Suppose* GAP $\mathscr{C}$ UNSAT *on n-input, 2^{n^ε}-size circuits is in nondeterministic $O(2^{n-n^\varepsilon})$ time. Then for all k, there is a $c \geq 1$ such that $\mathsf{NTIME}[2^{\log^{ck^4/\varepsilon} n}]$ does not have $2^{\log^k n}$-size $\mathscr{C}$-circuits.*

Proof. We use the exact same strategy as Theorem 1.1, with the following minor modifications. Assume

(A) for all c, there is a $k \geq 1$ such that $\mathsf{NTIME}[2^{\log^{ck^4/\varepsilon} n}]$ has $2^{\log^k n}$-size $\mathscr{C}$-circuits, and
(B) GAP $\mathscr{C}$ UNSAT on n-input 2^{n^ε}-size circuits is in $O(2^{n-n^\varepsilon})$ time.

Set $t(n) = 2^{\log^{ck^4/\varepsilon} n}$. As argued in Theorem 1.1, by the Easy Witness Lemma for NQP (Lemma 1.2), $\mathsf{NTIME}[t(n)]$ has $2^{O(\log^{k^3} n)}$-size *witness* circuits for every NP verifier. Use the PCP verifier of Ben-Sasson and Viola [BSV14] as before which outputs a circuit C_n^O of $\text{poly}(n)$ size with number of inputs

$$\ell = (\log_2 n)^{ck^4/\varepsilon} + O(\log\log n).$$

Take $L \in \mathsf{NTIME}[t(n)] - \mathsf{NTIME}[t(n)/2^{\log^{\varepsilon/2} t(n)}]$ such that $L \subseteq \{1^n \mid n \geq 0\}$ [SFM78, Žák83], and let V be a $O(t(n))$-time verifier for L. On an input 1^n, let N be a nondeterministic algorithm which:

1. runs in $\text{poly}(n)$ time to produce a circuit C_n^O,
2. guesses a witness circuit W_n of size $2^{O(\log^{k^3} n)}$ encoding the oracle O for the PCP verifier,
3. plugs the circuit W_n in place of the oracle O in the circuit C_n^O, obtaining a circuit of size $2^{O(\log^{k^3} n)} \cdot \text{poly}(n) \leq 2^{O(\log^{k^3} n)}$,
4. checks satisfiability of $C_n^{W_n}$ by exhaustive search, in time $2^\ell \cdot 2^{O(\log^{k^3} n)} \leq 2^{\log^{ck^4/\varepsilon} n + O(\log^{k^3} n)}$.

As before, we show how to simulate N faster. We define the EVAL-GATE problem as before; by assumption (A), it has $\mathscr{C}$-circuits of size $2^{\log^k n}$. Thus there is a circuit E of size $2^{O(\log^{k^4} n)}$ which can take W_n and C_n^O as input and faithfully simulate EVAL-GATE. Also as before, we construct the circuit D of size $2^{O(\log^{k^4} n)}$ and of ℓ inputs. For sufficiently large c, the size of D is

$$2^{O(\log^{k^4} n)} \leq 2^{\log^\varepsilon t(n)} \leq 2^{((\log^{ck^4} n) + O(\log\log n))^\varepsilon} \leq 2^{\ell^\varepsilon},$$

so the assumed nondeterministic GAP $\mathscr{C}$ UNSAT algorithm can be run on D, and it can be run on the circuit $E(C_n^O, W_n, \cdot, g_i)$, obtaining yes/no answers in nondeterministic time

$$O(2^{\ell - \ell^\varepsilon}) \leq t(n)/2^{\log^\varepsilon t(n)}.$$

Following the rest of the proof of Theorem 1.1, we can decide L in nondeterministic time $t(n)/2^{\log^\varepsilon t(n)}$, which is a contradiction. $\square$

Finally, to conclude lower bounds such as $\mathsf{NQP} \not\subset \mathsf{ACC} \circ \mathsf{THR}$ (Theorem 1.3), we only have to appeal to the $\mathsf{ACC} \circ \mathsf{THR}$-SAT algorithm:

Theorem 5.1 ([Wil14]) *For all integers d and $m \geq 2$, there is an $\varepsilon > 0$ and an $O(2^{n-n^\varepsilon})$-time deterministic algorithm for the satisfiability problem on depth-d size-2^{n^ε} circuits over unbounded fan-in AND, OR, and MODm gates, with linear threshold gates at the bottom layer.*

6 Conclusion

We have shown how slightly faster CIRCUIT SAT algorithms, even those distinguishing unsatisifiable circuits from circuits with many satisfying assigments, can also imply circuit lower bounds for the nondeterministic classes NP and NQP. This suggests that the age-old problem of proving that NP does not have linear-size circuits may well be in striking distance, or at least that we may find non-linear lower bounds for weak circuit classes in the near future (however, note $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[O(n)]$ is still open, and one has to prove this first!). We would like to draw attention to a few related open problems:

NP versus ACC? The main obstacle to proving $\mathsf{NP} \not\subset \mathsf{ACC}$ is the lack of a GAP ACC UNSAT algorithm which runs in $2^{(1-\varepsilon)n}$ time on $2^{\varepsilon n}$-size circuits, for some $\varepsilon > 0$. The ACC-SAT algorithm used in the NQP lower bound applies the well-known transformation of ACC to $\mathsf{SYM} \circ \mathsf{AND}$ [Yao90, BT94], which blows up the circuit by a quasi-polynomial factor; because of this blow-up it does not seem possible to get a much faster GAP ACC UNSAT algorithm using this transformation. But perhaps a better SAT algorithm can be derived under the assumption $\mathsf{NP} \subset \mathsf{ACC}$.

EXP or BPEXP lower bounds? We now have interesting circuit lower bounds for Nondeterministic Quasi-Polynomial Time (NQP). We do not believe that $\mathsf{NQP} \subseteq \mathsf{EXP}$, but it seems very unlikely that EXP-complete problems are solvable in NQP; EXP-complete problems ought to be just as hard. Can the framework be extended further to show that faster CIRCUIT SAT or GAP UNSAT algorithms imply EXP lower bounds? How about lower bounds for the randomized exponential time class, BPEXP? It is known that if we have non-trivial *learning algorithms* for circuit classes with appropriate parameters, then BPEXP lower bounds follow [FK09, OS17]. Perhaps SAT algorithms can help expedite learning?

Reduce $\mathbf{E}^{\mathbf{NP}}$? In several settings [Wil10, JMV15, CGI+16], it is known that faster SAT algorithms for weak classes of formulas would imply a circuit lower bound for the (huge) class E^{NP}. For example, a 1.9^n-time nondeterministic CNF-UNSAT algorithm would imply E^{NP} does not have linear-size "Valiant-series-parallel" circuits [JMV15, CGI+16]. When can the gigantic class E^{NP} be reduced to a smaller class?

Acknowledgements. We thank Marco Carmosino, Valentine Kabanets, and Igor Carboni Oliveira for helpful comments.

References

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

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BSV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *ICALP*, pages 163–173, 2014.

[BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, pages 350–366, 1994.

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 261–270, 2016.

[CP16] Shiteng Chen and Periklis A. Papakonstantinou. Depth-reduction for composites. In *FOCS*, pages 99–108, 2016.

[Din15] Ning Ding. Some new consequences of the hypothesis that p has fixed polynomial-size circuits. In *International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 75–86. Springer, 2015.

[FK09] Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences*, 75(1), 2009.

[FS16] Lance Fortnow and Rahul Santhanam. New non-uniform lower bounds for uniform classes. In *CCC*, pages 19:1–19:14, 2016.

[FS17] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Inf. Comput.*, 256:149–159, 2017.

[FSW09] Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed-polynomial size circuit bounds. In *CCC*, pages 19–26. IEEE, 2009.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[JMV15] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *ICALP*, pages 749–760. Springer, 2015.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

[KL82] Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

[KW98] Johannes Kobler and Osamu Watanabe. New collapse consequences of np having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1998.

[Lip94] Richard Lipton. Some consequences of our failure to prove non-linear lower bounds on explicit functions. In *Structure in Complexity Theory*, pages 79–87, 1994.

[MNW99] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, LNCS 1627, pages 210–220. Springer, 1999.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *CCC*, pages 18:1–18:49, 2017.

[San07] Rahul Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM Journal on Computing*, 39(3):1038–1061, 2009. Preliminary version in STOC'07.

[SFM78] Joel Seiferas, Michael Fischer, and Albert Meyer. Separating nondeterministic time complexity classes. *JACM*, 25(1):146–167, 1978.

[SW13] Rahul Santhanam and Ryan Williams. On uniformity and circuit lower bounds. *Computational Complexity*, 23(2):177–205, 2013. Preliminary version in CCC'13.

[TV02] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. Preliminary version in CCC'02.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[Wil14] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *STOC*, pages 194–202, 2014.

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal on Computing*, 45(2):497–529, 2016.

[Wil11] Ryan Williams. Nonuniform ACC circuit lower bounds. *JACM*, 61(1):2, 2014. See also CCC'11.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013. See also STOC'10.

[Yao90] Andrew Chi-Chih Yao. On ACC and threshold circuits. In *FOCS*, pages 619–627, 1990.

[Žák83] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.