Algorithms versus Circuit Lower Bounds

Igor C. Oliveira*

Department of Computer Science
Columbia University

oliveira@cs.columbia.edu

September 1, 2013

Abstract

Different techniques have been used to prove several transference theorems of the form "non-trivial algorithms for a circuit class $\mathcal{C}$ yield circuit lower bounds against $\mathcal{C}$". In this survey we revisit many of these results. We discuss how circuit lower bounds can be obtained from derandomization, compression, learning, and satisfiability algorithms. We also cover the connection between circuit lower bounds and useful properties, a notion that turns out to be fundamental in the context of these transference theorems. Along the way, we obtain a few new results, simplify several proofs, and show connections involving different frameworks. We hope that our presentation will serve as a self-contained introduction for those interested in pursuing research in this area.

1

Contents

1   Introduction

This survey deals with two fundamental problems in theoretical computer science: the design of nontrivial algorithms for difficult computational tasks, and the search for unconditional proofs that some natural computational problems are inherently hard (more specifically, do not admit polynomial size circuits).

Perhaps surprisingly, these problems are deeply related. For instance, it follows from the work of Karp and Lipton [KL80] (attributed to Meyer) that if 3-SAT admits a polynomial time algorithm, then there are problems solved in exponential time that cannot be computed by polynomial size circuits. On the other hand, it is known that constructive proofs of circuit lower bounds lead to algorithms breaking exponentially hard pseudorandom generators that are conjectured to exist (Razborov and Rudich [RR97]).

The last decade has produced several additional *transference theorems*[1] of this form, under many different algorithmic frameworks. For instance, the existence of subexponential time learning algorithms for a class of functions $\mathcal{C}$ leads to circuit lower bounds against $\mathcal{C}$ (Fortnow and Klivans [FK09]). In a different domain, it is known that the design of subexponential time deterministic algorithms for problems with efficient randomized algorithms implies circuit lower bounds that have eluded researchers for decades (Kabanets and Impagliazzo [KI04]). More recently, it has been shown that new circuit lower bounds can be obtained from efficient compression algorithms (Chen et al. [CKK+13]), not to mention the connection between satisfiability algorithms and circuit lower bounds (Williams [Wil10], [Wil11], [Wil13b]). Several additional results have appeared in the literature ([KvMS12], [AvM11], [AAB+10], [AGHK11], [HH11], [KKO13], among others). For a gentle introduction to some of these connections, see Santhanam [San12].

These results are interesting for several reasons. For instance, as far as we know, there may be an efficient compression scheme that works well for any string possessing some structure, or circuits of polynomial size can be learned in quasipolynomial time by a very complicated learning algorithm. Nevertheless, the transference theorems discussed before show that even if complicated tasks like these are actually easy, some natural computational problems are inherently hard (even for non-uniform algorithms).

How could one prove (unconditionally) that a natural computational problem is hard? How would a mathematical proof of such result look like? This is one of the most fascinating questions of contemporary mathematics, and it is related to deep problems about algorithms, combinatorics, and mathematical logic (cf. Krajícek [Kra95], Immerman [Imm99], Cook and Nguyen [CN10]).

It turns out that the connection between algorithms and circuit lower bounds ("transference theorems") can be used to prove *new* circuit lower bounds that had resisted the use of more direct approaches for decades. Let $\mathcal{C}$ be a class of circuits, such as $\mathsf{AC}^0, \mathsf{TC}^0, \mathsf{NC}^1$, etc. We say that a satisfiability algorithm for $\mathcal{C}$ is *nontrivial* if it runs in time time $2^n/s(n)$, for some function $s(n) \gg \mathsf{poly}(n)$. Building on work done by many researchers, Williams ([Wil11], [Wil10]) proved the following transference theorem: the existence of a nontrivial $\mathcal{C}$-SAT algorithm implies $\mathsf{NEXP} \nsubseteq \mathcal{C}[\mathsf{poly}]$. In other words, faster satisfiability algorithms lead to languages computed in nondeterminstic exponential time that cannot be computed by polynomial size circuits from $\mathcal{C}$.

Most importantly, by designing a new ACC-SAT algorithm, Williams [Wil11] was able to obtain a circuit lower bound for the circuit class ACC.[2] Moreover, this is the *only* known proof of this result.

---

[1] In other words, these theorems show that one can transform an algorithmic result into a circuit lower bound, i.e., they allow us to transfer a result from one area to another.

[2] This is the class of languages computed by polynomial size constant-depth circuits consisting of AND, OR, NOT

Other approaches that have been proposed are also based on the design of new ACC algorithms (Chen et al. [CKK+13]). While his result is still weak compared to the main open problems in circuit complexity and complexity theory in general, it is a landmark in our understanding of the connection between nontrivial algorithms and the existence of hard computational problems.

Can we extend this technique to prove stronger circuit lower bounds? Is there any connection between Williams' transference theorem and other similar results discussed before? This survey is motivated by these questions. We break the presentation into two parts. The first part is a fast-paced introduction to some known results connecting algorithms to circuit lower bounds. The second part of this survey presents complete proofs for most of these theorems, along with some extensions that may be of independent interest. Of course we are not able to cover every result that relates algorithms to circuit lower bounds, but we tried to describe representative results from many areas.

We stress that we focus on *generic* connections between faster algorithms and circuit lower bounds, instead of particular techniques that have found applications in both areas (Fourier representation of boolean functions [LMN93], satisfiability coding lemma [PPZ99], random restriction method [CKK+13], etc.). For the reader with basic background in complexity theory, our presentation is essentially self-contained.

1.1 A summary of some known results

1.1.1 Satisfiability algorithms and circuit lower bounds

The connection between algorithms for hard problems and circuit lower bounds has been known for decades. More precisely, a collapse theorem attributed to Meyer [KL80] states that if $EXP \subseteq P/poly$ then $EXP = \Sigma_2^p$ (recall this is the second level of PH, the polynomial time hierarchy). On the other hand, it is not hard to prove that if $P = NP$ then $P = \Sigma_2^p = PH$. Together, the assumptions that there are efficient algorithms for NP-complete problems and that every problem in EXP admits polynomial size circuits lead to $P = EXP$, a contradiction to the deterministic time hierarchy theorem. In other words, if there exists efficient algorithms for 3-SAT, it must be the case that $EXP \not\subseteq P/poly$.[3] Similar transference results can be obtained from the assumption that there are subexponential time algorithms for 3-SAT (i.e., with running time $2^{n^{o(1)}}$).

The existence of such algorithms is a very strong assumption. The best known algorithms for k-SAT run in time $2^{n(1-\delta(k))}$, for some fixed constant $\delta(k) > 0$ that goes to zero as k goes to infinity (cf. Dantsin and Hirsch [DH09]). These algorithms offer an exponential improvement over the trivial running time $\tilde{O}(2^n)$. If we only require the running time to be faster than exhaustive search ("nontrivial"), then improved algorithms are known for many interesting circuit classes (see for instance [San10], [CKK+13], [ST13], [IPS12], [BIS12], [IMP12]). For an introduction to some of these algorithms, see Schneider [Sch13].

It makes sense therefore to consider more refined versions of the transference theorem for satisfiability algorithms. This is precisely the first result in this direction obtained by Williams [Wil10]: the existence of nontrivial algorithms deciding the satisfiability of polynomial size circuits is enough to imply $NEXP \not\subseteq P/poly$. Unfortunately, $P/poly$ is a very broad class, and the algorithms mentioned before do not work or have trivial running time on such circuits.

---

and MOD_m gates (for a fixed integer $m \in \mathbb{N}$). Every gate other than NOT is allowed to have unbounded fan-in.

[3]Using the fact that $P = PH$ implies the collapse of the exponential time hierarchy to EXP, an even stronger consequence can be obtained. We omit the details.

In follow up work, Williams [Wil11] extended his techniques from [Wil10] to prove a more general result that holds for most circuit classes.

Proposition 1.1 ("SAT algorithms yield circuit lower bounds, I" [Wil11]).
Let $\mathcal{C}$ be a class of circuit families that is closed under composition (the composition of two circuits from $\mathcal{C}$ is also in $\mathcal{C}$) and contains AC^0. There is a $k > 0$ such that, if satisfiability of $\mathcal{C}$-circuits with n variables and n^c size can be solved in $O(2^n/n^k)$ time for every c, then $\mathsf{NEXP} \not\subseteq \mathcal{C}[\mathsf{poly}(n)]$.

In addition, he provided a nontrivial algorithm for $\mathsf{ACC}[2^{n^\delta}]$ (the class of ACC circuits of size 2^{n^δ}), where $\delta = \delta(d, m) > 0$ depends on the depth of the circuit and the modulo gate. Altogether, these results imply the following circuit lower bound.

Corollary 1.2. $\mathsf{NEXP} \not\subseteq \mathsf{ACC}$.

Subsequent work of Williams [Wil13b] has extended these techniques to prove the following stronger transference theorem, which provides better circuit lower bounds[4].

Proposition 1.3 ("SAT algorithms yield circuit lower bounds, II" [Wil13b]).
Let $\mathcal{C}$ be a class of circuit families that is closed under composition and contains AC^0. There is a $k > 0$ such that, if satisfiability of $\mathcal{C}$-circuits with n variables and $n^{\log^c n}$ size can be solved in $O(2^n/n^k)$ time for every c, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$.

Besides, the following strengthening of Corollary 1.2 is proven in the same paper (the first statement is implicit in his proof).

Corollary 1.4. $\mathsf{E} \not\subseteq \mathsf{ACC}[n^{\log n}]$ *or* $\mathsf{Quasi\text{-}NP} \cap \mathsf{i.o.Quasi\text{-}coNP} \not\subseteq \mathsf{ACC}[n^{\log n}]$. *In particular,* $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathsf{ACC}[n^{\log n}]$.

The proof of these transference theorems has been simplified since then. In Santhanam and Williams [SW12], self-reduction (cf. Allender and Koucký [AK10]) is used to obtain an equivalent circuit from a smaller circuit class given an arbitrary NC^1 circuit (under some assumptions). This simplifies one of the main technical lemmas from [Wil11].

1.1.2 Constructivity and circuit lower bounds

There are three significant barriers to circuit lower bound proofs: relativization (Baker, Gill, and Solovay [BGS75]), natural proofs (Razborov and Rudich [RR97]), and algebrization (Aaronson and Wigderson [AW09]). Roughly speaking, these barriers can be interpreted as follows: some proof methods are too general, and if a lower bound can be obtained by one of such techniques alone, then we get a contradiction to some known result or a widely believed conjecture[5]. As explained by Williams [Wil11], his lower bound proof combines several methods used in modern complexity theory, and each one avoids a particular barrier[6].

It was proven by Razborov and Rudich that most of the circuit lower bound proofs known at the time proceeded (at least implicitly) as follows. There is a circuit class $\mathcal{C}$ (say, AC^0) that one

---

[4]We use $\mathsf{NE} \cap \mathsf{i.o.coNE}$ instead of $\mathsf{NE} \cap \mathsf{coNE}$ in the statement of Proposition 1.3 because the proof described in [Wil13b] requires this extra condition [Wil13a].

[5]These barriers can also be interpreted as independence results for some formal theories ([AIV92], [Raz95], [IKK09]).

[6]We stress however that there is no widely believed conjecture that leads to pseudorandom function families in ACC, and this is an interesting open problem. As far as we know, there may exist a natural proof that $\mathsf{P} \not\subseteq \mathsf{ACC}$.

wants to separate from a complexity class Γ (say, P). In order to do that, one defines a property $\mathcal{P}$ of boolean functions (i.e., a subset of all boolean functions), and prove that *no* function in $\mathcal{C}$ satisfies $\mathcal{P}$, while there exists some hard function $h \in \Gamma$ for which $\mathcal{P}(h) = 1$ (in this case, we say that $\mathcal{P}$ is *useful* against $\mathcal{C}$). For instance, every AC^0 function simplifies after an appropriate random restriction ([FSS84], [Yao85], [Hås87]), while the parity function is still as hard as before.

As it turns out, for the property $\mathcal{P}$ defined in these proofs, there is an efficient algorithm $\mathcal{A}$ (with respect to the size of the truth-table of f) that is able to decide whether $\mathcal{P}(f) = 1$. Such properties are referred to as *constructive* properties. In addition, it is usually the case that a random function satisfies $\mathcal{P}$ with non-negligible probability ($\mathcal{P}$ satisfies the *denseness* condition). These two conditions imply that $\mathcal{A}$ can be used to distinguish a function in $\mathcal{C}$ from a random function. Put another way, if there exists a proof of this form that $\Gamma \nsubseteq \mathcal{C}$, then there is no pseudorandom function family in $\mathcal{C}$.

However, if some number-theoretic problems are exponentially hard on average (an assumption believed to be true by many researchers), then there are pseudorandom functions in circuit classes as small as TC^0_4 (Naor and Reingold [NR04], Krause and Lucks [KL01]). As a consequence, such proofs (dubbed *natural proofs* in [RR97]) are not expected to prove separations for more expressive circuit classes. Unfortunately, most (if not all) known combinatorial proofs of circuit lower bounds implicitly define such properties, and this explains the lack of significant progress obtained so far for more general classes of circuits using these techniques only. The interested reader is referred to Chow [Cho11] and Rudich [Rud97] for further developments.

As a consequence, any circuit lower bound proof for more expressive classes must violate either the denseness or the constructivity condition. Williams [Wil13b] shed light into this problem, by proving that any separation of the form $\mathsf{NEXP} \nsubseteq \mathcal{C}$ is actually equivalent to exhibiting a *constructive* property $\mathcal{P}$ that is useful against $\mathcal{C}$.

Proposition 1.5 ("Constructivity is unavoidable", **informal** [Wil13b]).
Let $\mathcal{C}$ be a typical circuit class. Then $\mathsf{NEXP} \nsubseteq \mathcal{C}$ if and only if there exists a constructive property $\mathcal{P}$ that is useful against $\mathcal{C}$.

In other words, any lower bound proof against NEXP implies the existence of a property that is both useful and constructive, but not necessarily dense. As we explain later in the text, $\mathcal{P}$ is actually computed with a small amount of advice. We clarify this point in Section 1.2.2, where we discuss some additional results about useful properties and circuit lower bounds.

1.1.3 Additional transference theorems

As alluded to earlier, several additional transference theorems of the form "faster algorithms yield circuit lower bounds" have been discovered. In this section we describe some of these results in more detail. We focus on learning algorithms, derandomization, and algorithms for string compression.

Derandomization. There is a strong connection between the existence of pseudorandom generators and circuit lower bounds (see [Kab02], [Uma03]). Nevertheless, while there is evidence that pseudorandom generators are *necessary* in order to derandomize probabilistic algorithms (Goldreich [Gol11]), i.e. to prove that $\mathsf{P} = \mathsf{BPP}$, this is still open.

On the other hand, for the larger randomized complexity class MA, it is known that any derandomization (such as $\mathsf{MA} \subseteq \mathsf{NSUBEXP}$) implies superpolynomial circuit lower bounds for NEXP (Impagliazzo, Kabanets and Wigderson [IKW02]). Subsequent work of Kabanets and Impagliazzo

[KI04] shows that even the derandomization of a single, specific problem in BPP leads to some circuit lower bounds. More precisely, let PIT be the language consisting of all arithmetic circuits that compute the zero polynomial over $\mathbb{Z}$, and PERM be the problem of computing the permanent of integer matrices. We use SIZE[poly] to denote the set of languages computed by polynomial size boolean circuits. Similarly, let ASIZE[poly] be the family of languages computed by arithmetic circuits of polynomial size over $\mathbb{Z}$.

Proposition 1.6 ("Derandomization yields circuit lower bounds" [KI04]).
If PIT $\in$ NSUBEXP, *then at least one of the following results hold:*

(i) NEXP $\nsubseteq$ SIZE[poly]*; or*

(ii) PERM $\nsubseteq$ ASIZE[poly].

Aaronson and van Melkebeek [AvM11] proved a parameterized version of the result, in addition to showing that NEXP $\cap$ coNEXP can be used in place of NEXP. Another extension appears in Kinne, van Melkebeek and Shaltiel [KvMS12].

Learning algorithms. Fortnow and Klivans [FK09] were the first to exhibit a formal connection between learning algorithms and circuit lower bounds. Recall that a learning algorithm $\mathcal{A}$ is given restricted access to a fixed but arbitrary function f from a class of functions $\mathcal{C}$, and it should output a hypothesis h that is as close to f as possible. Distinct learning models provide difference access mechanisms to f, and impose specific requirements over h (h should be close to f, $h \equiv f$, etc.) and $\mathcal{A}$ (learner is randomized, deterministic, etc).

An exact learning algorithm is a *deterministic* algorithm that has access to a membership query oracle MQ^f and an equivalence query oracle EQ^f, and it is required to output a hypothesis h which agrees with f over all inputs[7].

Proposition 1.7 ("Learning yields circuit lower bounds" [FK09]).
Let $\mathcal{C}$ be a circuit class. If there exists a subexponential time exact learning algorithm for $\mathcal{C}$, then $\mathsf{E}^{\mathsf{NP}} \nsubseteq \mathcal{C}$.

The original proof of Proposition 1.7 relies on many complexity theoretic results. Subsequent work done by Harkins and Hitchcock [HH11] strengthened the conclusion to EXP $\nsubseteq \mathcal{C}$. Finally, Klivans, Kothari and Oliveira [KKO13] used a very simple argument to prove the essentially optimal result that exact learning algorithms for $\mathcal{C}[s(n)]$ running in time $t(n)$ lead to a circuit lower bound of the form DTIME[poly$(t(n))$] $\nsubseteq \mathcal{C}[s(n)]$.[8]

For *randomized* learning algorithms, much weaker results are known (a formal definition of the model is discussed in Section 5.3). For instance, *efficient* PAC learning algorithms are known to lead to circuit lower bounds against BPEXP, the exponential time analogue of BPP [FK09]. A slightly stronger result was obtained by Klivans et al. [KKO13], but the underlying techniques do not provide interesting results for randomized subexponential time algorithms. It is an interesting open problem to obtain such extension.

---

[7]On input $x \in \{0,1\}^n$, $\mathsf{MQ}^f(x)$ returns $f(x)$. On input a circuit c, EQ^f outputs "yes" if $c \equiv f$, otherwise it outputs an arbitrary input z such that $c(z) \neq f(z)$.
[8]The same result was obtained independently by Russell Impagliazzo and Valentine Kabanets [Kab13].

Truth-table compression. More recently, Chen et al. [CKK+13] considered the problem of designing efficient algorithms that obtain nontrivial compression of strings representing truth-tables from a circuit class $\mathcal{C}$. In other words, given a string $tt(f_n) \in \{0,1\}^N$, where $f_n : \{0,1\}^n \to \{0,1\}$ is a function from $\mathcal{C} \subseteq$ P/poly and $N = 2^n$, a compression algorithm should run in time poly(N) and output a circuit C over n inputs and size $\ll 2^n/n$ such that $tt(C) = tt(f)$. In the same paper, they observed that several circuit lower bounds proofs relying on the method of random restrictions yield efficient compression algorithms. On the other hand, they obtained the following transference theorem.

Proposition 1.8 ("Compression leads to circuit lower bounds" [CKK+13])**.**
Let $\mathcal{C}$ be a circuit class. Suppose that for every $c \in \mathbb{N}$ there is a deterministic polynomial-time algorithm that compresses a given truth table of an n-variate boolean function $f \in \mathcal{C}[n^c]$ to an equivalent circuit of size $o(2^n/n)$. Then NEXP $\nsubseteq \mathcal{C}$.

It follows from Proposition 1.8 that designing a compression algorithm for ACC would provide an alternative proof of Corollary 1.2. This is left as an interesting open problem by [CKK+13].

1.2 A guide to the results discussed in this survey

1.2.1 Lower bounds from satisfiability algorithms for low depth circuits

Let TC_2^0 denote the class of polynomial size circuits of depth two with gates corresponding to arbitrary linear threshold functions. As far as we know, it may be the case that NEXP $\subseteq \mathsf{TC}_2^0$. It makes sense therefore to see if the techniques used in the proof of Corollary 1.2 can be helpful in obtaining a separation against bounded-depth circuit classes of this form.

A more refined version of Proposition 1.1 discussed in [Wil11] shows that circuit lower bounds for circuits of depth d follow from satisfiability algorithms for depth $2d+O(1)$. We observe here that it is possible to obtain a tight transference theorem for satisfiability algorithms for constant-depth circuits. Let $\mathcal{C}_d$ be a circuit class consisting of circuits of depth d, and g be an arbitrary function. We write $g[k] \circ \mathcal{C}_d$ to denote the class of functions computed by circuits of depth $d+1$ consisting of a top layer gate g of fan-in k that is fed by k circuits from $\mathcal{C}_d$.

Theorem 1.9 ("SAT algorithms for depth $d+2$ yield circuit lower bounds for depth d")**.**
Let $\mathcal{C}$ be a reasonable circuit class. If there exists a nontrivial satisfiability algorithm for AND[3] $\circ$ OR[2] $\circ \mathcal{C}_d$[poly]*, then* NEXP $\nsubseteq \mathcal{C}_d$[poly]*.*

We define reasonable circuit classes in Section 2. This result can be obtained through a simple extension of the original technique used by Williams [Wil10]. In particular, our presentation avoids the technical details from [Wil11]. A similar theorem is described in Jahanjou, Miles and Viola [JMV13], but the argument they use is more involved. The proof of Theorem 1.9 and some additional remarks are presented in Section 3.

1.2.2 Useful properties and circuit lower bounds

We discuss in more detail the relation between circuit lower bounds and useful properties (Proposition 1.5). Useful properties play a fundamental concept in the context of transference theorems, as explained in the next section.

For a uniform complexity class Γ (such as P, NP, etc), we say that a property of boolean functions $\mathcal{P}$ is a Γ-property if it can be decided in Γ. We use $\Gamma/s(m)$ to denote the corresponding

8

complexity class with advice of size $s(m)$, where m is the size of the input. Recall that a property is useful against $\mathcal{C}$ if it distinguishes some hard function from all functions in $\mathcal{C}$ (a formal definition is presented in Section 2).

First we observe that nondeterminism is of no use in the context of useful properties, which is a somewhat surprising result. The proof of this fact relies on some ideas introduced by Williams [Wil13b].

Theorem 1.10 ("NP-property yields P-property")**.**
Let $\mathcal{C}$ be a circuit class. If there exists a NP-property that is useful against $\mathcal{C}[\mathsf{poly}]$, then there is a P-property that is useful against $\mathcal{C}[\mathsf{poly}]$.

Now we discuss in more detail the connection discovered by Williams (Proposition 1.5) between constructive useful properties (P-properties under our notation) and circuit lower bounds. It turns out that the statement of Proposition 1.5 requires a broader definition, one for which the algorithm deciding the property is allowed inputs of arbitrary size instead of size $N = 2^n$, where $n \in \mathbb{N}$. Put another way, the algorithm receives any string as input, and is allowed to parse its input size as $2^n + k$. Now it is free to interpret k as an advice string of length $\log N$. We clarify this issue here, and observe that Theorem 1.10 together with standard techniques imply the following characterization of NEXP circuit lower bounds.[9]

Theorem 1.11 ("Equivalence between NEXP lower bounds and useful properties")**.**
Let $\mathcal{C}$ be a circuit class. Then $\mathsf{NEXP} \nsubseteq \mathcal{C}[\mathsf{poly}]$ if and only if there exists a $\mathsf{P}/\log N$-property that is useful against $\mathcal{C}[\mathsf{poly}]$.

It makes sense therefore to investigate whether there exists an equivalence between useful properties computed without advice and circuit lower bounds. The following result holds.

Theorem 1.12 ("NE ∩ coNE lower bounds and useful properties")**.**
Let $\mathcal{C}$ be a circuit class. The following holds:

(i) *If $\mathsf{NE} \cap \mathsf{coNE} \nsubseteq \mathcal{C}[\mathsf{poly}]$ then there is a P-property that is useful against $\mathcal{C}[\mathsf{poly}]$.*

(ii) *If for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \nsubseteq \mathcal{C}[n^{\log n}]$.*

One direction follows from Theorem 1.10, while the other is implicit in Williams [Wil13b]. Given these results, the following conjecture seems plausible.

Conjecture 1.13 ("Equivalence between NE ∩ coNE lower bounds and useful properties?")**.**
Let $\mathcal{C}$ be a circuit class. Then $\mathsf{NE} \cap \mathsf{coNE} \nsubseteq \mathcal{C}[\mathsf{poly}]$ if and only if there exists a P-property that is useful against $\mathcal{C}[\mathsf{poly}]$.

We discuss how this conjecture relates to Williams' program for circuit lower bounds in Section 4.1. The results for useful properties are discussed in Section 1.2.2 of the survey.

---

[9]We stress that in this survey any algorithm that decides a property of boolean functions works over strings of length $N = 2^n$, where $n \in \mathbb{N}$.

1.2.3 Applications

It is possible to use the results mentioned in the previous sections to prove the propositions stated in Section 1.1.3. In particular, several transference theorems are in fact *connected*, and improvements in one framework propagates to other results.

The first application that we discuss is for compression algorithms, as investigated by Chen et al. [CKK$^+$13]. Observe that Proposition 1.8 shows circuit lower bounds for NEXP from exact compression of truth-tables of polynomial size circuits. As mentioned in the same paper, their result can be extended to show that even *lossy* compression algorithms lead to circuit lower bounds. We flesh out the details here.

We say that a circuit class $\mathcal{C}$ admits *lossy compression* algorithms if there exists an efficient algorithm $\mathcal{A}$ (over inputs of size $N = 2^n$) that when given as input a truth-table $tt(f)$ from $\mathcal{C}$, where $f : \{0,1\}^n \to \{0,1\}$, outputs a circuit C of size $o(2^n/n)$ such that $\Pr_x[C(x) = f(x)] \geq .51$. A more general definition is discussed in Section 5.1.

Theorem 1.14 ("Circuit lower bounds from lossy compression").
Let $\mathcal{C}$ be a circuit class. The following results hold.

(*i*) *If for every $c \in \mathbb{N}$ there exists a lossy compression algorithm for $\mathcal{C}[n^c]$, then* NEXP $\not\subseteq$ $\mathcal{C}[\mathsf{poly}(n)]$.

(*ii*) *If for every $c \in \mathbb{N}$ there exists a lossy compression algorithm for $\mathcal{C}[n^{\log^c n}]$, then* NE$\cap$i.o.coNE $\not\subseteq$ $\mathcal{C}[n^{\log n}]$.

In other words, any efficient algorithm for lossy compression of strings is either trivial on infinitely many input strings represented by truth-tables from TC_2^0 (i.e., does not provide a lossy encoding of significantly smaller size), or a new circuit lower bound follows. This result can be obtained as an easy application of Theorem 1.11, and its proof is presented in Section 5.1.

Next we observe that Proposition 1.6 ("derandomization yield circuit lower bounds") follows from the transference theorem for satisfiability. More precisely, Theorem 1.9 extends to slightly more general algorithms, an observation that we discuss in more detail in Section 3.1. Using this generalization, it is possible to prove that if Proposition 1.6 is false, then a contradiction can be obtained. This proof is presented in Section 5.2.

In the context of learning algorithms, some extensions of the main result of Fortnow and Klivans [FK09] for exact learning (Proposition 1.7) follow easily from results for useful properties (Theorems 1.11 and 1.12). In addition, it is not hard to show that even subexponential time randomized learning leads to useful properties decided by efficient randomized algorithms.

Theorem 1.15 ("Useful properties and learning algorithms").
Let $\mathcal{C} = \mathcal{C}[\mathsf{poly}]$ be a circuit class. If there exists a subexponential time randomized PAC *learning algorithm for $\mathcal{C}$, then there exists a* (promise)coRP*-property that is useful against $\mathcal{C}$.*

These transference theorems can be obtained by interpreting learning algorithms as lossy compression schemes, or by relying directly on Theorems 1.11 and 1.12. These results are discussed in more detail in Section 5.3.

Overall, these observations show that an improvement of a transference theorem in one framework leads to similar improvements in other frameworks. For instance, a proof of Conjecture 1.13 implies many interesting results of the form "nontrivial algorithms yield circuit lower bounds".

More precisely, it immediately implies new transference theorems for both (lossy) compression and satisfiability algorithms, and an alternative proof of the extension of Proposition 1.6 obtained by Aaronson and van Melkebeek [AvM11]. Moreover, a direct improvement of the transference theorems for satisfiability is likely to imply a similar strengthening of Proposition 1.6.

1.2.4 An overview of the results

For convenience of the reader, Figure 1 summarizes the relations between algorithms and circuit lower bounds discussed in this survey.
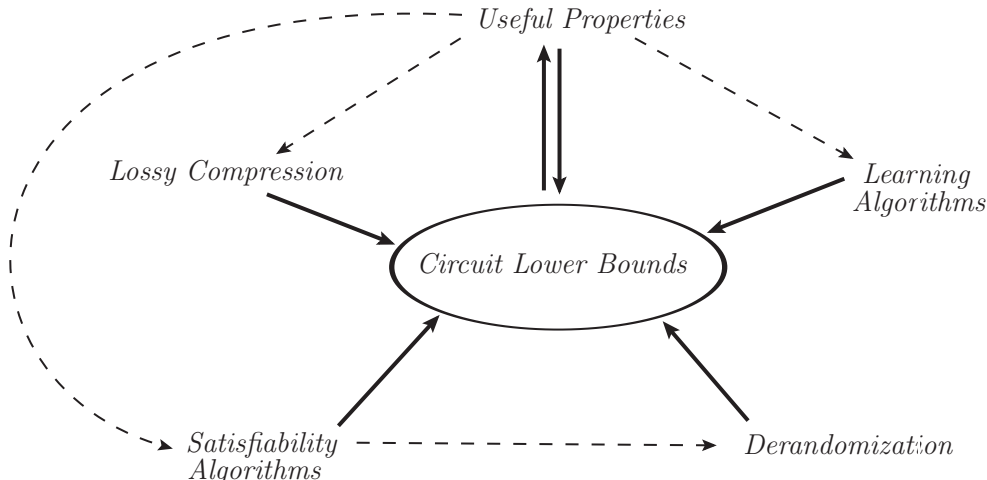

Figure 1: Bold arrows represent transference theorems, while a dotted arrow from A to B indicates that an improvement of the transference theorem for A implies a similar improvement of the transference theorem for B.

2 Preliminaries and Notation

We assume familiarity with basic notions from computational complexity theory. The reader is referred to Arora and Barak [AB09] and Goldreich [Gol08] for more details. For convenience, we postpone some definitions that are specific to a particular section of the paper to that corresponding section.

A boolean function is any function $h_n : \{0,1\}^n \to \{0,1\}$, for some fixed $n \in \mathbb{N}$. We say that h is a family of boolean functions if $h = \{h_n\}_{n \in \mathbb{N}}$. Any family of functions corresponds to a language $L \subseteq \{0,1\}^*$, and vice versa. For a language L, we use L_n to denote $L \cap \{0,1\}^n$.

We will use Γ to denote uniform complexity classes such as P, coRP and NP. Sometimes we will extend these complexity classes to the corresponding classes with advice of size $s(m)$, where m is the input size. In this case, we use $\Gamma/s(m)$. A language L is in i.o.Γ if there exists $L' \in \Gamma$ such that $L_n = L'_n$ for infinitely many values of n.

Following [Wil13b], we say that a circuit class $\mathcal{C}$ is *typical* if $\mathcal{C} \in \{\mathsf{AC}^0, \mathsf{ACC}, \mathsf{TC}^0, \mathsf{NC}^1, \mathsf{P/poly}\}$. The results stated for typical classes hold for more general circuit classes. We use $\mathsf{SIZE}[s(n)]$ to denote the family of functions computed by circuits of size $s(n)$. Similarly, $\mathsf{ASIZE}[s(n)]$ denotes the family of functions computed by arithmetic circuits of size $s(n)$. Although each circuit class

11

corresponds to a set of languages, we may abuse notation and say that a given circuit D is from $\mathcal{C}$. In general, for any circuit class $\mathcal{C}$, let $\mathcal{C}_d[s(n)]$ be the family of functions computed by circuits from $\mathcal{C}$ of depth d and size $s(n)$, where the size of a circuit is the number of gates in the circuit. If for convenience we omit $s(n)$, assume the circuits are of polynomial size. For instance, $\mathsf{TC}_2^0[n^2]$ corresponds to the class of languages computed by circuits of depth-two with $O(n^2)$ gates, each one corresponding to some linear threshold function. All circuit classes considered here are *non-uniform*. If we mention a circuit D of size $s(n)$ without attributing it to a specific circuit class, assume it is composed of AND, OR and NOT gates of fan-in at most two.

In order to prove a tight transference theorem for some circuit classes, we make the following definition.

Definition 2.1. *A circuit class $\mathcal{C}$ is* reasonable *if:*

(i) *The constant zero function $f : \{0,1\}^n \to \{0,1\}$ with $f(x) = 0$ for every input x is in $\mathcal{C}$.*

(ii) *For every function $g \in \mathcal{C}$, the function $\bar{g} = \mathsf{NOT}(g)$ is in $\mathcal{C}$, i.e., $\mathcal{C}$ is closed under complementation. In addition, there is an efficient algorithm that, given the description of a circuit computing g, outputs a circuit from $\mathcal{C}$ of the same size computing $\bar{g}$.*

(iii) *The gates of circuits from $\mathcal{C}$ may have direct access to constant inputs 0 and 1 in addition to the input variables and their negations[10].*

(iv) *Any language in $\mathcal{C}[\mathsf{poly}(n)]$ is in P/poly.*

The results that are stated for reasonable classes hold for more general circuit classes, but for simplicity we stick with this definition. In any case, most circuit classes are reasonable (in the sense of Definition 2.1), including AC^0, TC_2^0, NC^1, P/poly, etc.

We say that a deterministic algorithm is *nontrivial* if it runs in time $2^n/n^{\omega(1)}$. We may use this terminology to talk about nondeterministic and randomized algorithms with similar time bounds.

The following folklore result shows that to prove a circuit lower bound for P it is enough to obtain a circuit lower bound for the non-uniform class P/poly.

Lemma 2.2. *Let $\mathcal{C}_d[\mathsf{poly}(n)]$ be a reasonable circuit class. If $\mathsf{P} \subseteq \mathcal{C}_d[\mathsf{poly}(n)]$, then for every $b \in \mathbb{N}$ there exists a $t \in \mathbb{N}$ such that every boolean circuit over n inputs of size n^b admits an equivalent circuit from $\mathcal{C}_d$ of size n^t.*

Proof. Assume that $\mathsf{P} \subseteq \mathcal{C}_d[\mathsf{poly}(n)]$. Consider the following problem:

$$\mathsf{Circuit\text{-}Eval}_b = \{\langle E, x \rangle \ : \ E \text{ is a circuit on } n \text{ variables of size } \leq n^b \text{ and } E(x) = 1\}$$

Clearly, $\mathsf{Circuit\text{-}Eval}_b$ is in P (for any fixed b), and thus there exists t such that $\mathsf{Circuit\text{-}Eval}_b \in \mathcal{C}_d[n^t]$. In other words, there exists a sequence $\{D_n\}_{n \in \mathbb{N}}$ of circuits from $\mathcal{C}_d$ of size $O(n^t)$ that computes $\mathsf{Circuit\text{-}Eval}_b$.

Let $E_n : \{0,1\}^n \to \{0,1\}$ be a function over n boolean variables computed by a circuit of size at most n^b. We can hardwire the description of E_n inside circuit D_n (recall that $\mathcal{C}$ is reasonable, and that this operation does not increase the depth of the circuit). The resulting circuit is in $\mathcal{C}_d$, has size at most n^t, and it computes E_n by definition of D_n. $\qquad\square$

The next definition will play an important role in many results discussed later.

---

[10] This allows us to hardwire some values without increasing the depth of the circuit.

Definition 2.3 (Properties that are useful against $\mathcal{C}$ [Wil13b])**.** *A property of boolean functions is a subset of the set of all boolean functions. For a typical circuit class $\mathcal{C}$, a property $\mathcal{P}$ is said to be useful against $\mathcal{C}$ if, for all k, there are infinitely many positive integers n such that*

- *$\mathcal{P}(f_n)$ is true for at least one function $f_n : \{0,1\}^n \to \{0,1\}$, and*

- *$\mathcal{P}(g_n)$ is false for all functions $g_n : \{0,1\}^n \to \{0,1\}$ that admit circuits from $\mathcal{C}[n^k]$.*

We say that $\mathcal{P}$ is a Γ-property if, given the truth-table $tt(f_n) \in \{0,1\}^N$ (where $N = 2^n$) of any boolean function $f_n : \{0,1\}^n \to \{0,1\}$, $\mathcal{P}(f_n)$ can be decided in complexity class Γ. In other words, the language

$$L_{\mathcal{P}} = \{w \in \{0,1\}^N \mid w = tt(f_n) \text{ for some function } f_n : \{0,1\}^n \to \{0,1\} \text{ with } \mathcal{P}(f_n) = 1\}$$

is in Γ.

A useful property distinguishes some "hard" function from all easy ones. This is weaker than the notion of *natural properties* studied by [RR97], which also requires $\mathcal{P}$ to be dense, i.e., $\mathcal{P}(f) = 1$ for a non-negligible fraction of functions.

Recall that a verifier V for a language $L \in \mathsf{NTIME}[t(n)]$ satisfies the following properties:

- $V(x, w)$ runs in deterministic time $O(t(n))$, where $n = |x|$.

- $x \in L$ if and only if there exists $w \in \{0,1\}^{O(t(n))}$ such that $V(x, w) = 1$.

If $L \in \mathsf{NEXP}$ and V is a verifier for L running in time $2^{n^{O(1)}}$, we say that V is a NEXP-verifier for L. Similarly, we may talk about NE-verifiers running in time $2^{O(n)}$.

Definition 2.4. *Let $\mathcal{C}$ be a typical circuit class. We say that a NEXP-verifier V for a language $L \in \mathsf{NEXP}$ admits witness circuits from $\mathcal{C}[s(n)]$ if for all $x \in L$, there exists a circuit $C \in \mathcal{C}[s(n)]$ such that $V(x, tt(C)) = 1$.*

Proposition 2.5 (Impagliazzo et al. [IKW02], Williams [Wil11])**.** *Let $\mathcal{C}$ be a typical circuit class. If $\mathsf{NEXP} \subset \mathcal{C}$ then for any language $L \in \mathsf{NEXP}$ and every NEXP-verifier V for L, there exists $c \in \mathbb{N}$ such that V admits witness circuits from $\mathcal{C}[n^c]$.*

Definition 2.6. *Given functions $f, g : \{0,1\}^n \to \{0,1\}$ and $\delta > 0$, we say that g computes f with advantage δ if*

$$\Pr_{x \in_R \{0,1\}^n}[f(x) = g(x)] \geq \frac{1}{2} + \delta.$$

The results for learning algorithms and lossy compression rely on the following fact.

Lemma 2.7 ("Random functions are hard to approximate")**.**
There exists a constant $\alpha > 0$ such that for any sufficiently large n, there exists a function $h : \{0,1\}^n \to \{0,1\}$ that cannot be computed with advantage $\delta > 0$ by any circuit of size $\alpha \cdot 2^n \delta^2 / n$.

Proof. Fix any circuit $C : \{0,1\}^n \to \{0,1\}$. Using the Chernoff-Hoeffding bound, we get that the probability that C computes a random function $r : \{0,1\}^n \to \{0,1\}$ with advantage δ is at most $\exp(-2\delta^2 N)$, where $N = 2^n$ as usual. There are at most $2^{O(s(n) \log s(n))}$ functions on n inputs computed by circuits with $s(n)$ gates. Therefore, it follows by a simple union bound that for some $\alpha > 0$, there exists a function h that is not computed with advantage δ by any circuit of size $\alpha \cdot 2^n \delta^2 / n$. $\qquad\square$

3   Lower bounds from SAT algorithms for low depth circuits

In this section we present the transference theorem for satisfiability algorithms. We start with the following definition.

Definition 3.1. *Let $\mathcal{C}$ be a circuit class. We define the computational problem* Equiv-AND-$\mathcal{C}$ *as follows. Given the description of circuits from $\mathcal{C}$ computing functions $f_1, f_2, f_3 : \{0,1\}^n \rightarrow \{0,1\}$, check if* AND$(f_1, f_2)(x) = f_3(x)$ *for every $x \in \{0,1\}^n$. The* Equiv-OR-$\mathcal{C}$ *problem is defined analogously.*

Remark 1. *Observe that if $\mathcal{C}_d$ is reasonable, then an algorithm for* Equiv-AND-$\mathcal{C}_d$ *can be used to solve $\mathcal{C}_d$-SAT. Moreover, the same algorithm can be used to solve* Equiv-OR-$\mathcal{C}$, *since two functions are equivalent if and only if their negations are equivalent (by assumption, any reasonable circuit class is closed under negations).*

The proof presented here follows closely the original argument used by Williams [Wil10], which works for P/poly. However, we introduce a new technique that allows us to obtain an equivalent $\mathcal{C}$-circuit from a general P/poly-circuit. It simplifies the proof in [Wil11], and provides a tighter connection between satisfiability algorithms and circuit lower bounds in the case of bounded-depth circuits. The proof of the next lemma is partially inspired by some ideas in Rossman [Ros10].

Lemma 3.2 ("Conversion Lemma"). *Let $\mathcal{C}$ be a reasonable circuit class, and suppose that $\mathsf{P} \subseteq \mathcal{C}$. In addition, assume that there is a nontrivial algorithm for* Equiv-AND-$\mathcal{C}$. *Then there exists a nondeterministic algorithm $\mathcal{N}$ with the following properties. Given as input any circuit B over m variables of size m^b,*

- *$\mathcal{N}$ has at least one accepting path, and in every accepting path it outputs a circuit G from $\mathcal{C}[m^t]$ that is equivalent to B (where $t = O(b)$).*

- *$\mathcal{N}$ runs in time at most $\frac{2^m}{s(m)}$, for some superpolynomial function $s(m)$.*

Proof. Let $\mathcal{A}$ be an algorithm for Equiv-AND-$\mathcal{C}$ running in time $2^m/a(m)$, for a superpolynomial function $a(m)$. We proceed as follows. Let $x_1, x_2, \ldots, x_m, g_1, \ldots, g_k$ for $k = m^b$ be a topological sort of the gates of B, where each gate $g_i \in \{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ has fan-in at most two. We will guess and verify (by induction) equivalent $\mathcal{C}$-circuits of size m^t for each gate g_i in B. Since $\mathsf{P} \subseteq \mathcal{C}$, it follows from Lemma 2.2 that the functions computed at the internal gates of B admit such circuit.

More details follow. Suppose (by induction) that $\mathcal{N}$ has produced equivalent $\mathcal{C}$-circuits $B_i^{\mathcal{C}}$ of size at most m^t for every gate g_i of B, where $i < \ell$ (otherwise it has aborted already). If g_ℓ is an AND gate with inputs g_{i_1}, g_{i_2}, where $i_1, i_2 < \ell$, $\mathcal{N}$ guesses a circuit $B_\ell^{\mathcal{C}}$ in $\mathcal{C}[m^t]$ over the same input variables, then use the Equiv-AND-$\mathcal{C}$ algorithm to check if AND$(B_{i_1}^{\mathcal{C}}, B_{i_2}^{\mathcal{C}})$ and $B_\ell^{\mathcal{C}}$ are equivalent. $\mathcal{N}$ rejects if these circuits are not equivalent, otherwise it continues the computation, completing the induction step. If g_ℓ corresponds to an OR gate, a similar computation is performed, this time applying an algorithm for Equiv-OR-$\mathcal{C}$ (check Remark 1). Finally, if g_ℓ is a NOT gate, using the fact that $\mathcal{C}$ is reasonable, it is possible to produce in polynomial time an equivalent $\mathcal{C}$-circuit for g_ℓ of the same size. This completes the induction step. Observe that the base case is trivial.

Note that $\mathcal{N}$ runs algorithm $\mathcal{A}$ for at most k times, i.e., a polynomial number of times. In addition, each execution is performed over circuits from $\mathcal{C}$ of size $O(m^t)$. Therefore the total running time of $\mathcal{N}$ is $\mathsf{poly}(m) \cdot 2^m/a(m)$, for some superpolynomial function $a(m)$. Setting $s(m) = a(m)/\mathsf{poly}(m)$ completes the proof of Lemma 3.2. $\qquad\square$

14

In addition, we will need the following auxiliary results, whose notation we borrow from Williams [Wil11].

Definition 3.3. *The computational problem* Succinct-SAT *is defined as follows. Given a circuit C over n input variables, denote by F_C the instance of* 3-SAT *obtained by evaluating C over all inputs in lexicographic order (i.e., F_C is the 2^n-bit string representing the truth-table $tt(C)$ of C). Decide if F_C is satisfiable.*

We say that F_C is the *decompression* of C, and call C the *compression* of F_C.

Lemma 3.4 (Tourlakis [Tou01], Fortnow et a. [FLvMV05], Williams [Wil10])**.** *There is a fixed constant $c > 0$ for which the following holds. For every $L \in$ NTIME$[2^n]$ there is a polynomial time reduction from L to* Succinct-SAT *that maps every input x of size n to a circuit C_x over at most $n + c \log n$ input variables and size $O(n^c)$, such that $x \in L$ if and only if the decompressed formula F_{C_x} is satisfiable (observe that this is a formula of size $2^n \mathsf{poly}(n)$).*

Definition 3.5. *We say that* Succinct-SAT *admits succinct satisfying assignments if there exists a constant $c > 0$ such that for every language $L \in$ NTIME$[2^n]$ the following holds. Given any $x \in L$, there exists some circuit W_x of polynomial size over $k \leq n + c \log n$ input variables for which the assignment $z_i = W(i)$ for $i \in \{1, \ldots, 2^k\}$ is a satisfying assignment for F_{C_x}, where C_x is the circuit obtained from the reduction to* Succinct-SAT *given by Lemma 3.4.*

The following lemma is an easy consequence of Proposition 2.5.

Lemma 3.6. *If* NEXP $\subseteq$ P/poly *then* Succinct-SAT *admits succinct satisfying assignments.*

We use these auxiliary results to prove the following proposition. For simplicity, we only state it for polynomial size classes, but a parameterized version can be obtained using the same techniques.

Proposition 3.7. *Let $\mathcal{C} = \mathcal{C}_d[\mathsf{poly}(n)]$ be a reasonable circuit class. If there exist a nontrivial algorithm for* Equiv-AND-$\mathcal{C}$, *then* NEXP $\nsubseteq \mathcal{C}$.

Proof. Let $\mathcal{A}$ be a nontrivial algorithm for Equiv-AND-$\mathcal{C}$, and assume for the sake of a contradiction that NEXP $\subseteq \mathcal{C}$. We use these assumptions to show that every language $L \in$ NTIME$[2^n]$ is in NTIME$[o(2^n)]$, a contradiction to the nondeterministic time hierarchy theorem ([Coo73], [SFM78], [Žák83]).

The proof relies on the fact that every language $L \in$ NTIME$[2^n]$ can be efficiently reduced to an instance of the Succinct-SAT problem (Lemma 3.4). In other words, there is a polynomial time algorithm that maps any input $x \in \{0,1\}^n$ to a circuit D_x on $n + c \log n$ input variables and at most $O(n^c)$ gates such that $x \in L$ if and only if the decompression $F_x = tt(D_x)$ of D_x is satisfiable.

It follows from NEXP $\subseteq \mathcal{C} \subseteq$ P/poly ($\mathcal{C}$ is reasonable) and Lemma 3.6 that if F_x is satisfiable then there is a satisfying assignment encoded by a circuit E_x of polynomial size over $n + O(\log n)$ variables. Summarizing what we have so far:

$$x \in L \iff \exists \text{ circuit } E : \{0,1\}^{n+O(\log n)} \to \{0,1\} \text{ of size } O(n^d) \text{ such that } F_x(tt(E)) = 1,$$

where $F_x = tt(D_x)$ is a 3-CNF formula and $D_x : \{0,1\}^{n+O(\log n)} \to \{0,1\}$ is an arbitrary circuit (not necessarily in $\mathcal{C}$) of size $O(n^c)$ encoding this formula.

Our nontrivial algorithm for L now guesses a candidate circuit E of this form. It uses D_x and three copies of E to build a circuit $B = B(D_x, E)$ of size $O(n^b)$ over $n + O(\log n)$ inputs such that:

B is satisfiable $\quad\Longleftrightarrow\quad$ some clause C_i of F_x is not satisfiable by the assignment $tt(E)$.

The description of B is as follows. An input y to B is interpreted as an integer i, and B uses this index to obtain from D_x the description of the i-th clause C_i in F_x. Let z_1, z_2, z_3 be the literals in C_i. Circuit B uses three copies of E to obtain the boolean values of the variables corresponding to these literals, and finally outputs 1 if and only if these values do not satisfy C_i. This last verification can be done by a polynomial size circuit. Observe that B is not a $\mathcal{C}$-circuit: D_x and E are arbitrary circuits, these circuits are composed, and there is additional circuitry computing the final output value of B. Overall, we obtain:

$$x \in L \quad\Longleftrightarrow\quad \text{circuit } B : \{0,1\}^{n+O(\log n)} \to \{0,1\} \text{ is unsatisfiable.} \tag{1}$$

Note that all these steps can be performed in $\mathsf{NTIME}[\mathsf{poly}(n)]$. Recall that we can use Equiv-AND-$\mathcal{C}$ to solve $\mathcal{C}$-SAT in less than 2^n steps, but B is not a circuit from $\mathcal{C}$. We can assume without loss of generality that B is a circuit of size m^b (where $m = n + O(\log n)$) consisting of AND, OR and NOT gates of fan-in at most two.

While in Williams' original proof there is a step that guesses and verifies an equivalent $\mathcal{C}$-circuit for D_x (and already assumes E in $\mathcal{C}[\mathsf{poly}(n)]$ with some extra work), our nondeterministic algorithm for L produces directly an equivalent $\mathcal{C}$-circuit for the final circuit B. Under our assumptions, Lemma 3.2 can be applied, and it allows the nondeterministic algorithm for L to obtain a circuit G over m inputs from $\mathcal{C}[m^t]$ that is equivalent to B. This step can be performed in time $2^m/s(m)$ for a superpolynomial function $s(m)$. Since $m = n + O(\log n)$, this running time is still nontrivial in n.

Using condition (1), it follows that $x \in L$ if and only if G is unsatisfiable. Finally, since $\mathcal{C}$ is reasonable, we can use algorithm $\mathcal{A}$ to check if this is true, in which case our algorithm for L accepts input x. Again, this is a computation that can be performed in nontrivial running time by our assumption over $\mathcal{A}$. Overall, it follows that we can decide L in $\mathsf{NTIME}[o(2^n)]$, which completes the proof of the theorem. $\qquad\square$

Corollary 3.8. *Let $\mathcal{C} = \mathcal{C}_d[\mathsf{poly}(n)]$ be a reasonable circuit class. If there exist nontrivial satisfiability algorithms for both $\mathsf{AND}[3] \circ \mathcal{C}$ and $\mathsf{AND}[2] \circ \mathsf{OR}[2] \circ \mathcal{C}$, then $\mathsf{NEXP} \not\subseteq \mathcal{C}$.*

Proof. It is enough to observe that these satisfiability algorithms can be used to solve Equiv-AND-$\mathcal{C}$ in nontrivial running time (Proposition 3.7). Let f_1, f_2, f_3 be functions from $\mathcal{C}$. Then

$$\neg\mathsf{EQUIV}(\mathsf{AND}(f_1, f_2), f_3) \quad\Longleftrightarrow\quad \mathsf{XOR}(\mathsf{AND}(f_1, f_2), f_3) \text{ is satisfiable.} \tag{2}$$

For bits $a, b \in \{0, 1\}$, we have $\mathsf{XOR}(a, b) \equiv \mathsf{OR}(\mathsf{AND}(a, \bar{b}), \mathsf{AND}(\bar{a}, b))$. Using de Morgan's rules and combining gates, it is not hard to see that

$$\mathsf{XOR}(\mathsf{AND}(f_1, f_2), f_3) \equiv \mathsf{OR}(\mathsf{AND}(f_1, f_2, \bar{f}_3), \mathsf{AND}(\mathsf{OR}(\bar{f}_1, \bar{f}_2), f_3)).$$

It follows from (2) that an algorithm for $\neg$Equiv-AND-$\mathcal{C}$ should output 1 if and only if either $\mathsf{AND}(f_1, f_2, \bar{f}_3)$ or $\mathsf{AND}(\mathsf{OR}(\bar{f}_1, \bar{f}_2), f_3)$ is satisfiable. Since $\mathcal{C}$ is reasonable, a circuit for $\bar{f}_i$ can be computed efficiently from a circuit for f_i. Hence nontrivial algorithms for $\mathsf{AND}[3] \circ \mathcal{C}$-SAT and $\mathsf{AND}[2] \circ \mathsf{OR}[2] \circ \mathcal{C}$-SAT can be used to solve $\neg$Equiv-AND-$\mathcal{C}$, which completes the proof. $\qquad\square$

3.1 A remark for the algorithm designer

It is hard to find satisfiability algorithm for expressive circuit classes even when we allow very modest running times, such as $2^n/n^{\log n}$. Here we mention a weaker assumption on the algorithmic side that could be of practical significance[11].

Definition 3.9 ("Algorithms useful for circuit lower bounds").
Let $\mathcal{C}$ be a circuit class. A nondeterministic algorithm $\mathcal{A}$ for Equiv-AND-$\mathcal{C}$ *is* useful *if the following conditions hold:*

- *Every path of the (nondeterministic) computation of $\mathcal{A}$ either outputs "abort", or provides the correct answer.*

- *At least one path of the computation of $\mathcal{A}$ does not abort, and runs in time bounded by $2^n/s(n)$ for some superpolynomial function $s(n)$.*

Proposition 3.10. *Let $\mathcal{C} = \mathcal{C}_d[\mathsf{poly}(n)]$ be a reasonable circuit class. If there exists a useful algorithm for* Equiv-AND-$\mathcal{C}$ *then* NEXP $\nsubseteq \mathcal{C}$.

Proof. Observe that the proof of Proposition 3.7 still holds with such algorithms, provided that we abort in any computation path that runs for more than $2^n/s(n)$ steps. It is still the case that $x \in L$ if and only if there exists a computation path that accepts x. More precisely, if $x \notin L$, even if equivalent circuits are guessed and verified in each stage, a useful algorithm for Equiv-AND-$\mathcal{C}$ will never output "yes" in the last step of the computation that checks if the final circuits is equivalent to the zero function (i.e., it is unsatisfiable). On the other hand, for $x \in L$, it is clear from the definition of useful algorithm that some computation path will accept in nontrivial running time. $\square$

Observe that useful algorithms for *unsatisfiability* also lead to circuit lower bounds, since these can be used in place of an Equiv-AND-$\mathcal{C}$ algorithm. The same is true for satisfiability algorithms, since useful algorithms are closed under complementation. In Section 5.2 we will use Proposition 3.10 to prove that derandomization implies circuit lower bounds (Proposition 1.6).

Why is this a natural relaxation? Suppose there exists a class $\mathcal{C}$ such that for any circuit D in this class, there exists some subset $S \subset [n]$ of the inputs of D such that by trying all assignments to the variables in S, we can check on *average time* strictly less than $2^{n-|S|}$ (over the restrictions) the satifiability of the remaining circuits. Then $\mathcal{C}$ admits a useful satisfiability algorithm, since the set S can be guessed at the beginning of the execution. For the reader familiar with the satisfiability algorithm for small threshold circuits described by Impagliazzo, Paturi and Schneider [IPS12], it means that their algorithm gives more than what is needed for lower bounds. There the expected running time is nontrivial over the subset of inputs to be restricted, which is a stronger guarantee. It is sufficient that a single subset provides a nontrivial running time.

4 Useful properties and circuit lower bounds

In this section we focus on the relation between *useful properties* and circuit lower bounds, a connection that was made explicit in a recent paper written by Williams [Wil13b]. We start with

---

[11]This is not the most encompassing definition, but it is a very natural one to have in mind.

the following simple, but somewhat surprising result. Recall that an algorithm that computes a property of boolean functions receives as input a string of size $N = 2^n$ representing the truth-table $tt(f)$ of a function $f : \{0,1\}^n \to \{0,1\}$. The following result follows from techniques introduced by Williams [Wil13b].

Proposition 4.1 ("Useful NP-property yields useful P-property").
Let $\mathcal{C}$ be a typical circuit class, and let $s : \mathbb{N} \to \mathbb{N}$ be any function. If there is a $\mathsf{NP}/s(N)$-property useful against $\mathcal{C}$ then there is a $\mathsf{P}/s(N)$-property useful against $\mathcal{C}$.

Proof. First we prove the proposition without advice, then we observe that the same proof works in the presence of advice strings as well. Let $\mathcal{P}$ be a NP-useful property against $\mathcal{C}$. In other words, for any fixed k, there exists an infinite subset $S_k \subseteq \mathbb{N}$ such that for any $n \in S_k$:

- $\mathcal{P}(f_n) = 1$ for at least one function $f_n : \{0,1\}^n \to \{0,1\}$.

- $\mathcal{P}(g_n) = 0$ for any function $g_n : \{0,1\}^n \to \{0,1\}$ computed by circuits in $\mathcal{C}[n^k]$.

In addition, there exists a polynomial time verifier $V_{\mathcal{P}} : \{0,1\}^N \times \{0,1\}^{N^c - N} \to \{0,1\}$ (where $N = 2^n$ and $c \in \mathbb{N}$) for $L_{\mathcal{P}}$. Put another way, for any function h_n,

$$\mathcal{P}(h_n) = 1 \quad \Longleftrightarrow \quad \exists w \in \{0,1\}^{N^c - N} \text{ such that } V_{\mathcal{P}}(tt(h_n), w) = 1.$$

Let $A = \{n' \mid n' = cn, n \in \mathbb{N}\}$. For convenience, set $N' = 2^{n'} = N + (N^c - N)$. We define a predicate $\mathcal{P}'$ defined on any function over n' inputs, where $n' \in A$ (the definition of $\mathcal{P}'$ over functions with a different number of inputs can be arbitrary). For any $h'_{n'} : \{0,1\}^{n'} \to \{0,1\}$, view its representation $tt(h'_{n'}) \in \{0,1\}^{N'}$ as a pair of strings $(tt(h_n), w)$, where $tt(h_n) \in \{0,1\}^N$ and $w \in \{0,1\}^{N^c - N}$. To be more precise, let $h_n : \{0,1\}^n \to \{0,1\}$ be the restriction of $h'_{n'}$ defined by $h_n(x) = h'_{n'}(x0^{(c-1)n})$, where $x \in \{0,1\}^n$. Finally, let

$$\mathcal{P}'(h'_{n'}) = 1 \quad \Longleftrightarrow \quad V_{\mathcal{P}}(tt(h_n), w) = 1.$$

We claim that $\mathcal{P}'$ is a P-property that is useful against $\mathcal{C}$. First observe that since $V_{\mathcal{P}}$ is an efficient algorithm, $\mathcal{P}'$ can be computed in time polynomial in $N' = |tt(h'_{n'})|$. Fix any $k \in \mathbb{N}$. We need to define an infinite set $S'_k \subseteq A$ such that for every $n' \in S'_k$,

- $\mathcal{P}(f'_{n'}) = 1$ for at least one function $f'_{n'} : \{0,1\}^{n'} \to \{0,1\}$.

- $\mathcal{P}(g'_{n'}) = 0$ for any function $g'_{n'} : \{0,1\}^{n'} \to \{0,1\}$ computed by circuits in $\mathcal{C}[n'^k]$.

Let $S'_k = \{n' \mid n' = cn, n \in S_{k+1}\}$. This set is infinite because so is S_{k+1}. Let $n' \in S'_k$. It follows from the definition of S_{k+1} that there is a function $f_n : \{0,1\}^n \to \{0,1\}$ for which $\mathcal{P}(f_n) = 1$. Hence there exists $w \in \{0,1\}^{N^c - N}$ such that $V_{\mathcal{P}}(tt(f_n), w) = 1$. By construction, the corresponding function $f'_{n'} : \{0,1\}^{n'} \to \{0,1\}$ whose truth-table is the concatenation of the pair (f_n, w) satisfies $\mathcal{P}'$.

Finally, in order to establish the second bullet, assume for the sake of a contradiction that there exists a function $g'_{n'} : \{0,1\}^{n'} \to \{0,1\}$ computed by circuits from $\mathcal{C}[n'^k]$ for which $\mathcal{P}'(g'_{n'}) = 1$. Clearly, the function $g_n : \{0,1\}^n \to \{0,1\}$ defined as before by the restriction $g_n(x) = g'_{n'}(x0^{(c-1)n})$ also admits circuits from $\mathcal{C}$ of size $n'^k = (cn)^k \leq n^{k+1}$, for sufficiently large values of n. But then $\mathcal{P}(g_n) = 0$, since $n \in S_{k+1}$. However, this contradicts the assumption that $\mathcal{P}'(g'_{n'}) = 1$, since in this

case there is no $w \in \{0,1\}^{N^c - N}$ such that $V_{\mathcal{P}}(tt(g_n), w) = 1$. In other words, for every function $g'_{n'}$ with $n' \in S'_k$ that is computed by circuits from $\mathcal{C}[n'^k]$, we have $\mathcal{P}'(g'_{n'}) = 0$.

If the original verifier works with advice strings of length $s(N)$, then property $\mathcal{P}'$ can be decided correctly using the same advice. However, the definition of $\mathcal{P}'$ over functions on $n' = cn$ inputs is based on the definition of $\mathcal{P}$ over functions on n inputs. Therefore, the advice for the new algorithm is of size $s(N'^{1/c})$, since it gets as input truth-tables of size $N' = N^c$. Assuming that $s(.)$ is non-decreasing and $c \geq 1$, it follows that $\mathcal{P}'$ can be decided with advice of size $s(N'^{1/c}) \leq s(N')$. This completes the proof of Proposition 4.1. $\qquad\square$

The new useful property may not be dense, even if the original property is dense. The reason is that there may be just a few certificates for each hard function, thus almost no function will satisfy the newly defined property. However, if we start with an RP-natural property useful against $\mathcal{C}$ (i.e., a dense property in which every hard function has many certificates), the proof of Proposition 4.1 yields a corresponding P-natural property.

The next proposition clarifies the relation between NEXP circuit lower bounds and the existence of properties that are useful against $\mathcal{C}$. Recall that for any typical circuit class, standard arguments can be used to prove that NEXP $\not\subseteq \mathcal{C}$ if and only if NE $\not\subseteq \mathcal{C}$.

Proposition 4.2. *Let $\mathcal{C}$ be a typical class. Then* NEXP $\not\subseteq \mathcal{C}$ *if and only if there exists a* P$/\log N$-*property that is useful against $\mathcal{C}$.*

Proof. Let $N = 2^n$ as usual. First assume that NEXP $\not\subseteq \mathcal{C}$, and let $L \in$ NE$\backslash \mathcal{C}$. Let $L' = L \cup \{1^n \mid n \in \mathbb{N}\}$, and notice that $L' \in$ NE$\backslash \mathcal{C}$. For every $n \in \mathbb{N}$, let $b(n)$ be the number of strings of size n in L'. Observe that $b(n) \in [1, 2^n]$. Therefore $b(n)$ can be encoded by a string $a(n) \in \{0,1\}^{\log N}$. Let $f_n = L_n$, i.e., $f_n(x) = 1$ if and only if $x \in L$. Consider the property $\mathcal{P}$ such that $\mathcal{P}(g) = 1$ if and only if $g = f_n$ for some $n \in \mathbb{N}$. We claim that $\mathcal{P}$ is a NP$/\log N$-property that is useful against $\mathcal{C}$. Let V' be an NE-verifier for L' accepting witnesses of size 2^{cn}.

Clearly, $\mathcal{P}$ is useful against $\mathcal{C}$, because $L' \notin \mathcal{C}$. On the other hand, the following NP-verifier decides $\mathcal{P}$ when it is given the correct advice string $a(n)$:

Verifier V for $\mathcal{P}$:
On inputs $tt(h) \in \{0,1\}^N$ and advice string $z \in \{0,1\}^{\log N}$, reject if $|h^{-1}(1)| \neq z$. Otherwise, guess witnesses $w_x \in \{0,1\}^{N^c}$ for every $x \in h^{-1}(1)$, and accept if and only if $V'(x, w_x) = 1$ for every such x.

Clearly, when $z = a(n)$, the only function over n inputs accepted by V is $f_n = L_n$. In addition, V runs in time $\mathsf{poly}(N)$. It follows that $\mathcal{P}$ is computed in NP$/\log N$. Therefore, there is a NP$/\log N$-property that is useful against $\mathcal{C}$, and Proposition 4.1 guarantees the existence of a P$/\log N$-property useful against $\mathcal{C}$.

Now suppose that there exists a P$/\log N$-property $\mathcal{P}'$ that is useful against $\mathcal{C}$. We use this assumption to define a NEXP-verifier V' that does not admit witness circuits of polynomial size. Observe that it follows then from Proposition 2.5 that NEXP $\not\subseteq \mathcal{C}$, which completes the proof our result.

Let $\mathcal{A}'$ be an algorithm running in time N^d that decides $\mathcal{P}'$ on inputs $tt(f) \in \{0,1\}^N$ when it is given access to an appropriate advice string $a(N) \in \{0,1\}^{\log N}$, i.e, a string of size n. Consider the following verifier.

NEXP-verifier V':
On input $\langle x, w \rangle$, where $x \in \{0,1\}^n$ and $w \in \{0,1\}^N$, output $\mathcal{A}'(w)/x$ (i.e., run $\mathcal{A}'$ on input w with advice string x).

First observe that V' is a NEXP-verifier. Fix any $c \in \mathbb{N}$. We prove that V' does not admit witness circuits from $\mathcal{C}[n^c]$. First, there are infinitely many inputs n for which $\mathcal{P}'$ correctly discriminates a hard function $h_n : \{0,1\}^n \to \{0,1\}$ from a function in $\mathcal{C}[n^c]$. For any such value of n, there is a correct advice string $a(N)$ for which algorithm $\mathcal{A}'$ computes $\mathcal{P}'$. However, whenever $x = a(N)$, it follows from the definition of V' that it only accepts certificates for x that do not correspond to any truth-table from $\mathcal{C}[n^c]$. In addition, V' accepts at least one truth-table, by definition of $\mathcal{P}'$. As discussed before, this completes the proof of Proposition 4.2. $\qquad\square$

One may be tempted to pose the following conjecture.

Conjecture 4.3. *Let $\mathcal{C}$ be a typical circuit class. If there exists a $\mathsf{P}/O(\log N)$-property that is useful against $\mathcal{C}$, then there is a P-property that is useful against $\mathcal{C}$.*

We will see shortly that if a slightly more general version of this conjecture holds, then a generic NEXP circuit lower bound can always be converted into a $\mathsf{NE} \cap \mathsf{coNE}$ lower bound, a rather surprising consequence, given its generality.

Now we move to the relation between useful properties decided without advice and circuit lower bounds.

Proposition 4.4. *For any typical $\mathcal{C}$, if $\mathsf{NE} \cap \mathsf{coNE} \nsubseteq \mathcal{C}$ then there exists a P-property that is useful against $\mathcal{C}$.*

Proof. Let $L \in \mathsf{NE} \cap \mathsf{coNE} \setminus \mathcal{C}$, and let V^0 and V^1 be verifiers running in time $2^{O(n)}$ for $n = |x|$ such that:

$$x \in L \quad \Longleftrightarrow \quad \exists w_x \in \{0,1\}^{2^{O(n)}} \text{ such that } V^1(x, w_x) = 1.$$
$$x \notin L \quad \Longleftrightarrow \quad \exists w_x \in \{0,1\}^{2^{O(n)}} \text{ such that } V^0(x, w_x) = 1.$$

We view L as a family of functions $f = \{f_n\}_{n \in \mathbb{N}}$, where $f_n^{-1}(1) = L_n$. Let $\mathcal{P} = \{f_n \mid n \in \mathbb{N}\}$. First observe that this property is useful against $\mathcal{C}$, since $L \notin \mathcal{C}$. In addition, there is an efficient verifier $V_\mathcal{P}$ for $\mathcal{P}$: on input a string $tt(h) \in \{0,1\}^N$ representing the truth-table of a function $h : \{0,1\}^n \to \{0,1\}$, guess 2^n certificates $y_x \in \{0,1\}^{N^c}$, one for each $x \in \{0,1\}^n$, and accept if and only if $V^{h(x)}(x, y_x) = 1$ for every such x. Clearly, $V_\mathcal{P}$ is a NP-verifier for $\mathcal{P}$. It follows then from Proposition 4.1 that there exists a P-property $\mathcal{P}'$ that is useful against $\mathcal{C}$, which completes the proof. $\qquad\square$

Conversely, which consequences can we obtain from the existence of P-properties (without advice) that are useful against $\mathcal{C}$? The following result is implicit in the work of Williams [Wil13b], and it shows that without advice even stronger consequences can be obtained (although in the quasipolynomial size regime).

Proposition 4.5. *Let $\mathcal{C}$ be a typical circuit class. If for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \nsubseteq \mathcal{C}[n^{\log n}]$.*

We give a self-contained proof of this result in Appendix A.

Proposition 4.5 sheds some light into Conjecture 4.3. It shows that if the analogue of this conjecture for quasipolynomial size circuits holds, then NEXP lower bounds against such circuits can be translated into similar NE ∩ coNE circuit lower bounds (via a generalization of Proposition 4.2 to quasipolynomial size circuits).

Given the statement of Propositions 4.4 and 4.5, it is plausible to conjecture that there is a tight correspondence between useful properties computed without advice and circuit lower bounds for NE ∩ coNE.

Conjecture 4.6. *Let $\mathcal{C} = \mathcal{C}[\mathsf{poly}]$ be a typical circuit class. Then $\mathsf{NE} \cap \mathsf{coNE} \nsubseteq \mathcal{C}$ if and only if there exists a P-property that is useful against $\mathcal{C}$.*

We will see in Section 5 that useful properties are powerful enough to simplify and generalize many results of the form "nontrivial algorithms yield circuit lower bounds". In particular, a proof of Conjecture 4.6 would provide stronger transference theorems in different frameworks.

4.1 Satisfiability algorithms and useful properties

It is possible to formulate the main result from Section 3 as follows: the existence of nontrivial satisfiability algorithms leads to useful properties, which in turn imply circuit lower bounds. This can be accomplished using the fact that the nondeterministic hierarchy theorem also holds for unary languages. In other words, if there exists a nontrivial SAT algorithm for a circuit class $\mathcal{C}$, the proof of Proposition 3.7 shows that any verifier for a hard unary language must have infinitely many inputs that only admit certificates of high $\mathcal{C}$-circuit complexity. This verifier can be used to define a property that is useful against $\mathcal{C}$: given a truth table $tt(h_n)$, check if it is a valid certificate for the input 1^n.

More specifically, satisfiability algorithms for polynomial size circuits lead to P-properties useful against circuits of polynomial size, while algorithms for quasipolynomial size circuits lead to P-properties useful against circuits of such size. The reader should compare the transference theorems from [Wil11] and [Wil13b] (Propositions 1.1 and 1.3, respectively) to the statements of Propositions 4.2 and 4.5. If Conjecture 4.6 is true, the existence of nontrivial satisfiability algorithms for $\mathcal{C}[\mathsf{poly}]$ would imply that $\mathsf{NE} \cap \mathsf{coNE} \nsubseteq \mathcal{C}[\mathsf{poly}]$, a new result.

5 Applications

5.1 Lower bounds from lossy compression

In this section we prove the transference theorem obtained by Chen et al. [CKK+13], which we state again for convenience.

Proposition 5.1 (Compression yields circuit lower bounds [CKK+13])**.**
Let $\mathcal{C}$ be a typical circuit class. Suppose that for every $c \in \mathbb{N}$ there is a deterministic polynomial-time algorithm that compresses a given truth table of an n-variate boolean function $f \in \mathcal{C}[n^c]$ to an equivalent circuit of size $o(2^n/n)$. Then $\mathsf{NEXP} \nsubseteq \mathcal{C}$.

As mentioned before, it is possible to show a similar result from the existence of lossy compression algorithms.

Definition 5.2 (Lossy compression scheme). *Let $\mathcal{C}$ be a typical circuit class. We say that a deterministic algorithm $\mathcal{A}$ is a $(\delta(n), s(n))$-compression algorithm for $\mathcal{C}$ if $\mathcal{A}$ runs in time $\mathsf{poly}(N)$, and for any fixed $k \in \mathbb{N}$, there are infinitely many integers n for which the following holds. Given any string $tt(f_n) \in \{0,1\}^N$ representing a function $f_n : \{0,1\}^n \to \{0,1\}$ computed by circuits in $\mathcal{C}[n^k]$, $\mathcal{A}$ outputs a circuit C on n inputs of size at most $s(n)$ that computes f_n with advantage $\delta(n)$.*

Proposition 5.3 (Lossy compression yields circuit lower bounds).
Let $\mathcal{C}$ be a typical circuit class, and let $\delta(n) : \mathbb{N} \to (0, 1/2]$ be an arbitrary function. If there exists a $(\delta(n), o(2^n \delta^2/n))$-compression algorithm for $\mathcal{C}$, then $\mathsf{NEXP} \nsubseteq \mathcal{C}$.

Proof. Let $\mathcal{C}$ be a typical circuit class. Fix any function $\delta = \delta(n)$. Let $\mathcal{A}$ be an efficient $(\delta, o(2^n \delta^2/n))$-compression algorithm for $\mathcal{C}$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that implicitly defines a property that is useful against $\mathcal{C}$. The proof then follows immediately from Proposition 4.2.

We define $\mathcal{B}$ as follows. Given any truth table $tt(f) \in \{0,1\}^N$ as input, apply $\mathcal{A}$ to $tt(f)$ to obtain the description of a circuit C over n inputs. If C is not a valid circuit, or it has more than $\alpha \cdot 2^n \delta^2/n$ gates, accept. Otherwise, check if C computes f with advantage δ, and accepts $tt(f)$ if and only if this is *not* the case.

Let $\mathcal{P}$ be the property computed by $\mathcal{B}$. We need to check that $\mathcal{P}$ is a P-property that is useful against $\mathcal{C}$. First, observe that $\mathcal{B}$ runs in time $\mathsf{poly}(N)$, since by assumption $\mathcal{A}$ is efficient, and $N = 2^n$. Also, B will always accept some family of hard functions, since it follows from Lemma 2.7 that for sufficiently large n there are functions that cannot be computed with advantage δ by circuits of size less than $\alpha \cdot 2^n \delta^2/n$. Finally, for any fixed k, it follows from the definition of lossy compression that there are infinitely many input sizes n on which $\mathcal{A}$ succeeds. For all such inputs sizes, algorithm $\mathcal{B}$ will correctly reject functions computed by circuits from $\mathcal{C}[n^k]$. $\qquad\square$

This result is optimal for very small δ. More precisely, it follows from elementary Fourier analysis of boolean functions that for every boolean function f_n there is a parity function over some subset $S \subseteq [n]$ that computes f_n with advantage $\Omega(2^{-n/2})$. Further, it is possible to check all parity functions in deterministic time $\mathsf{poly}(N)$.

Remark 2. *Similar techniques can be used to show that lossy compression of quasipolynomial size circuits leads to circuit lower bounds for $\mathsf{NE} \cap \mathsf{i.o.coNE}$. This can be obtained through an application of Proposition 4.5.*

5.2 Derandomization, SAT algorithms and circuit lower bounds

In this section we use Williams' framework to prove that derandomization yields circuit lower bounds. Recall that PIT is the language consisting of all arithmetic circuits that compute the zero polynomial over $\mathbb{Z}$, and PERM is the problem of computing the permanent of integer matrices.

Our proof uses the notion of useful algorithms introduced in Definition 3.9. The following consequence is immediate from Proposition 3.10.

Corollary 5.4. *Assume that $\mathsf{NEXP} \subseteq \mathsf{SIZE}[\mathsf{poly}]$. Then there is $c \in \mathbb{N}$ such that there is no useful algorithm for $\mathsf{Equiv\text{-}AND\text{-}SIZE}[n^c]$.*

In addition, we will need the following auxiliary lemma.

Lemma 5.5 (Kabanets and Impagliazzo [KI04], Aaronson and van Melkebeek [AvM11])**.**
There exists an efficient algorithm that takes as input an arithmetic circuit A_m and an integer m, and produces an arithmetic circuit C_m such that A_m computes the permanent of $m \times m$ matrices matrices over $\mathbb{Z}$ if and only if $C_m \in$ PIT.

We are now ready to give a short proof of the following result. Our argument follows the same high-level approach employed by [KI04] and [AvM11].

Proposition 5.6 (Kabanets and Impagliazzo [KI04])**.**
If PIT $\in$ NSUBEXP, then at least one of the following results hold:

(*i*) NEXP $\not\subseteq$ SIZE[poly(n)]*; or*

(*ii*) PERM $\not\subseteq$ ASIZE[poly(n)]*.*

Proof. In order to derive a contradiction, assume that:

- PIT $\in$ NSUBEXP;

- NEXP $\subseteq$ SIZE[poly(n)];

- PERM $\subseteq$ ASIZE[poly(n)].

More precisely, NEXP $\subseteq$ SIZE[poly(n)] implies that there exists a family of circuits $D = \{D_n\}_{n \in \mathbb{N}}$ of size n^d that solves Equiv-AND-SIZE[n^c]. In addition, PERM over matrices of order m can be solved by a family of arithmetic circuits $A = \{A_m\}_{m \in \mathbb{N}}$ of size m^a (for some $a \in \mathbb{N}$). We prove that these assumptions contradict Corollary 5.4. We construct a useful algorithm $\mathcal{A}$ for Equiv-AND-SIZE[n^c] as follows.

Algorithm $\mathcal{A}$:
Input: Circuits C_1, C_2 of size n^c.

- First, $\mathcal{A}$ guesses a circuit D_n of size n^d.

- $\mathcal{A}$ prepares a query to the polynomial time hierarchy[12] to check if D_n solves Equiv-AND-SIZE[n^c].

- It uses Toda's theorem [Tod91] together with the completeness of the permanent problem [Val79] to reduce this query to a call to PERM over matrices of dimension m, where $m = $ poly(n^d).

- Next, $\mathcal{A}$ guesses an arithmetic circuit A_m of size m^a.

- It then applies Lemma 5.5 to obtain a circuit C_m such that A_m computes the permanent of $m \times m$ matrices matrices over $\mathbb{Z}$ if and only if $C_m \in$ PIT.

- Now $\mathcal{A}$ uses nondeterminism and the assumption that PIT $\in$ NSUBEXP to check if $C_m \in$ PIT. It aborts otherwise.

---

[12]Observe that D_n does not solve the equivalence problem if and only if ($\exists C_1, C_2\ \exists x$ such that $C_1(x) \neq C_2(x)$ and $D_n(C_1, C_2) = 1$) or ($\exists C_1, C_2$ such that $\forall x(C_1(x) = C_2(x))$ and $D_n(C_1, C_2) = 0$).

- It uses A_m to answer the initial query, and aborts if D_n does not solve Equiv-AND-SIZE$[n^c]$.

- Finally, $\mathcal{A}$ uses D_n to solve Equiv-AND-SIZE$[n^c]$ on inputs C_1 and C_2.

Clearly, $\mathcal{A}$ runs in nondeterministic subexponential time. In addition, it is easy to see that it is a useful algorithm for Equiv-AND-SIZE$[n^c]$, which completes the proof of Proposition 1.6. □

Most importantly, this proof shows that any improvement over Corollary 5.4 implies a corresponding improvement over Proposition 5.6. In addition, it is not hard to see that Conjecture 4.6 immediately implies the extension of Proposition 5.6 obtained by Aaronson and van Melkebeek [AvM11][13].

5.3 Useful properties and learning algorithms

The existence of learning algorithms in many different models yields circuit lower bounds, as first shown by Fortnow and Klivans [FK09]. In this section we discuss two frameworks for learning: deterministic exact learning from membership and equivalence queries (Angluin [Ang87]), and randomized PAC learning (Valiant [Val84]).

Exact learning algorithms. Let $\mathcal{C}$ be a typical circuit class. In this model, a *deterministic* algorithm is given access to oracles MQ^f and EQ^f for some function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{C}$. There oracles are defined as follows.

MQ^f: Given $x \in \{0,1\}^n$, returns $f(x)$.

EQ^f: Given a hypothesis $h : \{0,1\}^n \to \{0,1\}$ represented as a circuit, returns 1 if $h \equiv f$. Otherwise, returns an arbitrary input $x \in \{0,1\}^n$ such that $f(x) \neq h(x)$.

For a size function $s : \mathbb{N} \to \mathbb{N}$, we say that a learning algorithm $\mathcal{A}$ *exact learns* $\mathcal{C}[s(n)]$ in time $t(n)$ if for every $f \in \mathcal{C}$, when given access to oracles MQ^f and EQ^f, $\mathcal{A}$ runs in time at most $t(n)$, and outputs the description of a circuit C computing f. In particular, every equivalence query is invoked on a circuit of size at most $t(n)$, and the final hypothesis C is a circuit of size at most $t(n)$.

Recall that one of the main results from Fortnow and Klivans [FK09] states that exact learning a circuit class leads to circuit lower bounds against E^{NP} (Proposition 1.7). The original proof used by them is a clever combination of many results from complexity theory. Here we observe that it is relatively easy to prove results of this form using the machinery of useful properties. To simplify the argument even more, we can view learning as compression, which yields a quick proof of the following result.

Proposition 5.7 ("Learning yields circuit lower bounds").
Let $\mathcal{C}$ be a circuit class. Suppose there exists an exact learning algorithm for $\mathcal{C}[\mathsf{poly}]$ that runs in subexponential time. Then $\mathsf{NEXP} \nsubseteq \mathcal{C}[\mathsf{poly}]$.

---

[13]Here is a sketch of the argument. Assume that $\mathsf{NE} \cap \mathsf{coNE} \subseteq \mathsf{P/poly}$. Then by Conjecture 4.6 there is no P-property useful against $\mathsf{P/poly}$. However, it is possible to show that useful algorithms for satisfiability lead to useful properties. Altogether, these assumptions imply the desired strengthening of Corollary 5.4.

Proof. Let $\mathcal{A}$ be an exact learning algorithm for $\mathcal{C}$. It is easy to see that given any truth-table $tt(h) \in \{0,1\}^N$ from $\mathcal{C}[\text{poly}]$, we can simulate $\mathcal{A}$ on input h in time $2^{O(n)}$. In other words, it is possible to provide correct answers to the membership and equivalence queries asked during $\mathcal{A}$'s computation. By assumption, the learning algorithm outputs a circuit of subexponential size that computes h. This is therefore a valid compression algorithm for $\mathcal{C}[\text{poly}]$, and Proposition 5.7 follows immediately from Proposition 5.3 with $\delta = 1$. $\qquad\square$

In addition to its simplicity, this proof offers other advantages. The framework of useful properties is more flexible with respect to changes in the learning model. For instance, one could consider deterministic learning algorithms using equivalence queries over subsets $S \subseteq \{0,1\}^n$ encoded by subexponential size circuits, and only require that the learning algorithm outputs a hypothesis that is ε-close to the unknown concept. Again, Proposition 5.3 easily implies circuit lower bounds.

Next we turn our attention to randomized learning algorithms, a class of algorithms for which theorems of the form "learning implies circuit lower bounds" are still much weaker than their deterministic counterpart.

Randomized PAC learning algorithms. In the PAC learning framework, there is an unknown function $f \in \mathcal{C}$ that the learning algorithm is supposed to learn (after obtaining limited information about f). Here we concentrate on the stronger model in which the learner can ask membership queries, and only needs to learn under the uniform distribution[14]. In other words, the learner can query the value $f(x)$ on any input x, and should be able to obtain, with high probability, a good approximation h for f. In general, for any function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{C}[s(n)]$, given parameters n, ε (accuracy), δ (confidence), and an upper bound $s(n)$ on the size of the circuit computing f, the learning algorithm should output with probability at least $1 - \delta$ a hypothesis h such that $\Pr_x[f(x) \neq h(x)] \leq \varepsilon$ (i.e., h is ε-close to f), where the probability is taken over all strings x of size n under the uniform distribution. We measure the running time $t_{\mathcal{A}}(n, 1/\delta, 1/\varepsilon, s(n))$ of a learning algorithm $\mathcal{A}$ as a function of these parameters. As opposed to what is usually called proper learning, the learning algorithm is allowed to output the description of any circuit of size at most $t_{\mathcal{A}}(.)$ as its final hypothesis. For simplicity, we say that an algorithm $\mathcal{A}$ PAC learns $\mathcal{C}$ if it learns any function from $\mathcal{C}$ to accuracy $1/4$ with probability at least $1 - 1/n$.

It is known that the existence of a polynomial time PAC learning algorithm for $\mathcal{C}[\text{poly}]$ implies that $\mathsf{BPEXP} \nsubseteq \mathcal{C}[\text{poly}]$ (Fortnow and Klivans [FK09]). However, the same proof provides much weaker results for subexponential time learning, and it is an interesting open problem to show that the existence of subexponential time PAC learning algorithms lead to similar circuit lower bounds. The next proposition shows that this problem is related to the power of randomness in the context of useful properties. First, we extend the definition of useful properties to promise properties.

Definition 5.8 ("Promise properties useful against $\mathcal{C}$").
A promise property *of boolean functions $\mathcal{P} = (\mathcal{P}_{\text{yes}}, \mathcal{P}_{\text{no}})$ consists of two nonempty disjoint subsets of the set of all boolean functions. For a typical circuit class $\mathcal{C}$, $\mathcal{P}$ is said to be* useful *against $\mathcal{C}$ if, for all k, there are infinitely many positive integers n such that*

- $\mathcal{P}_{\text{yes}}(f) = 1$ *for at least one function $f : \{0,1\}^n \to \{0,1\}$, and*

---

[14]In other words, a transference theorem for this learning model is a stronger result. In addition, it is easy to see that the results discussed here hold under even more powerful learning models.

- $\mathcal{P}_{\mathsf{no}}(g) = 1$ *for all* $g : \{0,1\}^n \to \{0,1\}$ *that admits circuits from* $\mathcal{C}[n^k]$.

We say that a promise property $\mathcal{P}$ *is a* Γ-*property if its corresponding promise problem* $L_{\mathcal{P}}$ *is in* promise-Γ.

Proposition 5.9 ("Useful properties from randomized learning")**.**
Let $\mathcal{C}$ *be a typical circuit class. Suppose there exists a randomized algorithm* $\mathcal{A}$ *that* PAC *learns* $\mathcal{C}[\mathsf{poly}]$ *in time* $2^{n^{o(1)}}$. *Then there exists a* (promise-coRP)-*property that is useful against* $\mathcal{C}$.

Proof. We use a subexponential time randomized learning algorithm $\mathcal{A}$ for $\mathcal{C}$ to define a (promise) coRP-property $\mathcal{P}$ that is useful against $\mathcal{C}$. Consider the following randomized algorithm $\mathcal{B}$. Given the truth-table $tt(f_n) \in \{0,1\}^N$ of an arbitrary function $f_n : \{0,1\}^n \to \{0,1\}$, it simulates the computation of $\mathcal{A}$ over f_n, until $\mathcal{A}$ outputs a circuit C of size $2^{n^{o(1)}}$ as its final hypothesis. Algorithm $\mathcal{B}$ accepts f_n if and only if C is *not* 1/10-close to f_n.

It follows from Lemma 2.7 that for any large enough n there is a function h_n that cannot be 1/10-approximated by circuits of subexponential size (for definiteness, fix some constructive size bound). In other words, for any large n, there exists at least one function h_n not in $\mathcal{C}[\mathsf{poly}]$ that is accepted with probability one. In addition, since $\mathcal{A}$ is a PAC learning algorithm for $\mathcal{C}$, every function in $\mathcal{C}$ is rejected with high probability. Clearly, $\mathcal{B}$ computes a promise coRP-property that is useful against $\mathcal{C}$: $\mathcal{P}_{\mathsf{yes}}$ consists of boolean functions that cannot be approximated by circuits of subexponential size, and $\mathcal{P}_{\mathsf{no}} = \mathcal{C}$. $\square$

This result gives another example of the fundamental importance of the notion of useful properties in the context of results of the form "algorithms yield circuit lower bounds".

6   Some broad research directions

Here is a list of problems related to the results discussed in this survey that we find particularly interesting.

Strengthening the ACC lower bound. Williams proved that $\mathsf{NEXP} \nsubseteq \mathsf{ACC}$. It follows easily from Lemma 2.2 that either $\mathsf{P} \nsubseteq \mathsf{ACC}$ or $\mathsf{NEXP} \nsubseteq \mathsf{P/poly}$. Give an unconditional proof that one of these circuit lower bounds hold.

Lossy compression of ACC and TC_2^0. Design efficient lossy compression schemes for circuit classes such as ACC or TC_2^0. To the best of our knowledge, these results do not violate any widely believed cryptographic assumption.

Satisfiability algorithms. Can we make progress on satisfiability algorithms for threshold circuits?

SAT algorithms for depth $d+1$ versus lower bounds against depth d. Is it possible to show that, in general, nontrivial satisfiability algorithms for $\mathcal{C}_{d+1}$ lead to circuit lower bounds against $\mathcal{C}_d$?

Acknowledgements

References

[AAB+10] Scott Aaronson, Baris Aydinlioglu, Harry Buhrman, John M. Hitchcock, and Dieter van Melkebeek. A note on exponential circuit lower bounds from derandomizing arthur-merlin games. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:174, 2010.

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

[AGHK11] Baris Aydinlioglu, Dan Gutfreund, John M. Hitchcock, and Akinori Kawachi. Derandomizing Arthur-Merlin games and approximate counting implies exponential-size lower bounds. *Computational Complexity*, 20(2):329–366, 2011.

[AIV92] Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. *Manuscript*, 1992.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3), 2010.

[Ang87] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

[AvM11] Scott Aaronson and Dieter van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1), 2009.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizatons of the P = NP question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BIS12] Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating AC^0 by small height decision trees and a deterministic algorithm for $\#AC^0$. In *IEEE Conference on Computational Complexity*, pages 117–125, 2012.

[Cho11] Timothy Y. Chow. Almost-natural proofs. *J. Comput. Syst. Sci.*, 77(4):728–737, 2011.

[CKK+13] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:57, 2013.

[CN10] Stephen Cook and Phuong Nguyen. *Logical foundations of proof complexity.* Cambridge University Press, 2010.

[Coo73]   Stephen A. Cook. A hierarchy for nondeterministic time complexity. *J. Comput. Syst. Sci.*, 7(4):343–353, 1973.

[DH09]   Evgeny Dantsin and Edward A. Hirsch. Worst-case upper bounds. In *Handbook of Satisfiability*, pages 403–424. 2009.

[FK09]   Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009.

[FLvMV05]   Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, 2005.

[FSS84]   Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[Gol08]   Oded Goldreich. *Computational Complexity - A Conceptual Perspective*. Cambridge University Press, 2008.

[Gol11]   Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography*, pages 191–232. 2011.

[Hås87]   Johan Håstad. *Computational limitations of small-depth circuits*. MIT press, 1987.

[HH11]   Ryan C. Harkins and John M. Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. In *ICALP*, pages 416–423, 2011.

[IKK09]   Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *STOC*, pages 695–704, 2009.

[IKW02]   Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[Imm99]   Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[IMP12]   Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *SODA*, pages 961–972, 2012.

[IPS12]   Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth-2 threshold circuits. *CoRR*, abs/1212.4548, 2012.

[JMV13]   Hamidreza Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:20, 2013.

[Kab02]   Valentine Kabanets. Derandomization: a brief overview. *Bulletin of the EATCS*, 76:88–103, 2002.

[Kab13]   Valentine Kabanets. Private communication, 2013.

[KI04]   Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KKO13] Adam Klivans, Pravesh Kothari, and Igor C. Oliveira. Constructing hard functions from learning algorithms. In *IEEE Conference on Computational Complexity*, 2013.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *STOC*, pages 302–309, 1980.

[KL01] Matthias Krause and Stefan Lucks. Pseudorandom functions in TC^0 and cryptographic limitations to proving lower bounds. *Computational Complexity*, 10(4):297–313, 2001.

[Kra95] Jan Krajícek. *Bounded arithmetic, propositional logic, and complexity theory*. Cambridge University Press, 1995.

[KvMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Computational Complexity*, 21(1):3–61, 2012.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

[MVW99] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, pages 210–220, 1999.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[PPZ99] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

[Raz95] Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: Mathematics*, 59(1):205–227, 1995.

[Ros10] Benjamin Rossman. *Average-case complexity of detecting cliques*. PhD thesis, MIT, 2010.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[Rud97] Steven Rudich. Super-bits, demi-bits, and NP/qpoly-natural proofs. In *RANDOM*, pages 85–93, 1997.

[San10] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *FOCS*, pages 183–192, 2010.

[San12] Rahul Santhanam. Ironic complicity: Satisfiability algorithms and circuit lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:84, 2012.

[Sch13] Stefan Schneider. Satisfiability algorithms for restricted circuit classes. *CoRR*, abs/1306.4029, 2013.

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

[ST13] Kazuhisa Seto and Suguru Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Computational Complexity*, 22(2):245–274, 2013.

[SW12] Rahul Santhanam and Ryan Williams. Uniform circuits, lower bounds, and QBF algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:59, 2012.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[Tou01] Iannis Tourlakis. Time-space tradeoffs for sat on nonuniform machines. *J. Comput. Syst. Sci.*, 63(2):268–287, 2001.

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

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[Val84] Leslie G. Valiant. A theory of the learnable. In *STOC*, pages 436–445, 1984.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *STOC*, pages 231–240, 2010.

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *IEEE Conference on Computational Complexity*, pages 115–125, 2011.

[Wil13a] Ryan Williams. Private communication, 2013.

[Wil13b] Ryan Williams. Natural proofs versus derandomization. In *STOC*, pages 21–30, 2013.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *FOCS*, pages 1–10, 1985.

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

A NE ∩ i.o.coNE lower bounds from useful properties

In this section we describe the proof of Proposition 4.5, which we state again for convenience.

Proposition. *Let $\mathcal{C}$ be a typical circuit class. If for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$.*

This result is implicit in the work of Williams [Wil13b], and it consists of an interesting combination of nondeterminism, a collapse theorem, a hardness vs. randomness result, and simple diagonalization. We will need the following auxiliary results.

Lemma A.1. *Let $\mathcal{C}$ be a typical circuit class, and assume that $\mathsf{P} \subseteq \mathcal{C}[n^{\log n}]$. Then for every $d \in \mathbb{N}$, any function $f : \{0,1\}^n \to \{0,1\}$ computed by circuits of size $n^{\log^d n}$ is computed by circuits from $\mathcal{C}[n^{\log^{O(d)} n}]$.*

Proof. The result follows from a parameterized version of Lemma 2.2, and the proof is similar. $\square$

Lemma A.2 (Miltersen, Vinodchandran and Watanabe [MVW99]).
Let $g(n) > 2^n$ and $s(n) \geq n$ be functions that are both increasing and time-constructible. There exists a constant $d \in \mathbb{N}$ for which the following holds. If $\mathsf{E} \subseteq \mathsf{SIZE}(s(n))$ then $\mathsf{DTIME}[g(n)] \subseteq \mathsf{MATIME}[s(d \log g(n))^d]$.

For a function $h_\ell : \{0,1\}^\ell \to \{0,1\}$, let $\mathsf{CC}(h)$ be the size (number of gates) of the smallest circuit computing h.

Proposition A.3 (Umans [Uma03]).
There is a constant $k \in \mathbb{N}$ and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ for which the following holds. For every $s \in \mathbb{N}$ and boolean function $h_\ell : \{0,1\}^\ell \to \{0,1\}$ satisfying $\mathsf{CC}(h_\ell) \geq s^k$, and for all circuits C of size at most s over s inputs,

$$\left| \Pr_{z \in \{0,1\}^{k \cdot \ell}} [C(G(tt(h_\ell), z)) = 1] - \Pr_{z \in \{0,1\}^s} [C(z) = 1] \right| < \frac{1}{s}.$$

In addition, G can be computed in $\mathsf{poly}(2^\ell)$ time.

The next lemma shows that useful properties together with the lack of circuit lower bounds for P allow us to obtain a nontrivial derandomization of Merlin-Arthur games.

Lemma A.4. *Let $\mathcal{C}$ be a typical circuit class, and suppose that for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$. In addition, assume that $\mathsf{P} \subseteq \mathcal{C}[n^{\log n}]$. Then there is an infinite subset $S \subseteq \mathbb{N}$ such that for any $L \in \mathsf{MATIME}[n^{O(\log^3 n)}]$, there exists a language $L' \in \mathsf{NE}$ such that for every $n \in S$, we have $L_n = L'_n$. In addition, for all $n \notin S$, we have $L'_n = \emptyset$.*

Proof. First, observe that Lemma A.1 implies that for every $c \in \mathbb{N}$ there exists a property $\mathcal{P}_c$ that is useful against $\mathsf{SIZE}[n^{\log^c n}]$. Let $\mathcal{A}^c$ be an efficient algorithm computing $\mathcal{P}_c$ (we set the value of c later). Let $L \in \mathsf{MATIME}[n^{O(\log^3 n)}]$. There exists a MA-verifier V for L running in time $s = n^{O(\log^3 n)}$ such that

$$x \in L \quad \Longrightarrow \quad \exists y \in \{0,1\}^s \Pr_{w \in \{0,1\}^s} [V(x, y, w) = 1] \geq \frac{2}{3}$$

$$x \notin L \quad \Longrightarrow \quad \forall y \in \{0,1\}^s \Pr_{w \in \{0,1\}^s} [V(x, y, w) = 1] \leq \frac{1}{3}$$

Our nondeterministic algorithm $\mathcal{N}$ for L proceeds as follows. On input $x \in \{0,1\}^n$, it first guesses a string $y \in \{0,1\}^s$, then constructs a circuit $C_{x,y}$ from $\mathsf{SIZE}[s]$ such that for all $w \in \{0,1\}^s$ we have $C_{x,y}(w) = V(x,y,w)$. Then $\mathcal{N}$ guesses truth-tables $tt(h_m) \in \{0,1\}^M$ for every $m \in [2^{(\log n)^{5/(c+1)}}, 2^{(\log(n+1))^{5/(c+1)}})$, where $M = 2^m$ as usual. If $\mathcal{A}^c$ rejects all such functions, then $\mathcal{N}$ rejects x. Otherwise, let h_ℓ be the first function for which $\mathcal{A}^c(h_\ell) = 1$. Since $\mathcal{A}^c$ computes a useful property, $s = n^{O(\log^3 n)}$, and $\ell \geq 2^{(\log n)^{5/(c+1)}}$, for any $c \in \mathbb{N}$ we have:

$$\mathsf{CC}(h_\ell) \geq \ell^{\log^c \ell} \geq n^{\log^4 n} \gg s^k,$$

for any $k \in \mathbb{N}$ and sufficiently large n. Finally, $\mathcal{N}$ runs the algorithm granted by Proposition A.3 on $C_{x,y}$ using h_ℓ, and accepts its input x if and only if

$$\Pr_{z \in \{0,1\}^{k \cdot \ell}} [C_{x,y}(G(tt(h_\ell), z)) = 1] \geq \frac{1}{2}. \tag{3}$$

Observe that there exists an infinite set $S \subseteq \mathbb{N}$ such that for each $n \in S$ and for every $x \in \{0,1\}^n$, $\mathcal{N}$ is able to find a function h_ℓ for which $\mathsf{CC}(h_\ell) \geq s^k$, where k is the constant in the statement of Proposition A.3. Put another way, $\mathcal{N}$ is correct on input sizes in S, and by construction $\mathcal{N}$ rejects every other input whose input size is not in S. Also, S depends only on $\mathcal{P}_c$.

The (nondeterministic) running time of $\mathcal{N}$ is dominated by the computation of the probability in (3), and the time required to verify using $\mathcal{A}^c$ whether some hard function has been guessed. Finally, set $c = 5$, and observe that for this value of c we have $\ell \ll n$. It follows therefore that $\mathcal{N}$ runs in time at most 2^n. This completes the proof that there exists $L' \in \mathsf{NE}$ such that for every $n \in S$, $L'_n = L_n$, and for all $n \notin S$, we have $L'_n = \emptyset$. □

We are now ready to give the proof of Proposition 4.5.

Proof of Proposition 4.5. Assume that $\mathsf{NE} \cap \mathsf{i.o.coNE} \subseteq \mathcal{C}[n^{\log n}]$. In particular, $\mathsf{E} \subseteq \mathsf{SIZE}[n^{\log n}]$. Let $g(n) = 2^{n^{2\log n}}$ and $s(n) = n^{\log n}$. Using Lemma A.2, we get $\mathsf{DTIME}[2^{n^{2\log n}}] \subseteq \mathsf{MATIME}[n^{O(\log^3 n)}]$. Clearly, our assumptions also imply that $\mathsf{P} \subseteq \mathcal{C}[n^{\log n}]$.

Let $L \in \mathsf{DTIME}[2^{n^{2\log n}}]$. It follows from Lemma A.4 that there exists an infinite set $S \subseteq \mathbb{N}$ and a language $L' \in \mathsf{NE}$ such that $L_n = L'_n$ for every $n \in S$. Consider $\overline{L} \in \mathsf{DTIME}[2^{n^{2\log n}}]$, the complement of L. Then, again, there exists a language $L'' \in \mathsf{NE}$ such that for every $n \in S$, $\overline{L}_n = L''_n$. Clearly, $\overline{L''} \in \mathsf{coNE}$, and for every $n \in S$ we have $\overline{L''}_n = L_n = L'_n$. In other words, $L' \in \mathsf{NE} \cap \mathsf{i.o.coNE}$. Overall, we get

$$\mathsf{DTIME}[2^{n^{2\log n}}] \subseteq \mathsf{i.o.}(\mathsf{NE} \cap \mathsf{i.o.coNE}) \subseteq \mathsf{i.o.}\mathcal{C}[n^{\log n}],$$

where the last inclusion uses our initial assumption.

However, using a simple diagonalization argument, we can define a language $L^* \in \mathsf{DTIME}[2^{n^{2\log n}}]$ such that for all $n \geq n_0$, L^*_n is not computed by circuits from $\mathcal{C}[n^{\log n}]$. This contradiction completes the proof of Proposition 4.5. □