Improving Exhaustive Search Implies Superpolynomial Lower Bounds

Ryan Williams*
IBM Almaden Research Center

May 4, 2010

Abstract

The P vs NP problem arose from the question of whether exhaustive search is necessary for problems with short verifiable solutions. We do not know if even a slight algorithmic improvement over exhaustive search is universally possible for all NP problems, and to date no major consequences have been derived from the assumption that an improvement exists.

We show that there are natural NP and BPP problems for which minor algorithmic improvements over the trivial deterministic simulation already entail lower bounds such as NEXP $\not\subseteq$ P/poly and LOGSPACE $\neq$ NP. These results are especially interesting given that similar improvements *have* been found for many other hard problems. Optimistically, one might hope our results suggest a new path to lower bounds; pessimistically, they show that carrying out the seemingly modest program of finding slightly better algorithms for all search problems may be extremely difficult (if not impossible).

We also prove unconditional superpolynomial time-space lower bounds for improving on exhaustive search: there is a problem verifiable with $k(n)$ length witnesses in $O(n^a)$ time (for some a and some function $k(n) \leq n$) that cannot be solved in $k(n)^c n^{a+o(1)}$ time and $k(n)^c n^{o(1)}$ space, *for every* $c \geq 1$. While such problems can always be solved by exhaustive search in $O(2^{k(n)} n^a)$ time and $O(k(n) + n^a)$ space, we can prove a *superpolynomial* lower bound in the parameter $k(n)$ when space usage is restricted.

*This work originated while the author was a member of the Institute for Advanced Study in Princeton, New Jersey, supported by NSF Grant CCF-0832797 (Expeditions in Computing) and NSF Grant DMS-0835373 (Pseudorandomness). At IBM, the author is supported by the Josef Raviv Memorial Fellowship.

1 Introduction

To what extent can we avoid exhaustive search for generic search problems? This is one of the many offshoots of the P versus NP problem and it is the central question addressed in the area of exact algorithms for NP-hard problems.[1] The general message of this research has been that the trivial enumeration of all possible solutions *can* be quantitatively improved upon for many problems (and their number is increasing every year). The broad success of this program leads one to wonder if *all* NP problems allow some modest improvement over brute-force search. More precisely, let $V(x, y)$ be a verifier that runs on witnesses y of length $|x|^c$ and takes $O(|x|^d)$ time. The runtime of exhaustive search for a witness is $O(2^{|x|^c}|x|^d)$. Can we always find a witness in $O(2^{.99|x|^c}|x|^d)$? What about $O(2^{|x|^c - \log^2 |x|}|x|^{100d})$? Can we approximate the fraction of witnesses that V accepts in this time? If V uses only $s(n)$ space, can we find a witness in $O(2^{.99|x|^c}|x|^d)$ time and $O(|x|^{100c} + s(n))$ space? These questions are the focus of the present paper.

Here we offer the first concrete evidence that the above minor improvements will be difficult to achieve for some problems. We do not rule out the possibility of these improvements; rather, we show that they would already imply superpolynomial lower bounds in complexity theory. (Optimists might say we have given a new approach to proving class separations.) We show that the task of finding improved algorithms is connected naturally to the task of proving superpolynomial lower bounds. Regardless of one's complexity-theoretic worldview, our results show that one cannot simultaneously believe that "superpolynomial lower bounds are far beyond our current techniques" and "simple tricks should suffice for improved exponential algorithms."[2]

1.1 Main Results

1.1.1 Improved Algorithms Imply Circuit Lower Bounds

Let CIRCUIT SAT be the problem of finding a satisfying assignment to a Boolean circuit (over AND, OR, NOT gates). The CIRCUIT SAT problem is often the first NP-complete problem introduced in textbooks. Due to its applications in hardware/software verification and automated reasoning, the problem has been extensively studied in many areas of computer science. (Although CNF satisfiability gets all the fanfare, most formulas obtained in practice start as Boolean circuits.) Clearly, for circuits with n input variables and poly(n) gates, the problem can be solved in $2^n \cdot \text{poly}(n)$ time on any robust model of computation, and it is not known how to solve the problem even slightly faster than this.[3]

On a seemingly unrelated front, progress has been nearly nonexistent on proving superpolynomial circuit lower bounds for over a decade. It is known that the exponential-time version of Merlin-Arthur does not have polynomial size circuits [BFT98], but it is not known how to improve the lower bound to even $\mathsf{EXP}^{\mathsf{NP}}$, the class of problems solvable in exponential time with an NP oracle. Since it appears unlikely that NP does not have polynomial size circuits, this is indeed a frustrating state of affairs.

These two lines of research can be related to each other in a striking way:

[1] In fact, in Gödel's famous letter to Von Neumann, he posed precisely this question:

> "It would be interesting to know... in the case of finite combinatorial problems, how strongly *in general* the number of steps vis-á-vis *dem blossen Probieren* [the "bare trying" of solutions] can be reduced." [Har89]

[2] This was Mihai Patrascu's eloquent way of putting it.

[3] We use the poly(n) notation to denote $O(n^c)$ factors for a fixed $c > 0$ independent of n.

Theorem 1.1 *Suppose there is a superpolynomial function $s(n)$ such that* CIRCUIT SAT *on circuits with n variables and n^k gates can be solved in $2^n \cdot \text{poly}(n^k)/s(n)$ time by a (co-non)deterministic algorithm, for all k. Then* $\mathsf{NEXP} \not\subseteq \mathsf{P}/\text{poly}$.

That is, practically any nontrivial improvement over exhaustive search for CIRCUIT SAT (or nontrivial proofs of unsatisfiability) would already imply superpolynomial circuit lower bounds for nondeterministic exponential time. The best known deterministic algorithm for CNF satisfiability takes $2^{n-\Omega(n/\ln(m/n))}\text{poly}(m)$, where m is the number of clauses and n is the number of variables [DH08, CIP06]. For more complex circuits, the current state of the art is a randomized SAT algorithm for AC^0 circuits which runs in $2^{n-n^{1-o(1)}}$ time on circuits with $n^{1+o(1)}$ gates [CIP09]. Both time bounds are noticeably smaller than the hypothesis of Theorem 1.1.[4] Of course this does not mean that the hypothesis can be achieved, but it does seem possible at the present time.

One intuitive way of viewing Theorem 1.1 is that, if we could understand the structure of circuits well enough to solve their satisfiability problem faster, then this understanding could be translated into concrete lower bounds for those circuits. We solidify this intuition in two ways, by extending Theorem 1.1 to restricted circuit classes, and by showing that even faster Circuit Satisfiability would lead to stronger lower bounds.

For example, if satisfiability of Boolean formulas on n variables and n^c connectives can be solved in $2^n \cdot \text{poly}(n^c)/s(n)$, then E^{NP} does not have (non-uniform) polynomial size formulas.[5] If satisfiability of $\mathsf{AC}^0[6]$ circuits (constant depth circuits with AND, OR, NOT, and MOD6 gates) on n variables and n^c gates can be solved in $2^{n/2} \cdot \text{poly}(n^c)/s(n)$, then E^{NP} does not have polynomial size $\mathsf{AC}^0[6]$ circuits.

Interestingly, it does not take a much stronger assumption to get a nearly optimal circuit lower bound for E^{NP}, and consequently $\mathsf{EXP}^{\mathsf{NP}}$:

Theorem 1.2 *If* CIRCUIT SAT *on n variables and m gates is in $O(2^{(1-\delta)n}\text{poly}(m))$ (co-non)deterministic time for some $\delta > 0$, then there is $\varepsilon > 0$ and a language in E^{NP} that does not have $2^{\varepsilon n}$ size circuits.*

That is, an algorithm for CIRCUIT SAT with running time comparable to the best known algorithms for k-CNF SAT would entail strong circuit lower bounds.[6]

One may be skeptical that these small improvements are possible for Circuit Satisfiability. Similar results can also be established for problems efficiently solvable with randomness. The *Circuit Acceptance Probability Problem* (CAPP) is to approximate (within $\pm 1/6$) the fraction of satisfying assignments of a given Boolean circuit on n variables and n^c gates for some c [KC99, KRC00, For01, Bar02, IKW02]. CAPP can be solved in $O(n^c)$ time with a randomized algorithm. It is known that CAPP is in polynomial time if and only if $\text{PromiseBPP} = \mathsf{P}$ [For01], so the problem is "complete" in some sense. We show that essentially any nontrivial derandomization for CAPP implies superpolynomial circuit lower bounds.

[4] AC^0 circuits have constant depth and are comprised of AND, OR, NOT gates, with unbounded fan-in for each gate.

[5] Recall E^{NP} is the class of languages recognized by an algorithm running in $2^{O(n)}$ time that can query an NP oracle (with queries of $2^{O(n)}$ length). To the best of our knowledge, linear size circuit lower bounds are not known for this class. However an $n^{3-o(1)}$ lower bound on *formula* size follows from work of Hastad [Has98].

[6] This should be contrasted with the case of general CNF-SAT, where no algorithm of the kind required in Theorem 1.2 is known. Recent work with Patrascu indicates that the problem of finding such a CNF-SAT algorithm is also closely related to other problems in theoretical computer science [PW10].

Theorem 1.3 *If there is an $O(2^n \cdot \text{poly}(n^c)/s(n))$ nondeterministic algorithm for CAPP (for any super-polynomial $s(n)$), then* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$.

(Here, a *nondeterministic approximation algorithm* for CAPP always has at least one accepting computation, and always outputs a good approximation when it enters an accept state.) The proof of Theorem 1.3 holds even if we replace CAPP with the problem of distinguishing between circuits which are unsatisfiable and circuits with at least 2^{n-1} satisfying assignments. We interpret Theorem 1.3 as saying that *circuit lower bounds are much easier than derandomization*: any weak derandomization of CAPP would already yield a circuit lower bound that is much stronger than anything we currently know.

Theorem 1.3 also implies interesting "amplification" results:

Theorem 1.4 *If CAPP has a nondeterministic $O(2^n \cdot \text{poly}(n)/s(n))$ time algorithm that succeeds on all inputs, then for all $\varepsilon > 0$, CAPP has a nondeterministic $O(2^{n^\varepsilon})$ time algorithm with n^ε advice that succeeds on infinitely many inputs.*

Theorem 1.5 *If CAPP on n variables and m gates has a nondeterministic $O(2^{(1-\delta)n} \cdot \text{poly}(m))$ time algorithm that succeeds on all inputs, then CAPP has a nondeterministic polynomial time algorithm with $O(\log n)$ advice that succeeds on infinitely many inputs.*

The prospects seem better for achieving an improved CAPP algorithm. (Note we do not necessarily have to build a pseudorandom generator to achieve the simulations.) For CNF formulas of length n, Luby and Velikovic [LV96] have given a deterministic $n^{2^{O(\sqrt{\log \log n})}}$ algorithm for approximating the number of solutions. Note it was already known that $\mathsf{NEXP} \neq \mathsf{MA}$ iff $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ [IKW02]. However it could still have been the case that $\mathsf{NEXP} = \mathsf{MA}$ and yet Circuit SAT and CAPP have slightly better algorithms. The above results show this conjunction is impossible.

1.1.2 Improved Algorithms Imply $\mathsf{LOGSPACE} \neq \mathsf{NP}$ and *Subexponential* Algorithms

One strength of the above results is that there is no space restriction needed for the improved algorithms: our hypotheses only require SAT algorithms running in $O(2^n/s(n))$ time and $O(2^n/s(n))$ space for sufficiently large $s(n)$. Next we show that if exhaustive search can be improved in a way that preserves the verifier's space bound, then very surprising consequences result.

Here we shall study problems with *limited nondeterminism*, which only need short witnesses (e.g., of $\text{poly}(\log n)$ length) although verification still takes polynomial time. In particular we look at the case where the amount of nondeterminism is *logarithmic* in n ("log-nondeterminism"), and hence exhaustive search is already in *polynomial time*. This case is of interest as there are many polynomial time problems which fall in this category, and for which researchers have tried to find faster algorithms than a canonical one. The 3SUM problem is a well-known example [GO95]; satisfiability of exponential size circuits is another.[7]

It is natural to think that it may be easier to universally improve on log-nondeterminism problems. After all, the search space is no longer exponential in the running time of the verifier — now they are both

[7] In 3SUM, we are given a set A of n numbers and wish to find three numbers that sum to zero. The problem can be easily solved in $O(n^2)$, and it is a key bottleneck in the solution of many problems in geometry and data structures. Finding an $O(n^{1.99})$ algorithm is a major challenge. Note 3SUM (on $\text{poly}(\log n)$-bit numbers) can be reduced to CIRCUIT SAT with $\log n$ inputs and $O(n \cdot \text{poly}(\log n))$ gates. The idea is to sort A in increasing order, and on input $i \in [n]$ use two pointers (one starting at the beginning of A and one at the end) to sweep A for numbers a and b such that the ith number plus $a + b$ is zero. Thus any improvement on exhaustive search for exponential circuit satisfiability implies a 3SUM improvement.

polynomials in n. From this perspective it appears more likely that a clever pruning of the search space can be done.

Consider a problem solvable with witnesses of $\log n$ length, $O(n^c)$ time, and poly$(\log n)$ space.[8] The obvious deterministic simulation runs in $O(n^{c+1})$ time and poly$(\log n)$ space. We show that any universal improvement in the runtime exponent can be amplified into an *arbitrary polynomial* speedup with nondeterministic algorithms. Such a speedup would have dramatic consequences.

Theorem 1.6 *Suppose for all $c, d \geq 1$ that every problem Π solvable with $\log n$ nondeterministic bits, n^c time, and $(\log n)^d$ space can be solved by some deterministic algorithm in $O(n^{c+.99})$ time and* poly$(\log n)^d$ *space. Then every such Π can also be solved by some nondeterministic algorithm in $O(n^3)$ time.*

Of course, the .99 is not special, and can be substituted with any $\delta < 1$.

Corollary 1.1 *The hypothesis of Theorem 1.6 implies* LOGSPACE $\neq$ NP*; in fact,* SC $\neq$ NP*, where* SC *is the class of problems solvable in (simultaneous) polynomial time and polylogarithmic space.*

Corollary 1.2 *The hypothesis of Theorem 1.6 implies that the quantified Boolean formula problem has a proof system where every QBF of length n has proofs of $2^{\varepsilon n}$ length for all $\varepsilon > 0$.*

That is, either the exponent in the trivial algorithm is optimal for some constant c, or we separate complexity classes in a surprising way. Note if the trivial algorithm is optimal for $c = 1$, then SAT cannot be solved in subquadratic time and polylog space, a problem that remains open despite much effort.[9]

Identifying an explicit natural problem in place of "every problem" in Theorem 1.6 is nontrivial. (We could always use a form of the "Bounded Halting Problem", but this is undesirable.) The proof of Theorem 1.6 uses a delicate inductive argument that makes it difficult to extend to lower bounds on a natural problem.

1.1.3 Unconditional Lower Bounds

We want to understand the extent to which the exponential part of exhaustive search (namely, the exponential runtime in the witness length) can be reduced, without affecting the runtime of verification. We can prove unconditional lower bounds for improving on exhaustive search in this manner, based on Theorem 1.6 and its consequences. We first state a superpolynomial lower bound in terms of the witness length. To our knowledge no similar lower bounds have been reported.

Theorem 1.7 *There is a problem Π verifiable with $k(n)$-length witnesses in $O(n^a)$ time (for some constant a and some $k(n) \leq n$) that cannot be solved in $k(n)^c n^a \cdot$ poly$(\log n)$ time and $k(n)^c \cdot (\log n)^c$ space,* for all c.

Note if we could "only" change the "some $k(n)$" to "all $k(n)$" then we would separate P from NP. The theorem does rule out certain (daffy, but until now possible) strategies for trying to show LOGSPACE $=$ NP. For example, it is not possible to transform an arbitrary $O(n^a)$ time verifier with witnesses of length k into

[8] In place of poly$(\log n)$ space, one may substitute any constructible function $f(n)$ satisfying $\log n \leq f(n) \leq n^{o(1)}$ for the remaining results in this paper. We have chosen poly$(\log n)$ for concreteness.

[9] In fact, using the above footnote on 3SUM, one can show that if 3SUM on N numbers (expressible in poly$(\log N)$ bits) cannot be solved in $O(N^{2-\varepsilon})$ for all $\varepsilon > 0$, then SAT cannot be solved in $O(n^{2-\varepsilon})$, for all $\varepsilon > 0$.

an $O(n^{a+o(1)})$ time and $O(\text{poly}(\log n))$ space verifier with witnesses of length $10^{10} \log k$. The proof uses the framework of Theorem 1.6.

Extending this result to circuit satisfiability on general computational models is an interesting challenge. We manage to show a related lower bound for CIRCUIT SAT:

Theorem 1.8 *For every $\varepsilon > 0$,* CIRCUIT SAT *on k variables and m gates cannot be solved in $k^{1-\varepsilon} m^{1+o(1)}$ time and $m^{o(1)}$ space.*

This is already interesting, because we do not know how to prove that SAT is not in $O(n^{2-\varepsilon})$ time and $n^{o(1)}$ space on general models (the case where $k = m$).[10] It is known that SAT (and consequently, CIRCUIT SAT) cannot be solved in less than $n^{1.8}$ time and $n^{o(1)}$ space [Wil08b]. It is also known (in a precise sense) that current techniques will not allow us to extend this bound to n^2 [Wil10]. This limitation seems related to our inability to extend Theorem 1.8 to arbitrary polynomials in k.

1.1.4 A Nontrivial Simulation

In Appendix B, we report a baby step towards improving brute force search. We observe a nontrivial deterministic simulation of nondeterministic multitape Turing machines, inspired by a nontrivial derandomization of randomized multitape Turing machines of Van Melkebeek and Santhanam [vMS05].

Theorem 1.9 *For every multitape Turing machine M running in time t that accesses a tape of nondeterministic bits, there is a $\delta > 0$ and a deterministic TM that simulates M in $O(2^{(1-\delta)t})$ time.*

A group led by Lipton [Lip09a] has recently proved related results. Our simulation is not yet enough to imply lower bounds, as the simulation runtime does not scale with the amount of nondeterminism.

1.2 An Overview of Our Techniques

Our basic approach is simple: we assume the opposite of the desired lower bound (i.e., that we have decent uniform algorithms and very good non-uniform circuits), and construct an efficient simulation of the hard class, so tight that any nontrivial CIRCUIT SAT or CAPP algorithm contradicts a time hierarchy. This idea can be traced back to the first paper on P/poly, by Karp and Lipton [KL80]. One of their corollaries (credited to Meyer) is that $\mathsf{P} = \mathsf{NP}$ implies $\mathsf{EXP} \not\subseteq \mathsf{P}/\text{poly}$. To prove this, one shows that (1) $\mathsf{EXP} \subseteq \mathsf{P}/\text{poly}$ implies $\mathsf{EXP} = \Sigma_2\mathsf{P}$, and (2) $\mathsf{P} = \mathsf{NP}$ implies $\Sigma_2\mathsf{P} = \mathsf{P}$. Since $\mathsf{P} \neq \mathsf{EXP}$, it cannot be that both are true. Our results can be seen as extreme sharpenings of this basic result: to get circuit lower bounds for classes like NEXP, we may assume *far less* than $\mathsf{P} = \mathsf{NP}$.

To illustrate, let us sketch the result that faster circuit satisfiability implies NEXP circuit lower bounds. It is known that $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$ implies a simulation of $\mathsf{NTIME}[2^n]$ where we *existentially* guess a polynomial size circuit encoding a witness, then run an exponential time verifier [IKW02]. We make the observation that $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$ implies that *every* verifier for an NEXP language has small circuits that encode witnesses. Therefore we are free to construct any verifier we need to get a contradiction. We choose one that exploits efficient reductions from NP to SAT, translated up to NEXP. Using a small witness circuit, we can replace the exponential time verifier with a single call to CIRCUIT SAT on $n + O(\log n)$ variables and $\text{poly}(n)$ gates. It follows that an $O(2^n \cdot \text{poly}(n)/s(n))$ algorithm for CIRCUIT SAT implies

[10] Here n is the total length of the input.

$\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[2^n \cdot \text{poly}(n)/s(n)]$, a contradiction. For the lower bound that follows from approximating the solutions to a circuit, we use the highly efficient *PCP of Proximity* verifiers of Ben-Sasson *et al.* [BGHSV05] in our CIRCUIT SAT instance.

In order to prove that stronger improved algorithms yield stronger consequences such as $\mathsf{LOGSPACE} \neq \mathsf{NP}$, we use ideas from the existing time-space lower bounds for SAT, along with an inductive trick that (assuming a universal speedup over exhaustive search) lets us repeatedly reduce any exponent in the time bound of a complexity class until it becomes a fixed constant, leading to superpolynomial lower bounds. Further development of this trick produces unconditional lower bounds for bounded nondeterminism.

2 Preliminaries

We assume the reader is familiar with basic concepts in complexity and algorithms. In the following paragraphs, we focus on a few particulars needed for this paper.

2.1 Notation and Background

We define $[n] := \{1, \ldots, n\}$. As usual, unless otherwise specified a function has domain $\mathbb{N}$ and co-domain $\mathbb{N}$, is assumed to be time (or space) constructible within the appropriate bounds, and is monotone nondecreasing. All logarithms are in base two. A circuit is Boolean (with AND, OR, NOT gates), unless indicated otherwise.

Computational Model and Complexity. In almost all our results one may assume any deterministic computational model in which $O(\text{poly}(\log n))$ bits of information may be processed in a single step. This includes all flavors of Turing machines (including random access), random access machines with logarithmic size registers, and so on. (The main reason for this is given below.) We shall indicate clearly when this is not the case.

Fix a finite alphabet Σ. We usually assume $\Sigma = \{0, 1\}$. $\mathsf{NTIME}[t(n)]$ ($\mathsf{TIME}[t(n)]$) denote the classes of languages recognized in $O(t(n))$ time by a nondeterministic (deterministic) algorithm, respectively. $\mathsf{TISP}[t(n), s(n)]$ denotes those languages recognized by some algorithm that runs in both $O(t(n))$ time and $O(s(n))$ space. Recall $\mathsf{P} = \bigcup_{k\geq 1} \mathsf{TIME}[n^k]$ and $\mathsf{NP} = \bigcup_{k\geq 1} \mathsf{NTIME}[n^k]$. Steve's Class (abbreviated SC) is $\bigcup_{k\geq 1} \mathsf{DTISP}[n^k, (\log n)^k]$.

$i.o.-\mathcal{C}$ is the class of languages $L \subset \Sigma^*$ such that there is a language $L' \in \mathcal{C}$ where $L \cap \Sigma^n = L' \cap \Sigma^n$ holds for infinitely many n.

$(\exists\, f(n))\mathcal{C}$ is the class of languages which are recognized by machines with the following behavior: first, the machine nondeterministically guesses $f(n)$ bits, then it runs a subroutine recognizing a language in $\mathcal{C}$. The class $(\forall\, f(n))\mathcal{C}$ is defined similarly. For example, a language in the class $(\exists\, n)\mathsf{TISP}[n^2, \log n]$ is recognized by some machine that on input x guesses some $|x|$-bit string y then runs a computation taking $O(|x|^2)$ time and $O(\log |x|)$ space on the input $\langle x, y\rangle$. (We assume the machine has a tape alphabet large enough to accommodate an efficient pairing function.)

We shall use the following translation lemmas, whose proofs follow by standard "padding arguments.".

Lemma 2.1 (Translation Lemma) *For all constructible $f(n)$, $t_1(n)$, $t_2(n)$, and $s(n)$,*
If $\mathsf{NTIME}[t_1(n)] \subseteq \mathsf{TISP}[t_2(n), s(n)]$ then $\mathsf{NTIME}[t_1(f(n))] \subseteq \mathsf{TISP}[t_2(f(n)), s(f(n))]$.
If $\mathsf{TISP}[t_2(n), s(n)] \subseteq \mathsf{NTIME}[t_1(n)]$ then $\mathsf{TISP}[t_2(f(n)), s(f(n))] \subseteq \mathsf{NTIME}[t_1(f(n))]$.

We say that L is a *unary language* if $L \subseteq \{1\}^*$. We use the following nondeterministic time hierarchy theorem.

Theorem 2.1 ([Zak83]) *Let t_1 and t_2 be time constructible functions such that $t_1(n+1) \leq o(t_2(n))$. There is a unary language in $\mathsf{NTIME}[t_2(n)]$ that is not in $\mathsf{NTIME}[t_1(n)]$.*

Inspecting our proofs, one observes that we do not need the full strength of Theorem 2.1 but merely that there is a unary language in $\mathsf{NTIME}[t_1(n)\log^c t_1(n))]$ that isn't in $\mathsf{NTIME}[t_1(n)]$, for some $c > 0$. This particular time hierarchy is model-independent because the class $\mathsf{NTIME}[n \cdot \text{poly}(\log n)]$ is very robust with respect to the computational model. This is a family of results due to Gurevich and Shelah [GS89]. For example, let $\mathsf{NTIME}_{RTM}[t(n)]$ be the languages recognized with nondeterministic $t(n)$ time random-access Turing machines, and let $\mathsf{NTIME}_{TM}[t(n)]$ denote the same class for multitape Turing machines.

Theorem 2.2 (Gurevich and Shelah [GS89]) $\bigcup_{c>0} \mathsf{NTIME}_{RTM}[n \log^c n] = \bigcup_{c>0} \mathsf{NTIME}_{TM}[n \log^c n]$.

The random-access Turing machine can be replaced by any model M which has programs of $O(1)$ size and processes $O(\text{poly}(\log n))$ bits of information in a single step. Let us sketch the proof of Theorem 2.2 under this generic requirement. The idea is that a multitape TM can simulate an M-computation running in time $O(n \cdot \text{poly}(\log n))$ by guessing a computation history and verifying the correctness of the history. More precisely, the length of such a history is $O(n \cdot \text{poly}(\log n))$ bits: we use $O(\text{poly}(\log n))$ bits for each step, giving a "snapshot" which describes the timestep, bits read and written, the memory locations of these bits, and the state or program counter during that step. These snapshots are guessed in temporal order: the ith snapshot corresponds to the ith step. In this ordering, a multitape TM can verify that, assuming that all the reads are correct, the program counter and writes are all correct. A multitape TM can (in $O(n \cdot \text{poly}(\log n))$ time) sort these snapshots in *spatial* order as well, ordering the snapshots primarily by the index of the register/cell being read in that time step, and using the temporal order as a secondary key. In this ordering, a multitape TM can verify that the reads are correct by verifying that the previous snapshot *wrote* the same information.

2.2 Related Work

Subexponential Complexity and Parameterized Complexity. A theory of (sub)exponential time complexity for NP-hard problems has been developing. Much work has gone into extending reducibility notions to subexponential time algorithms (e.g., [IP01, IPZ01, CJ03, CG07, CIP06, Tra08]). The theory has found interesting evidence for the *exponential time hypothesis* (ETH) that 3-SAT is not in $2^{o(m)}$ time, where m is the number of clauses. If ETH is false, then many other problems have subexponential algorithms as well.

In the field of parameterized complexity initiated by Downey and Fellows [DF99], one studies paired problems with an input and an additional numerical parameter k. It is often convenient to define a parameterized problem when the computational problem of interest is hard, but only for certain inputs where the difficulty can be measured by the parameter. For example, the LONGEST PATH problem is NP-complete, but if one wants to find a longest path in a graph with treewidth at most k, there is a $2^{O(k \log k)}\text{poly}(n)$ algorithm [FL89]. Parameterized complexity is closely related to the study of exact algorithms for NP (although note that parameterized complexity can be used for other purposes as well, such as distinguishing between the efficiencies of approximation schemes, cf. [FG06]). A problem is *fixed-parameter tractable* (FPT) when it has an algorithm with $f(k)n^c$ runtime for some function f and a universal c.

If a problem does not seem to be FPT, one tries to find evidence for this, using reducibility notions similar to NP-completeness. There is a hierarchy of problems, all technically solvable in polynomial time when the parameter k is constant, which do not seem to be FPT. (In fact many hierarchies have been proposed; cf. the texts of Downey and Fellows [DF99] and Flum and Grohe [FG06].) The most basic is the W-hierarchy, where the lowest level is the class FPT of problems which are FPT. The next level W[1] already contains problems which are not in FPT, assuming ETH. There is a rough analogy between the W-hierarchy and the polynomial time hierarchy: in the W-hierarchy, $f(k)\log n$ bits are guessed in each alternation (instead of poly(n)), and the deterministic verifier only reads some $f(k)$ bits of the input. FPT equals the entire W-hierarchy if and only if Circuit SAT (with n variables and m gates) is in $2^{n/s(n)}\text{poly}(m)$ time for some unbounded $s(n)$ [ADF95]. (Note this runtime is faster than those in our hypotheses.) Indeed, there is a sense in which the aforementioned subexponential time theory is isomorphic to questions in parameterized complexity [CG07].

Limited Nondeterminism. Kintala and Fisher [KF77] initiated the study of classes with limited nondeterminism. Buss and Goldsmith [BG93] and Bloch, Buss, and Goldsmith [BBG98] considered classes with only logarithmic nondeterminism, as we study here. A related but more refined model (the guess-and-check model) was introduced by Cai and Chen [CC97]. Flum, Grohe, and Weyer [FGW06] gave a framework connecting limited nondeterminism with parameterized complexity.

Better Algorithms And Better Lower Bounds. The idea that better algorithms can lead to better lower bounds is a recurring theme in theoretical computer science. Here we cite a few examples which seem to be most relevant to our work.

- Work of Francis Zane and his co-authors from the late 90's [Zan98] alternated between finding faster SAT algorithms and proving new exponential lower bounds for depth-three circuits, both accomplished using new structure theorems for CNF formulas.
- In the DFA INTERSECTION problem, one is given k DFAs with at most n states each, and the task is to decide if all DFAs accept a common string. Karakostas, Lipton, and Viglas [KLV03] prove if DFA INTERSECTION is solvable in $n^{o(k)}$ time then LOGSPACE $\neq$ NP. However DFA INTERSECTION is PSPACE-complete [Koz77], while the problems we consider are either in BPP or NP, and we do not need to assume subexponential algorithms. Lipton has also advocated this type of approach for proving lower bounds in his popular blog [Lip09b].
- Impagliazzo, Kabanets, and Wigderson [IKW02] showed that if CAPP is solvable in nondeterministic $2^{n^{o(1)}}$ time infinitely often, then NEXP $\not\subseteq$ P/poly. Kabanets and Impagliazzo [KI04] showed that a nondeterministic $2^{n^{o(1)}}$ time algorithm for *Polynomial Identity Testing* implies that either NEXP $\not\subseteq$ P/poly or the Permanent does not have arithmetic circuits of polynomial size. Our results appear to be incomparable: while identity testing looks easier than the problems we study (even CAPP), our running time assumptions are much weaker and there is no "or" in our conclusion.
- Mike Fellows (personal communication) has observed that new separations between complexity classes in parameterized complexity would follow from improving exhaustive search. In particular, if there is an $n^{o(k)}$ time algorithm for k-Clique then M[1] $\neq$ XP, and if there is an $n^{o(k)}$ algorithm for k-Dominating Set then W[1] $\subseteq$ M[2] $\neq$ XP. (Very roughly speaking, the "W[1] vs XP" and "M[1] vs XP" problems are parameterized complexity versions of the "NP vs EXP" problem: see [DF99, FG06]

for definitions.) These observations follow from theorems which show that if the desired above algorithms exist, then the respective M-class is equal to FPT [CHKX06, CCFHJKX06]; however it is known that FPT $\neq$ XP [DF99].

If improved algorithms of the kind needed in this paper exist, their discovery probably will not be hindered by the known barriers in complexity theory (but perhaps another unforseen one). In Appendix C we briefly discuss this.

3 Improved Algorithms Imply Circuit Lower Bounds

We start by proving consequences of better CIRCUIT SAT algorithms. The notion of "universal witness circuits" shall be useful. A language L has universal witness circuits if every correct verifier for L has circuits that encode a witness accepted by the verifier, on every input in L.

Definition 3.1 *Let $L \in \mathsf{NTIME}[t(n)]$. A polynomial time algorithm V is a* verifier for L *if $x \in L \iff (\exists y \,:\, |y| \leq t(|x|))[V(x,y) = 1]$.*

Definition 3.2 *A language $L \in \mathsf{NTIME}[t(n)]$* has *$S(n)$-size universal witness circuits if* for all *polynomial time verifiers V for L, there is a k and an $O(S(n))$-size Boolean circuit family $\{C_n\}$ with the property:*

> *$x \in L \iff V(x, w(x)) = 1$, where $w(x)$ is the concatenation of the outputs of $C_\ell(\langle x, z\rangle)$ evaluated over all $z \in \{0,1\}^{\lceil \log_2 t(n)\rceil + 1}$ in lexicographical order, and $\ell \geq n$ is an integer of appropriate length.*

The universal witness property may look strong, but all of NEXP has universal witness circuits of polynomial size, in the case that NEXP $\subseteq$ P/poly. The key to this observation is that if a language does not have such circuits, then *some* correct verifier for NEXP accepts witnesses that *cannot* be encoded with small circuits, infinitely often. This verifier can be used to test strings for high circuit complexity, which is enough to obtain pseudorandom generators, leading to a contradiction. The following can be easily obtained from Impagliazzo, Kabanets, and Wigderson [IKW02]; we include a proof for completeness in Appendix A.

Lemma 3.1 (Follows from [IKW02]) *If* NEXP $\subseteq$ P/poly *then every language in* NEXP *has universal witness circuits of polynomial size.*

We also need a strong completeness result for satisfiability.

Theorem 3.1 (Tourlakis [Tou01], Fortnow *et al.* [FLvMV05]) *There is a fixed d such that for every $L \in \mathsf{NTIME}[n]$, L reduces to 3SAT in $O(n(\log n)^d)$ time. Moreover there is an algorithm (with random access to its input) that, given an instance of L and an integer $i \in [cn(\log n)^d]$ in binary (for some c depending on L), outputs the ith clause of the resulting 3SAT formula in $O((\log n)^d)$ time.*

The value of d depends on the particular computational model chosen. For most models one can take d to be small, e.g. $d = 4$. Theorem 3.1 holds under any computational model that accesses up to $O(\text{poly}(\log n))$ bits per step. Most proofs of it essentially rely the fact that nondeterministic Turing machines can simulate most other nondeterministic models with polylog overhead, as well as the Hennie-Stearns two-tape simulation of multitape Turing machines (cf. [AB09], Section 1.7) which introduces another log overhead. The Hennie-Stearns simulation can be converted (with constant overhead) into a circuit [PF79], which can then

be efficiently converted to 3-CNF using the Tseitin transformation [Tse68]. See Van Melkebeek ([vM07], Lemma 2.2) for an alternative proof based on sorting networks. By standard translation/padding arguments (substituting 2^n in place of n in the above), the theorem can be scaled up to exponential time bounds:

Corollary 3.1 *Every language L in $\mathsf{NTIME}[2^n]$ can be reduced to 3SAT instances of $c2^n \cdot n^4$ size. Moreover there is an algorithm that, given an instance of L and an integer $i \in [c2^n \cdot n^4]$ in binary, outputs the ith clause of the resulting 3SAT formula in $O(n^4)$ time.*

We are now ready to prove the main result of this section: a generic theorem relating the solvability of CIRCUIT SAT to lower bounds on universal witness circuits for NEXP.

Theorem 3.2 (Faster Circuit SAT Implies Circuit Lower Bounds) *Let $c \geq 1$. Let $a(n)$ be a monotone increasing and unbounded function. Let $S(n)$ and $T(n)$ be functions such that*

- $T(n)/(S(n) + n^8)^c \geq \Omega(n^4 \cdot a(n))$ *and*
- $n \leq S(n) \leq O(2^n/n \cdot 1/a(n))$.

Suppose CIRCUIT SAT *on n variables and m gates can be solved in $O(2^n m^c/T(n))$ co-nondeterministic time. Then $\mathsf{NTIME}[2^n]$ does not have $S(n)$-size universal witness circuits.*

Proof. Suppose $\mathsf{NTIME}[2^n]$ has $S(n)$ universal witness circuits. We show that a faster (co-non)deterministic algorithm for Circuit Satisfiability implies a contradiction to the nondeterministic time hierarchy.

Let $L \in \mathsf{NTIME}[2^n]$ be arbitrary. By Corollary 3.1, L can be reduced to 3SAT instances of $c2^n \cdot n^4$ size, for some c. Let $V(x, y)$ be a verifier for L that reduces x to a 3-CNF formula ϕ_x of $c2^n \cdot n^4$ size, substitutes the ith bit of y for the ith variable of ϕ_x, then returns 1 if and only if the resulting formula evaluates to 1.

Since L has universal witness circuits, it is the case that for all $x \in L$, there is some y of length at most $c2^n \cdot n^4$ such that $V(x, y) = 1$, and y can be encoded with a $S(|x|)$ size circuit. That is, for all x, there is a circuit C_y that takes inputs of length $\ell = \log(c2^{|x|}|x|^4)$ and has size at most $S(|x|)$, such that the witness y equals the concatenation of $C_y(z)$ over all $z \in \{0, 1\}^\ell$ in lexicographical order.

Consider the following nondeterministic algorithm N for L. On input x, existentially guess the circuit C_y, using $O(S(|x|) \log S(|x|)) \leq O(2^n/a(n))$ bits. Then construct a circuit D with ℓ input variables X, as follows. Given an integer $i \in [c2^n \cdot n^4]$, the ith clause of ϕ_x can be computed in $O(n^4)$ time (via Corollary 3.1). By the standard translation of algorithms into circuits, it follows that the ith clause can be computed with an $O(n^8)$ size circuit; call it E. Lead the input variables X of D into the inputs of E, whose $3n + O(\log n)$ output wires encode the Xth clause of ϕ_x. These output wires encode the indices of three variables in ϕ_x, along with three "negation bits" indicating 1 for variables which are negated, if any. For convenience, call the variable indices z_1, z_2, z_3. Evaluate $a_1 = C_y(z_1)$, $a_2 = C_y(z_2)$, and $a_3 = C_y(z_3)$. Letting b_1, b_2, b_3 be the negation bits of z_1, z_2, z_3 (respectively), output $\neg[(a_1 \oplus b_1) \vee (a_2 \oplus b_2) \vee (a_3 \oplus b_3)]$. That is, $D(X)$ outputs 1 if and only if the Xth clause is not satisfied, i.e., D is unsatisfiable if and only if C_y encodes a satisfying assignment for ϕ_x. The circuit D has $O(n^8 + S(|x|))$ size and ℓ input variables.

Finally, N calls a fast algorithm for circuit satisfiability on D, and *accepts* if and only if D is unsatisfiable. N runs in time

$$O(2^n/a(n) + 2^\ell \cdot (n^8 + S(n))^c/T(n)) \leq O(2^n/a(n) + 2^n n^4 \cdot (n^8 + S(n))^c/T(n)).$$

By assumption on $T(n)$ and $S(n)$, this time bound is at most $O(2^n/a(n))$.

Recall that L was an arbitrary language from $\mathsf{NTIME}[2^n]$, so we now have

$$\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[2^n/a(n)].$$

Since $2^{n+1}/a(n+1) = o(2^n)$, we have a contradiction to the strong nondeterministic time hierarchy theorem [SFM78, Zak83]. □

It is now easy to show that faster Circuit Satisfiability implies $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. We say that a function f is *superpolynomial in* n if for all k, $n^k = o(f(n))$.

Reminder of Theorem 1.1 *Suppose there is a superpolynomial function $s(n)$ such that* CIRCUIT SAT *on circuits with n variables and n^k gates can be solved in $2^n \cdot \text{poly}(n^k)/s(n)$ time by a (co-non)deterministic algorithm, for all k. Then* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$.

Proof. Let $S(n) = n^k$ and $T(n) = s(n)$ in Theorem 3.2. Then a $2^n \cdot \text{poly}(n^k)/s(n)$ time algorithm for Circuit Satisfiability implies that $\mathsf{NTIME}[2^n]$ does not have n^k universal witness circuits. Since k can be arbitrary, $\mathsf{NTIME}[2^n]$ does not have polynomial size universal witness circuits, hence NEXP also does not. By the contrapositive of Lemma 3.1, we conclude that $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. □

3.1 Extensions

It is also possible to extend Theorem 1.1 to larger circuit lower bounds, and to weaker problems such as FORMULA SAT, the problem of satisfying Boolean formulas. Unfortunately the derandomization results do not seem to apply to restricted circuit classes. Also, as far as we know, Lemma 3.1 does not extend to superpolynomial circuit sizes [Imp10], so we need another way to obtain universal witness circuits for NEXP. This can be accomplished with the following lemma:

Lemma 3.2 (Folklore) *Let $\mathcal{C}$ be any class of circuits. If E^{NP} has (non-uniform) circuits of size $S(n)$ from class $\mathcal{C}$, then $\mathsf{NTIME}[2^n]$ has universal witness circuits of size $S(n)$ from class $\mathcal{C}$.*

Proof. Let L be a language in $\mathsf{NTIME}[2^n]$. Let $V(x,y)$ be a nondeterministic verifier for L running in $d2^{|x|}$ time. Consider the following E^{NP} machine:

$N(x,i)$: *Binary search for the lexicographically smallest z such that $V(x,z)$ accepts, by querying:* given (x,y) where $|y| = d2^{|x|}$, is there $z \leq y$ such that $V(x,z)$ accepts? *Then output the ith bit of z.*

Note the queries can be computed in NP, and N needs at most $d2^n$ queries to the oracle. Since every such N has size $S(n)$ circuits from class $\mathcal{C}$, $\mathsf{NTIME}[2^n]$ has $S(n)$-size universal witness circuits from $\mathcal{C}$. □

Reminder of Theorem 1.2 *If* CIRCUIT SAT *on n variables and m gates is in $O(2^{(1-\delta)n}m^c)$ time for some $\delta > 0$ and $c \geq 1$, then there is $\varepsilon > 0$ and a language in E^{NP} that does not have $2^{\varepsilon n}$ size circuits.*

Proof. Let $T(n) = 2^{\delta n}$ and $S(n) = 2^{\delta n/c}/n^5$. Note the constraints of Theorem 3.2 are satisfied, hence we have: *if Circuit SAT on n variables and m gates is in $O(2^{(1-\delta)n}m^c)$ time, then* NEXP *does not have $2^{\delta n/c}/n^5$-size universal witness circuits.* The result follows from Lemma 3.2 and setting $\varepsilon < \delta/c$. □

To get lower bound consequences from FORMULA SAT algorithms, we need another complete problem.

Theorem 3.3 ([BGHSV05], Section 4) *There is a constraint graph problem* Π *such that every language in* $\mathsf{NTIME}[2^n]$ *can be reduced to graph instances* Π *of size* $\ell = O(2^n \text{poly}(n))$*, such that the graph can be succinctly represented by a* $\text{poly}(n)$ *size formula on* $n + O(\log n)$ *variables. In particular, on an input* i *the formula outputs the* i*th edge of the graph along with the relevant constraints for that edge.*

The proof goes through nearly the same process as the completeness theorem for SAT (Theorem 3.1).[11] First use the fact that every $\mathsf{NTIME}[2^n]$ language can be accepted by a nondeterministic multitape Turing machine in $O(2^n \text{poly}(n))$ time. (The polynomial factor depends on the computational model.) Then make this TM "oblivious" in that its head movements have extremely regular structure, running in $O(2^n \text{poly}(n)^2)$ time. Now [BGHSV05] show this TM computation can be represented by a (deBruijn) graph of $O(2^n \text{poly}(n)^4)$ size, where the edge function can be described by a $\text{poly}(n)$ size formula. The latter step uses the regular structure of the simulation. Carrying out the same argument as Theorem 3.2, and applying Lemma 3.2:

Theorem 3.4 *Let* $s(n)$ *be superpolynomial. If* FORMULA SAT *on* n *variables and* n^c *connectives can be solved in* $2^n \cdot \text{poly}(n^c)/s(n)$ *time, then* E^{NP} *does not have (non-uniform) polynomial size formulas.*

Similarly, let $\mathcal{C}$ be any non-uniform class of circuits that contains AC^0 and is closed under composition of circuits. That is, if $\{C_n\}$ and $\{D_n\}$ are families in $\mathcal{C}$, then those circuit families consisting of circuits which take n bits of input, feed them to at most $\text{poly}(n)$ copies of circuits from C_n, and feed those outputs to the inputs of $D_{\text{poly}(n)}$, are also circuit families in $\mathcal{C}$. (Note that all well-studied classes of circuits have this property.)

Theorem 3.5 *If satisfiability of* $\mathcal{C}$*-circuits on* n *variables and* n^c *gates can be solved in* $2^{n/3} \cdot \text{poly}(n^c)/s(n)$ *time, then* E^{NP} *does not have polynomial size* $\mathcal{C}$*-circuits.*

Proof. (Sketch) By Theorem 2.2, we may assume (with only $\text{poly}(n)$ extra multiplicative factors in computation time) that our machine model for nondeterminism is the multitape Turing machine. Let $L \in \mathsf{NTIME}[2^n]$ and let N be a nondeterministic multitape TM recognizing it. By a standard simulation, there is a single-tape machine N' that is equivalent to N and runs in $O(2^{2n})$ time and $O(2^n)$ space.

Now consider a reduction from $L(N')$ to 3SAT which uses the standard tableau reduction (originally due to Savage, cf. Theorem 7.30 in Sipser [Sip05]). Here the tableau has $O(2^{2n})$ rows and $O(2^n)$ columns, due to the time and space usage, respectively. This reduction creates a 3SAT instance on $O(2^{3n})$ clauses and $O(2^{3n})$ variables, such that if we are given an index i of $3n + O(1)$ bits, we can compute the ith clause of this formula with an AC^0 circuit of $\text{poly}(n)$ size. In particular, we can make an AC^0 circuit D that given the pair $(i, j) \in [O(2^{2n})] \times [O(2^n)]$ outputs a group of $O(1)$ clauses which represent the circuitry for computing cell (i, j) in an accepting tableau. The only dependence that this circuitry has on i and j is that i and j are *added* to certain variable indices. Since addition is in AC^0, the circuits have constant depth and polynomial size.

If E^{NP} has polysize $\mathcal{C}$ circuits, then there are universal witness $\mathcal{C}$ circuits for $\mathsf{NTIME}[2^{3n}]$ of polynomial size, by Lemma 3.2. So let M be a nondeterministic machine which on input x guesses a witness circuit C of $|x|^k$ size, constructs a circuit E which feeds its input i to D, obtains $O(|x|)$ output bits specifying $O(1)$ clauses, feeds all $O(1)$ variable indices from these clauses to copies of C, then evaluates these outputs. E outputs 0 iff C encodes a satisfying assignment for the ith clause of the reduction. Provided C and D are $\mathcal{C}$-circuits, E is a $\mathcal{C}$-circuit also, since $\mathcal{C}$ is closed under composition. Finally, M runs a satisfiability algorithm

[11] In fact, the reduction of Theorem 3.3 can probably be carried out with SAT, although we have not verified this.

on E and accepts iff E is unsatisfiable. By arguments similar to those of Theorem 3.2, we conclude that $L(M) = L$.

If satisfiability of $\mathcal{C}$ circuits with N variables and $\text{poly}(N)$ gates can be determined in $2^{N/3}\text{poly}(N)/s(N)$ time, the machine M runs in $2^n/s(n)$ time, contradicting the nondeterministic time hierarchy. □

The general approach also has the potential of proving non-linear circuit lower bounds for much smaller classes such as P^{NP}. In this setting, we would need faster SAT algorithms for circuits with very few inputs. Here is an example theorem:

Theorem 3.6 *Let $s(n)$ be superpolynomial. If there is a $c > 1$ such that* CIRCUIT SAT *on instances with cn inputs and $O(2^n \cdot \text{poly}(n))$ gates can be solved in $O(2^{cn}/s(n))$ time, then P^{NP} does not have linear size circuits.*

Proof. If P^{NP} has linear size circuits, then there are linear size universal witness circuits for NP. (The proof is analogous to Lemma 3.2.) Applying the succinct SAT reduction of Theorem 3.1 and arguing similarly to Theorem 3.2, we can simulate an $\mathsf{NTIME}[n^c]$ computation by guessing a linear size circuit, then solving a CIRCUIT SAT instance with $c \log n + O(\log \log n)$ inputs and $O(n)$ gates. (Note that almost all of this circuit's size comes from the three copies of the witness circuit; the remaining components have only $O(\text{poly}(\log n))$ size.) The presumed circuit satisfiability algorithm would imply an $o(n^c)$ time nondeterministic simulation, contradicting the nondeterministic time hierarchy. □

Extending our approach to conclude EXP lower bounds is an interesting open problem. In order to get a lower bound for EXP, it seems we need much stronger hypotheses. The following can be easily shown.

Proposition 1 *If* CIRCUIT SAT *on n inputs and m gates is in $2^{n^{o(1)}}\text{poly}(m)$ time, then $\mathsf{EXP} \not\subseteq \mathsf{P}/\text{poly}$.*

Proposition 2 *Let $f : \mathbb{N} \to \mathbb{N}$ satisfy $f(f(n^k)^k) \leq o(2^n/n^2)$ for all constants k. If 3SAT is in $O(f(n))$ time then $\mathsf{EXP} \not\subseteq \mathsf{P}/\text{poly}$.*

3.2 Extremely Weak Derandomization Implies Circuit Lower Bounds

We now turn to the *Circuit Acceptance Probability Problem* (CAPP): given a Boolean circuit C with n input variables and $\text{poly}(n)$ gates, the goal is to compute an approximation to the number of assignments satisfying C, within a factor of $1/6$. More precisely, we wish to output a number v such that

$$\left| v - \frac{1}{2^n} \cdot \sum_{x \in \{0,1\}^n} C(x) \right| < 1/6.$$

CAPP has been studied extensively [KC99, KRC00, For01, Bar02, IKW02]. We prove:

Reminder of Theorem 1.3 *If there is an $O(2^n \cdot \text{poly}(n)/s(n))$ nondeterministic algorithm for CAPP (for any superpolynomial $s(n)$), then $\mathsf{NEXP} \not\subseteq \mathsf{P}/\text{poly}$.*

In the previous section, we saw that strong reductions to 3SAT imply an efficient NEXP verifier. The verifier universally tries all possible clauses in an exponentially long 3-CNF formula, and checks the values of three variables in each trial. This universal quantifier is replaced with a $\text{poly}(n)$ size circuit which (on a variable assignment) checks the clause indexed by the input to the circuit. Our idea is to replace this

universal quantifier with a random choice, so that instead of testing satisfiability of the circuit, it suffices to approximate the number of solutions. Naturally, this suggests the use of *probabilistically checkable proofs* (PCPs). We use the *PCPs of Proximity* of Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan, which imply PCPs for NEXP with nice properties.[12]

Theorem 3.7 ([BGHSV05]) *Let $T : \mathbb{Z}^+ \to \mathbb{Z}^+$ be a non-decreasing function. Then for every $s > 0$ and every language $L \in \mathsf{NTIME}[T(n)]$ there exists a PCP verifier $V(x, y)$ with soundness s, perfect completeness, randomness complexity $r = \log_2 T(|x|) + O(\log \log T(|x|))$, query complexity $q = \text{poly}(\log T(|x|))$, and verification time $t = \text{poly}(|x|, \log T)$. More precisely:*

- *V has random access to x and y, uses at most r random bits in any execution, makes q queries to the candidate proof y. and runs in at most t steps.*
- *If $x \in L$ then there is a y of length $T(|x|)\text{poly}(\log T(|x|))$ such that $\Pr[V(x, y)$ accepts$] = 1$.*
- *If $x \notin L$ then for all y, $\Pr[V(x, y)$ accepts$] \leq s$.*

We shall be interested in the case $T(n) = 2^n$. Then the above PCP verifier uses $n + O(\log n)$ bits of randomness, $\text{poly}(n)$ verification time, and $\text{poly}(n)$ query complexity.

Proof of Theorem 1.3. The proof proceeds similarly to Theorem 1.1. We start by assuming $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, so that all languages in NEXP have universal witness circuits. Let L be a language in $\mathsf{NTIME}[2^n] - \mathsf{NTIME}[2^n \cdot \text{poly}(n)/s(n)]$.

Let $V(x, y)$ be the PCP verifier of Theorem 3.7 for L where $T(n) = 2^n$ and $s = 1/2$. For some $k > 0$, V tosses $n + k \log n$ coins, queries the $O(2^n)$ length string y in $O(n^k)$ places, and runs in $O(n^k)$ time. Let $c > 0$ be such that the circuit complexity of any function encoding a $2^n n^k$-bit witness of V on inputs of length n is at most $n^c + c$.

We now describe an alternative nondeterministic algorithm N for L. On input x of length n, existentially guess a circuit C with $n^c + c$ size. Construct a circuit D that simulates V as follows. The circuit D has $n + k \log n$ input wires r, corresponding to the random bits of V. Once r is fixed, the verifier V runs in $O(n^k)$ time and queries y in $O(n^k)$ positions. By the standard translation of oracle machines to circuits, V can be simulated by a circuit of size $O(n^{2k})$ with *oracle gates* for the string y. These oracle gates each have $n + O(1)$ input wires and output the bit of y indexed by the input wires. Replace each oracle gate with a copy of C. The resulting circuit D has $O(n^{2k+c})$ size. Run the presumed nondeterministic algorithm for CAPP on the D, and *accept* if and only if the fraction returned is greater than $3/4$.

Let y be the string obtained by evaluating C on all inputs in lexicographical order. By construction, $D(z) = 1$ if and only if $V(x, y)$ outputs 1 on the string of coin tosses z.

If there is a witness y for the input x, then there is a C of size $O(n^c)$ which encodes y. On such a C, the fraction of inputs accepted by D is 1. On the other hand, if there is no witness y for x, then certainly no circuit C encodes a witness for x, and hence on every circuit C, the fraction of inputs accepted by D is at most $1/2$. Hence the algorithm N is correct.

Assuming CAPP can be solved in $2^n \cdot \text{poly}(n)/s(n)$ time for some superpolynomial $s(n)$, the running time of the entire simulation is $2^n \cdot \text{poly}(n)/s(n)$. This contradicts the choice of L. □

[12] Note that Or Meir has recently found a more combinatorial approach to these results, cf. [Mei09].

The above proof can be generalized along the lines of Theorem 3.2 in a straightforward way. A completely analogous argument to Theorem 1.2 shows the following.

Theorem 3.8 *If there is an $O(2^{(1-\delta)n}m^c)$ nondeterministic algorithm for CAPP on circuits of n variables and m gates, then* NEXP *does not have $2^{\varepsilon n}$-size universal witness circuits for some $\varepsilon > 0$. Consequently,* E^{NP} *does not have $2^{o(n)}$ size circuits.*

Impagliazzo, Kabanets, Wigderson [IKW02] showed partial converses to Theorem 1.3 and Theorem 3.8:

- If $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$, then CAPP has a nondeterministic algorithm that works on infinitely many inputs in $O(2^{n^\varepsilon})$ time with n^ε advice, for every $\varepsilon > 0$.
- (Implicitly shown) If $\mathsf{NTIME}[2^n]$ does not have $2^{o(n)}$-size universal witness circuits, then CAPP has a nondeterministic polytime algorithm that works on infinitely many inputs with $O(\log n)$ advice.[13]

Combining these results with Theorem 1.3 and Theorem 3.8, we have the following "amplifications" of CAPP algorithms:

Reminder of Theorem 1.4 *If CAPP has a nondeterministic $O(2^n \cdot \text{poly}(n)/s(n))$ time algorithm that succeeds on all inputs, then for all $\varepsilon > 0$, CAPP (in fact, all of* MA*) has a nondeterministic $O(2^{n^\varepsilon})$ time algorithm with n^ε advice that succeeds on infinitely many inputs.*

Reminder of Theorem 1.5 *If CAPP on n variables and m gates has a nondeterministic $O(2^{(1-\delta)n} \cdot \text{poly}(m))$ time algorithm that succeeds on all inputs, then CAPP has a nondeterministic polynomial time algorithm with $O(\log n)$ advice that succeeds on infinitely many inputs.*

The above two results can also be viewed as gap theorems: if CAPP can't be solved infinitely often in NP with logarithmic advice, then CAPP can't be in $2^{.99n} \cdot \text{poly}(m)$ nondeterministic time.

Evidence for Derandomization of CAPP(?) The hypothesis of Theorem 1.3 looks potentially provable to us. Van Melkebeek and Santhanam [vMS05] have shown that for every probabilistic k-tape Turing machine running in time t, there is a $\delta > 0$ such that the machine can be simulated deterministically in $O(2^{(1-\delta)t})$ time. Unfortunately this is not quite enough to achieve the circuit lower bound for NEXP. To do that, we would need (for example) that every probabilistic k-tape Turing machine with $b(n)$ bits of randomness running in time $t(n)$ can be simulated deterministically in $\text{poly}(t) \cdot 2^b/s(b)$ time, for a superpolynomial function s. In our desired application, $t(n)$ is a polynomial factor larger than $b(n)$.

[13] This follows because, if $\mathsf{NTIME}[2^n]$ did not have such circuits, then there is an $\varepsilon > 0$ and a $\text{poly}(2^n)$ time algorithm with n bits of advice that, on infinitely many inputs, nondeterministically generates a 2^n-bit truth table of a Boolean function f_n which has circuit complexity at least $2^{\varepsilon n}$ for sufficiently large n. (This is essentially the negation of what it means to have universal witness circuits: there is a verifier for some L which on infinitely many inputs $x_i \in L$, only accepts witnesses which *do not* have small circuits. Hardcoding these x_i as advice, we get that any witness accepted by the verifier on x_i has high circuit complexity.) But it is known (even from [KvM99]) that this assumption implies pseudorandom generators for polynomial size circuits, strong enough to prove $\mathsf{MA} \subseteq i.o. - \mathsf{NP}/O(\log n)$. Hence CAPP has the desired type of algorithm.

4 Further Improvements Imply LOGSPACE $\neq$ NP and Subexponential proofs for QBF

We now show that improving the runtime of exhaustive search for problems with limited nondeterminism would have surprising consequences. Our results use a tool from prior work on time-space lower bounds for satisfiability. Namely, we use a lemma to "speed up" arbitrary space-bounded computations with alternating machines. (The notation below is discussed in the Preliminaries.[14])

Lemma 4.1 (Speedup Lemma [Kan84, FLvMV05]) *Let $a \geq 2$ and $\log n \leq s(n) \leq o(n)$. Then*

$$\mathsf{TISP}[n^a, s(n)] \subseteq (\exists\, n)(\forall\ \log n - \log s(n))\mathsf{TISP}[n^{a-1} \cdot s(n), s(n)].$$

Moreover, the TISP *part of the RHS only reads $n + O(s(n))$ bits of its input, and erases the rest.*

Proof. Let M be an algorithm using n^a time and $s(n)$ space. Note that a configuration of M on an input of length n (a description of the entire state of the algorithm at any moment in time) can be represented within $\ell = dS(n)$ bits, for some $d > 1$.

To simulate M in $(\exists\, n)(\forall\ \log n - \log s(n))\mathsf{TISP}[n^{a-1} \cdot s(n), s(n)]$, the machine $N(x)$ existentially guesses a sequence of configurations $C_1, \ldots, C_{n/\ell}$ of $M(x)$, using n bits. Then $N(x)$ appends the initial configuration C_0 to the beginning of the sequence and the (unique) accepting configuration $C_{(n/\ell)+1}$ to the end of the sequence. Then $N(x)$ universally guesses $i \in \{0, \ldots, n/\ell\}$ using $\log(n/\ell)+1 \leq \log n - \log s(n)$ bits, erases all configurations except C_i and C_{i+1}, then simulates $M(x)$ starting from C_i, accepting if and only if the C_{i+1} is reached within $n^{a-1} \cdot \ell$ steps. It is easy to see the simulation is correct. □

The key idea behind our results is to exploit the fact that the universal quantifier in the Speedup Lemma is only *logarithmic*. Any improvement over exhaustive search of all $\log n - \log s(n)$ bit strings gives us a very slight advantage, which can be amplified by repeated applications of the assumed algorithm. We arrive at the main theorem of this section.

Reminder of Theorem 1.6 *Suppose there is $\delta > 0$ such that for all $c, d \geq 1$, every problem Π solvable with $\log n$ nondeterministic bits, n^c time, and $(\log n)^d$ space can be solved by a deterministic algorithm in $O(n^{c+(1-\delta)})$ time and $\mathrm{poly}(\log n)^d$ space. Then every such Π can also be solved with a nondeterministic algorithm in $O(n^3)$ time, i.e., $\mathsf{SC} \subseteq \mathsf{NTIME}[n^3]$.*

In fact, the proof shows that $(\exists\, n)\mathsf{TISP}[n^k, \mathrm{poly}(\log n)] \subseteq (\exists\, O(n))\mathsf{TISP}[n^3, \mathrm{poly}(\log n)]$, given the hypothesis. (Interestingly, this containment of classes is not known to be false, but as we shall see, it would be surprising if it were true.) The containment is proven by applying the Speedup Lemma along with the hypothesis for a constant number of times, with different values of c in each application.

Proof of Theorem 1.6. Written in our complexity class notation, the hypothesis is:

$$\forall c, d \geq 1,\ \exists k \geq 1\ \ (\exists \log n)\mathsf{TISP}[n^c, (\log n)^d] \subseteq \mathsf{TISP}[n^{c+1-\delta}, (\log n)^{dk}]. \tag{1}$$

By taking the complement of both sides, we also have $(\forall \log n)\mathsf{TISP}[n^c, (\log n)^d] \subseteq \mathsf{TISP}[n^{c+1-\delta}, (\log n)^{dk}]$. That is, there is always a deterministic algorithm which can determine if there is a witness iff there is always one that can determine if every string is a witness.

[14] Readers familiar with the notation of our prior work [Wil08b, Wil10] should beware that the notation in this paper has very slightly different meaning. In this work, we must pay attention to the constant factors in alternations, whereas in prior work we did not bother even with $n^{o(1)}$ factors. Our notation here reflects these differences.

Assume (1). First we prove the following containment by induction on ℓ, for all integers $\ell > 0$, $k > 2$, and $d > 0$ satisfying $k - \delta\ell \geq 2$:

$$(\exists\, n)\mathsf{TISP}[n^k, (\log n)^d] \subseteq (\exists\, \ell n)\mathsf{TISP}[n^{k-\delta\ell} \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)], \tag{2}$$

(Note the poly factors depend only on k, d, and ℓ.)

When $\ell = 1$, we have

$$(\exists\, n)\mathsf{TISP}[n^k, (\log n)^d] \subseteq (\exists\, n)(\forall \log n - d \log\log n)\mathsf{TISP}[n^{k-1}(\log n)^d, (\log n)^d] \tag{3}$$

by the Speedup Lemma (Lemma 4.1). Consider the class $(\forall \log n - d\log\log n)\mathsf{TISP}[n^{k-1}(\log n)^d, (\log n)^d]$ on the RHS of (3). An algorithm from the class receives an input of length $2n$ (the original input and the n bits guessed). Applying (1) with $c = k - 1$ and taking the complement, the universal quantifier can be removed, concluding that

$$(\forall \log n - d \log\log n)\mathsf{TISP}[n^{k-1}(\log n)^d, (\log n)^d] \subseteq \mathsf{TISP}[n^{k-1+(1-\delta)} \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)],$$

provided that $k \geq 2$. Substituting the RHS of the above into (3), we obtain

$$(\exists\, n)\mathsf{TISP}[n^k, (\log n)^d] \subseteq (\exists\, n)\mathsf{TISP}[n^{k-\delta} \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)].$$

This completes the base case of (2). For the inductive step, consider the following chain of containments. First, the induction hypothesis says

$$(\exists\, n)\mathsf{TISP}[n^k, (\log n)^d] \subseteq (\exists\, \ell n)\mathsf{TISP}[n^{k-\delta\ell} \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)].$$

Note the $\mathsf{TISP}[n^{k-\delta\ell}\cdot\mathrm{poly}(\log n), \mathrm{poly}(\log n)]$ part on the RHS receives an input of length $(\ell+1)n \leq O(n)$. Applying the Speedup Lemma to this part, the above class is then contained in

$$(\exists\, \ell n)(\exists\, n)(\forall\ \log n - \log\log n)\mathsf{TISP}[n^{k-\delta\ell-1}\mathrm{poly}(\log n), \mathrm{poly}(\log n)].$$

Observe that $\exists$ quantifiers can be merged, resulting in the class

$$(\exists\, (\ell+1)n)(\forall\ \log n - \log\log n)\mathsf{TISP}[n^{k-\delta\ell-1}\mathrm{poly}(\log n), \mathrm{poly}(\log n)].$$

Applying (1) with $c = k - \delta\ell - 1$ (this is possible when $k - \delta\ell - 1 \geq 1$), the above is in

$$(\exists\, (\ell+1)n)\mathsf{TISP}[n^{k-\delta\ell-1+(1-\delta)}\mathrm{poly}(\log n), \mathrm{poly}(\log n)].$$

Finally, note that the exponent $k - \delta\ell - 1 + (1 - \delta) = k - \delta(\ell + 1)$. This completes the proof of (2).

Now let $k > 3$ be arbitrary. Since $\delta > 0$, the quantity $\delta\ell$ is positive and can be made as large as desired, provided that $k - \delta\ell \geq 2$.

Set $\ell = \lfloor (k-2)/\delta \rfloor$. We have $k - \delta\ell \geq k - \delta(k-2)/\delta = 2$, and

$$k - \delta\ell = k - \delta\lfloor (k-2)/(\delta) \rfloor < k - \delta((k-2)/(\delta) - 1) = 2 + \delta < 3.$$

That is,

$$(\exists\, n)\mathsf{TISP}[n^k, (\log n)^d] \subseteq (\exists\, \ell n)\mathsf{TISP}[n^{k-\delta\ell} \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)]$$

by (2), and

$$(\exists\, \ell n)\mathsf{TISP}[n^{k-\delta\ell} \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \subseteq (\exists\, \ell n)\mathsf{TISP}[n^3, \mathrm{poly}(\log n)] \subseteq \mathsf{NTIME}[n^3],$$

by our choice of ℓ. □

Reminder of Corollary 1.1 *The hypothesis of Theorem 1.6 implies* $\mathsf{LOGSPACE} \neq \mathsf{NP}$.

Proof. If $\mathsf{LOGSPACE} = \mathsf{NP}$ then there is a c such that $\mathsf{NTIME}[n] \subseteq \mathsf{TISP}[n^c, \log n]$. We have $\mathsf{NTIME}[n^4] \subseteq \mathsf{TISP}[n^{4c}, \log n]$ by the translation lemma (Lemma 2.1). By Theorem 1.6, $\mathsf{TISP}[n^{4c}, \log n] \subseteq \mathsf{NTIME}[n^3]$. This is a contradiction to the nondeterministic time hierarchy. □

Reminder of Corollary 1.2 *The hypothesis of Theorem 1.6 implies that the quantified Boolean formula problem has a proof system where every QBF of length n has proofs of $2^{\varepsilon n}$ length for all $\varepsilon > 0$.*

Proof. Theorem 1.6 says that for all k, $\mathsf{TISP}[n^k, O(\log n)] \subseteq \mathsf{NTIME}[n^3]$.

Let $\varepsilon > 0$, and let $f(n) = 2^{\varepsilon n/3}$. By the translation lemma (Lemma 2.1), $\mathsf{TISP}[f(n)^k, O(\log f(n))] \subseteq \mathsf{NTIME}[f(n)^3]$, which is $\mathsf{TISP}[2^{\varepsilon kn/3}, O(n)] \subseteq \mathsf{NTIME}[2^{\varepsilon n}]$. Since k can be arbitrarily large, it follows that $\mathsf{SPACE}[O(n)] \subseteq \mathsf{NTIME}[2^{\varepsilon n}]$. A quantified Boolean formula of length n can be easily solved using $O(n)$ space. Therefore quantified Boolean formulas of length n can be solved in $2^{\varepsilon n}$ time with a nondeterministic algorithm. The conclusion follows. □

5 Unconditional Lower Bounds

The ideas of the previous section lead to an *unconditional* lower bound.

Reminder of Theorem 1.7 *For some a and some $k(n) \leq n$, there is a problem Π solvable with $k(n)$ nondeterminism and $O(n^a)$ time that cannot be solved in $k(n)^c n^a \cdot \mathrm{poly}(\log n)$ time and $k(n)^c \cdot \mathrm{poly}(\log n)$ space,* for all constants c.

Proof. Assume the opposite: for all $a \geq 1$ and for every problem Π solvable with $k(n)$ nondeterminism and $O(n^a)$ time, there is a c_a such that Π can be solved in $k(n)^{c_a} n^a \cdot \mathrm{poly}(\log n)$ time and $k(n)^{c_a} \cdot \mathrm{poly}(\log n)$ space. For each $a \geq 1$, define

$$\Pi_a = \{(i, x) \mid \exists y \in \{0,1\}^{\log |x|+1} \text{ machine } M_i(x, y) \text{ accepts within } |x|^a + a \text{ steps}\}.$$

Setting $k(n) = \log n$, we have by assumption that

$$\Pi_a \in \mathsf{TISP}[n^a (\log n)^{c_a} \mathrm{poly}(\log n), (\log n)^{c_a} \mathrm{poly}(\log n) \mathrm{poly}(\log n)].$$

By efficiently reducing all languages in $(\exists\, \log n)\mathsf{TIME}[n^a]$ to Π_a, we have for all $a \geq 1$ that

$$\begin{aligned} &(\exists\, \log n)\mathsf{TIME}[n^a] \subseteq \mathsf{TISP}[n^a \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \\ &(\forall\, \log n)\mathsf{TIME}[n^a] \subseteq \mathsf{TISP}[n^a \cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)]. \end{aligned} \tag{4}$$

Setting $k(n) = n$ and $a = 2$, we have a polynomial time algorithm for SAT. Let ℓ be an integer such that SAT is in $O(n^\ell)$ time. By Theorem 3.1, we have

$$\mathsf{NTIME}[n] \subseteq \mathsf{TIME}[n^\ell \mathrm{poly}(\log n)]. \tag{5}$$

Similar to the proof of Theorem 1.6, we derive

$$\begin{aligned}
\mathsf{TIME}[n^{\ell+1}] &\subseteq \mathsf{TISP}[n^{\ell+1}\cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \quad \text{(applying (4) with } a=\ell+1)\\
&\subseteq (\exists\, n)(\forall\, \log n)\mathsf{TISP}[n^{\ell}\cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \quad \text{(Speedup Lemma)}\\
&\subseteq (\exists\, n)\mathsf{TISP}[n^{\ell}\cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \quad \text{(applying (4))}\\
&\subseteq (\exists\, 2n)(\forall\, \log n)\mathsf{TISP}[n^{\ell-1}\cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \quad \text{(Speedup Lemma)}\\
&\subseteq (\exists\, 2n)\mathsf{TISP}[n^{\ell-1}\cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)] \quad \text{(applying (4))}\\
&\subseteq \ldots\\
&\subseteq (\exists\, \ell n)\mathsf{TISP}[n\cdot \mathrm{poly}(\log n), \mathrm{poly}(\log n)]\\
&\subseteq \mathsf{NTIME}[n\cdot \mathrm{poly}(\log n)] \quad \text{(trivial)}\\
&\subseteq \mathsf{TIME}[n^{\ell}\cdot \mathrm{poly}(\log n)], \quad \text{(applying (5))}
\end{aligned}$$

contradicting the deterministic time hierarchy. □

In fact, a stronger statement holds: either polynomial nondeterminism cannot be simulated in polytime, or we have strong time lower bounds on simulating log-nondeterminism with polylog space.

Theorem 5.1 *Either* $\mathsf{P} \neq \mathsf{NP}$ *or there is a problem solvable with* $\log n$ *bits of nondeterminism in* $O(n^c)$ *time that is* not *solvable in* $O(n^{c+.99})$ *time and* $\mathrm{poly}(\log n)$ *space, for some* $c \geq 1$.

Proof. Assume the opposite, so that $\mathsf{P} = \mathsf{NP}$ and

$$(\exists \log n)\mathsf{TIME}[n^c] \subseteq \mathsf{TISP}[n^{c+.99}, \mathrm{poly}(\log n)]. \tag{6}$$

for all c. Note that (6) implies the hypothesis of Theorem 1.6, so $\mathsf{SC} \subseteq \mathsf{NTIME}[n^3]$ by Theorem 1.6. Moreover, (6) implies that $\mathsf{P} \subseteq \mathsf{SC}$. Therefore $\mathsf{NP} \subseteq \mathsf{NTIME}[n^3]$, a contradiction. □

We cannot yet extend these lower bounds to problems like Circuit SAT on general computational models, due to the inefficiency of reductions from arbitrary languages to Circuit SAT. We could extend them to Circuit SAT on multitape Turing machines, but those lower bounds are easy: the opposite of the lower bound implies that Circuit Evaluation (the case where there are *no* input variables) is in $n \cdot \mathrm{poly}(\log n)$ time and $\mathrm{poly}(\log n)$ space, which is already known to be false. (In fact, on multitape Turing machines the set of palindromes requires nearly quadratic time in the polylog space setting.)

However we can extend the lower bound slightly to generic computational models, using the strong nondeterministic time hierarchy [Zak83] (Theorem 2.1).

Theorem 5.2 *For all* $\varepsilon > 0$, CIRCUIT SAT *on circuits with* k *input gates and* m *total gates cannot be solved in* $m^{1+o(1)}k^{1-\varepsilon}$ *time and* $m^{o(1)}$ *space.*

Recall we do not know how to prove that SAT can't be solved in $O(N^{2-\varepsilon})$ time and $N^{o(1)}$ space on general models (the particular case where $k = m$).

Proof. Let $\varepsilon > 0$ be arbitrarily small. By Theorem 2.1, there is a unary language L that is in $\mathsf{NTIME}[n^2]$ but not in $\mathsf{NTIME}[n^{2-\varepsilon+\varepsilon'}]$ for all $0 < \varepsilon' < \varepsilon$.

Assume that CIRCUIT SAT can always be solved in $k^{1-\varepsilon}m^{1+o(1)}$ time and $m^{o(1)}$ space. Letting $k = m = n^2$, we have that $L \in \mathsf{NTIME}[n^2]$ implies $L \in \mathsf{DTISP}[n^{4-2\varepsilon}, n^{o(1)}]$ by Theorem 3.1 (the strong completeness theorem for SAT).

We now describe a nondeterministic algorithm for L that runs in $n^{2-\varepsilon+o(1)}$ time (a contradiction). Let A be an algorithm for L running in $n^{4-2\varepsilon}$ time and $n^{o(1)}$ space. Start by rejecting if the input is not unary. Suppose the input is 1^n.

Similar to the Speedup Lemma (Lemma 4.1), nondeterministically guess a list of $n^{2-\varepsilon}$ configurations of $A(1^n)$ where the first configuration is the initial configuration and the last configuration is an accepting configuration. Build a circuit with $(2-\varepsilon)\log n$ input variables, which "selects" an adjacent pair out of the $n^{2-\varepsilon}$ configurations. This pair is then fed to another circuit, which given a pair of configurations of $A(1^n)$, outputs 1 iff the first configuration does not lead to the second configuration within $n^{2-\varepsilon}$ steps of running A. To simulate A on the input 1^n, store the length of the input n in a batch of $O(\log n)$ wires W. Then on any simulated step, the circuit just needs to check if the current position of the input head (or, the current input register being read) is greater than the value in W. If it is, the input symbol is *blank*, otherwise it is 1. The circuit only has to carry the current configuration and these wires W throughout the simulation of $n^{2-\varepsilon}$ steps. Without loss of generality, we may assume that A simply iterates through its working storage cell by cell in a predictive way, adding a multiplicative $n^{o(1)}$ to the runtime. The upshot is that each step of $A(1^n)$ can be simulated by a circuit of $n^{o(1)}$ size. The circuit size overall is $n^{2-\varepsilon+o(1)}$ and has $(2-\varepsilon)\log n$ inputs. Observe that the circuit is *unsatisfiable* if and only if the configuration list is part of a valid accepting computation history for $A(1^n)$.

Applying the assumed Circuit SAT algorithm to this circuit, we obtain a nondeterministic algorithm for L which guesses $n^{2-\varepsilon+o(1)}$ bits then runs deterministically in $n^{2-\varepsilon+o(1)} \cdot O(\log n)^c$ time and $n^{o(1)}$ space. That is, the nondeterministic algorithm for L runs in $n^{2-\varepsilon}$ time. This contradicts our choice of L. □

6 Discussion

We have seen that universal improvements over exhaustive search, even marginal ones, would have surprising consequences in complexity theory. This connection between improved exponential algorithms and superpolynomial lower bounds shows that two communities have been implicitly working towards similar goals. Let us point to a few open problems that should produce further connections.

1. It seems possible that we may be able to prove interesting non-uniform lower bounds assuming only a $2^{.9n} \cdot \text{poly}(m)$ algorithm for CNF SAT on n variables and m clauses. Using the reduction from Theorem 3.5, one can show that if 3SAT is in $2^{\varepsilon n}$ time for all $\varepsilon > 0$, then we obtain mild circuit lower bounds.

Theorem 6.1 *If the Exponential Time Hypothesis is false, then* E^{NP} *does not have linear size circuits.*

Proof. (Sketch) If E^{NP} has $O(n)$ size circuits, then $\mathsf{NTIME}[2^n]$ has $O(n)$-size universal witness circuits. We can convert our nondeterministic machine into a single-tape one that uses $O(2^{2n})$ time and $O(2^n)$ space. Consider the verifier that on input x constructs an accepting tableau of $O(2^{3n})$ size for the single-tape machine, turns that into a 3CNF formula ϕ_x, then treats the witness as a satisfying assignment to ϕ_x. We can therefore simulate every $L \in \mathsf{NTIME}[2^n]$ on input x by guessing a circuit C on cn gates (for some c) that encodes a satisfying assignment for ϕ_x, then calling CIRCUIT SAT on a circuit D with $O(cn)$ wires and $3n + O(1)$ inputs, where $D(i)$ determines if the ith clause of ϕ_x is satisfied by the variable assignment encoded by C. In more detail, the set of $O(1)$ clauses corresponding to the (i,j) cell of an $O(2^{2n}) \times O(2^n)$ tableau can be computed with $O(n)$ size circuits which are given the indices i and j as input. This set of clauses is nearly identical for each cell, except in some places where i and j are added to the indices of some variables. However note that addition can be done with linear size circuits.

If 3SAT is in $2^{\varepsilon n}$ time for all $\varepsilon > 0$, then the CIRCUIT SAT call can be simulated in $2^{\varepsilon n}$ time for all $\varepsilon > 0$, regardless of the value of c. This would imply $\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[2^{\varepsilon n}]$ for all $\varepsilon > 0$, a contradiction. □

2. A very interesting open question (asked by Russell Impagliazzo [Imp10]) is if any converses to our results hold. For example, if $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$, could this lower bound be used to obtain faster SAT algorithms? Note it is known how to partially derandomize CAPP assuming $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ [IKW02].

3. It would be nice to weaken the algorithmic hypotheses even further. Can we replace CAPP with a form of *Polynomial Identity Testing* (PIT) in our results? The known proofs that "subexponential algorithms for PIT imply circuit lower bounds" go through Toda's theorem [Tod91], which prevents us from getting a tight simulation.

4. We were able to show that some problems with k-bit witnesses verifiable in $O(n)$ time cannot be solved in $O(\text{poly}(k) \cdot n)$ time and $\text{poly}(k, \log n)$ space. However we do not know if NP problems such as SAT have *linear* time algorithms when the space bound is relaxed. Such lower bounds are not even known for the PSPACE-complete QBF problem [Wil08a]. Perhaps progress can be made on these old questions by considering parameterized versions. Is it possible to prove unconditionally that QBFs with k quantifier blocks cannot be solved in $O(kn)$ time? The existence of such an algorithm would imply that SAT can be solved in linear time, and that QBF can be solved in quadratic time. It seems plausible that the conjunction of these two propositions can be more easily refuted.

5. Finally, can we prove unconditionally that there is *some* $k > 3$ satisfying

$$(\exists\, n)\mathsf{TISP}[n^k, n^{o(1)}] \not\subseteq (\exists\, O(n))\mathsf{TISP}[n^3, n^{o(1)}]?$$

Given our results, this separation would imply strong unconditional lower bounds for improving on exhaustive search with space-bounded verifiers. (Interestingly, such lower bounds may be achievable *without* proving class separations like $\mathsf{LOGSPACE} \neq \mathsf{NP}$.)

Acknowledgments

I am grateful to Russell Impagliazzo, Dick Lipton, Ken Regan, Mohan Paturi, and John Rogers for useful discussions on this work. Special thanks are due to V. Arvind, Andrew Drucker, Mike Fellows, and Rahul Santhanam for their valuable comments and corrections on a draft of this paper.

References

[ADF95] K. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogs. *Annals of Pure and Applied Logic* 73:235–276, 1995.

[AB09] S. Arora and B. Barak. *Computational Complexity – a modern approach*. Cambridge University Press, 2009.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential simulations unless EXPTIME has publishable proofs. *Computational Complexity* 3:307–318, 1993.

[Bar02] B. Barak. A probabilistic-time hierarchy theorem for slightly non-uniform algorithms. In *Proc. of RANDOM*, 194–208, 2002.

[BGHSV05] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proc. IEEE Conference on Computational Complexity*, 120–134, 2005.

[BBG98] S. Bloch, J. F. Buss, and J. Goldsmith. Sharply bounded alternation and quasilinear time. *Theory of Computing Systems* 31(2):187–214, 1998.

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proc. IEEE Conference on Computational Complexity*, 8–12, 1998.

[BG93] J. F. Buss and J. Goldsmith. Nondeterminism within P. *SIAM J. Computing* 22:560–572, 1993.

[CJ03] L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *J. Computer and System Sciences* 67(4):789–807, 2003.

[CC97] L. Cai and J. Chen. On the amount of nondeterminism and the power of verifying. *SIAM J. Computing* 26(3):733–750, 1997.

[CIP06] C. Calabro, R. Impagliazzo, and R. Paturi. A duality between clause width and clause density for SAT. In *Proc. IEEE Conference on Computational Complexity*, 252–260, 2006.

[CIP09] C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In *Proc. International Workshop on Parameterized and Exact Computation*, 2009.

[CCFHJKX06] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. A. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inofmration and Computation* 201:216–231, 2006.

[CHKX06] J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *J. Computer and System Sciences* 72:1346–1367, 2006.

[CG07] Y. Chen and M. Grohe. An isomorphism between subexponential and parameterized complexity theory. *SIAM J. Computing* 37:1228–1258, 2007.

[DH08] E. Dantsin and E. A. Hirsch. Worst-case upper bounds. In *Handbook of Satisfiability*, A. Biere, M. Heule, H. van Maaren and T. Walsh (eds.), 341–362, 2008.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, 1999.

[FL89] M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. In *Proc. ACM Symposium on Theory of Computing*, 501–512, 1989.

[FGW06] J. Flum, M. Grohe, and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. *J. Computer and System Sciences* 72(1):34–71, 2006.

[FG06] J. Flum and M. Grohe. *Parameterized complexity theory*. Springer, 2006.

[For01] L. Fortnow. Comparing notions of full derandomization. In *Proc. IEEE Conference on Computational Complexity*, 28–34, 2001.

[FLvMV05] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *JACM* 52(6):835–865, 2005.

[GO95] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications* 5(3):165–185, 1995.

[GS89] Y. Gurevich and S. Shelah. Nearly linear time. *Logic at Botik '89*, Springer-Verlag LNCS 363, 108–118, 1989.

[Har89] J. Hartmanis. Gödel, von Neumann, and the P?=NP problem. *Bulletin of the European Association for Theoretical Computer Science* 101–107, June 1989. See also: M. Sipser. The history and status of the P versus NP question. In *Proc. ACM Symposium on Theory of Computing*, 603–618, 1992.

[Has98] J. Hastad. The shrinkage exponent of De Morgan formulae is 2. *SIAM J. Computing*, 27:48–64, 1998.

[Imp10] R. Impagliazzo, personal communication.

[IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time versus probabilistic polynomial time. *J. Computer and System Sciences* 65(4):672–694, 2002.

[IP01] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Computer and System Sciences* 62(2):367–375, 2001.

[IPZ01] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Computer and System Sciences* 63(4):512–530, 2001.

[KC99] V. Kabanets and J.-Y. Cai. Circuit minimization problem. In *Proc. ACM Symposium on Theory of Computing*, 73–79, 2000.

[KRC00] V. Kabanets, C. Rackoff, and S. A. Cook. Efficiently approximable real-valued functions. *Electronic Colloquium on Computational Complexity*, TR00-034, 2000.

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

[Kan84] R. Kannan. Towards separating nondeterminism from determinism. *Mathematical Systems Theory* 17(1):29–45, 1984.

[KLV03] G. Karakostas, R. J. Lipton, and A. Viglas. On the complexity of intersecting finite state automata and NL versus NP. *Theoretical Computer Science* 302(1-3):257–274, 2003.

[KL80] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. ACM Symposium on Theory of Computing* 302–309, 1980.

[KF77] C. M. R. Kintala and P. C. Fisher. Computations with a restricted number of nondeterministic steps (Extended Abstract). In *Proc. ACM Symposium on Theory of Computing*, 178–185, 1977.

[KvM99] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Computing* 31(5):1501–1526, 2002.

[Koz77] D. Kozen. Lower bounds for natural proof systems. In *Proc. IEEE Symposium on Foundations of Computer Science*, 254–266, 1977.

[Lip09a] R. J. Lipton. *Simulation of nondeterministic machines*. Weblog post available at `http://rjlipton.wordpress.com/2009/05/11/simulation-of-nondeterministic-machines/`

[Lip09b] R. J. Lipton. *An approach to the P=NP question?* Weblog post available at `http://rjlipton.wordpress.com/2009/10/02/an-approach-to-the-pnp-question/`

[LV96] M. Luby and B. Velickovic. On deterministic approximation of DNF. *Algorithmica* 16(4/5):415–433, 1996.

[vMS05] D. van Melkebeek and R. Santhanam. Holographic proofs and derandomization. *SIAM J. Computing*, 35:59–90, 2005.

[vM07] D. van Melkebeek. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science* 2:197303, 2007.

[Mei09] O. Meir. Combinatorial PCPs with efficient verifiers. In *Proc. IEEE Symposium on Foundations of Computer Science*, 463–471, 2009.

[PW10] M. Patrascu and R. Williams. On the possibility of faster SAT algorithms. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[PF79] N. Pippenger and M. J. Fischer. Relations among complexity measures. *JACM* 26(2):361–381, 1979.

[SFM78] J. Seiferas, M. J. Fischer, and A. Meyer. Separating nondeterministic time complexity Classes. *JACM* 25:146–167, 1978.

[Sip05] M. Sipser. Introduction to the theory of computation. *Course Technology*, 2005.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Computing* 20:865–877, 1991.

[Tou01] I. Tourlakis. Time-space tradeoffs for SAT on nonuniform machines. *J. Computer and System Sciences* 63(2):268–287, 2001.

[Tra08] P. Traxler. The time complexity of constraint satisfaction. In *Proc. International Workshop on Parameterized and Exact Computation*, 190–201, 2008.

[Tse68] G. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constr. Math. and Math. Logic*, 1968.

[Wil08a] R. Williams. Non-linear time lower bound for (succinct) quantified Boolean formulas. *ECCC Technical Report 08-076*, August 2008.

[Wil08b] R. Williams. Time-space lower bounds for counting NP solutions modulo integers. *Computational Complexity* 17(2):179–219, 2008.

[Wil10] R. Williams. Alternation-trading proofs, linear programming, and lower bounds. In *Proc. Symposium on Theoretical Aspects of Computer Science*, 2010.

[Zak83] S. Zak. A Turing machine hierarchy. *Theoretical Computer Science* 26:327–333, 1983.

[Zan98] F. Zane. *Circuits, CNFs, and satisfiability.* Ph.D. Thesis, University of California at San Diego, 1998.

A Appendix: Proof of Lemma 3.1

Here we show that if $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, then every language in NEXP has universal witness circuits. The argument is analogous to that used in the proof that $\mathsf{NEXP} \subseteq \mathsf{P/poly} \Longrightarrow \mathsf{NEXP} = \mathsf{MA}$ [IKW02].

Proof. First we show that the absence of witness-producing circuits for NEXP implies a faster nondeterministic simulation of MA. This is combined with the fact that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ implies an MA-simulation of NEXP, to yield a contradiction with a time hierarchy theorem.

Suppose there is an $L \in \mathsf{NEXP}$ which does not have universal witness circuits. Let $c > 0$ be such that $L \in \mathsf{NTIME}[2^{n^c}]$. Then there is *some* correct verifier V for L such that for all constants $d \geq 1$, there is an infinite sequence of inputs $S = \{x_{i_k}\}$ with the properties:

- for all k, $x_{i_k} \in L$ and
- for all sufficiently large k and all y of length $2^{|x_{i_k}|^c}$, $V(x_{i_k}, y) = 1$ implies that the circuit complexity of the function $f(x_{i_k}, i) = y_i$ is greater than $|x_{i_k}|^d$.

Work of Babai *et al.* [BFNW93] and Klivans-Van Melkebeek [KvM99] shows the following *hardness-randomness connection*: for every $\varepsilon > 0$ there is a $\delta < \varepsilon$ and integer e such that, given random access to a Boolean function on n^δ variables with circuit complexity at least $n^{\delta e}$, there is a pseudorandom generator $G : \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ computable in $2^{O(n^\varepsilon)}$ time which fools circuits of size n.

We can simulate MA infinitely often in nondeterministic time $O(2^n)$ with n bits of advice, as follows. Let P be an MA protocol, and suppose the computation of Arthur in P (given Merlin's string) can be simulated by a circuit of size n^a. For each input length n, we set the advice for n-bit inputs to be the string $x_{i_\ell} \in S$, where x_{i_ℓ} has length $n^{\varepsilon a}$; if there is no such string, we set the advice to 0^n.

On an input x of length n with advice x_{i_k}, the MA simulation first nondeterministically guesses a witness y of length $2^{|x_{i_k}|^c}$ for the verifier V, and checks that $V(x_{i_k}, y) = 1$ (if this is not the case, the simulation rejects). Note by our choice of advice, this step takes $O(2^{n^{\varepsilon ac}})$ time. Next, we nondeterministically guess Merlin's polynomial length string in the MA simulation. Finally, we simulate Arthur by evaluating G on all $n^{\varepsilon a}$ possible seeds (treating the string y as a hard function), evaluating the circuit for Arthur on the outputs of G, and taking the majority answer.

We claim the generator G fools Arthur. On the $n^{\varepsilon a}$ length input x_{i_k}, the string y can be treated as a Boolean function on $n^{\varepsilon ac}$ variables with circuit complexity at least $n^{\varepsilon ad}$. We can make $n^{\varepsilon ad} \geq n^{\varepsilon a \delta e}$ since we can set d to be arbitrarily large. Hence G armed with y can fool circuits of size $n^{\varepsilon ad/(\delta e)} \geq n^a$, by the hardness-randomness connection. That is, G fools circuits of size n^a, and therefore Arthur as well.

The total running time of the simulation is $O(2^{n^{c\varepsilon a}} + 2^{n^{\varepsilon a}})$. Setting $\varepsilon > 0$ to be arbitrarily small (note it is independent of c and a), we have established $\mathsf{MA} \subseteq i.o. - \mathsf{NTIME}[2^{n^{\varepsilon'}}]/n^{\varepsilon'}$ for all $\varepsilon' > 0$. Since $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, there is a fixed constant q such that $\mathsf{NTIME}[2^n]/n$ has circuits of size $O(n^q)$. So MA has infinitely many input lengths that can be computed by a circuit family of $O(n^q)$ size. (In the language of complexity classes, we have $\mathsf{MA} \subseteq i.o. - \mathsf{SIZE}[O(n^q)]$.)

Now, $\mathsf{EXP} \subseteq \mathsf{P/poly}$ implies that $\mathsf{EXP} = \mathsf{MA}$, by Babai, Fortnow, Nisan, and Wigderson [BFNW93]. Hence there is a $c' > 0$ such that

$$\mathsf{EXP} = \mathsf{MA} \subseteq i.o. - \mathsf{NTIME}[2^n]/n \subseteq i.o. - \mathsf{SIZE}[O(n^q)].$$

However, the above is false, by a simple diagonalization argument. □

B Appendix: Proof of Theorem 1.9

Reminder of Theorem 1.9 *For every multitape Turing machine M running in time t and accessing a tape of nondeterministic bits, there is a $\delta > 0$ and a deterministic Turing machine that can simulate M in $O(2^{(1-\delta)t})$ time.*

The idea is to perform a brute-force simulation of M up to $(1-\delta)t$, for an appropriate $\delta > 0$. This generates $2^{(1-\delta)t}$ different configuration states of M. Observe that for a k-tape machine, there are only $2k\delta t$ different tape cells that could possibly be accessed in δt steps. Hence we can "compress" these configurations so that only $c^{2k\delta t}$ configurations remain for some $c > 0$, and the necessary information to complete the simulation is still preserved. Then we simulate the remaining configurations for δt steps to see if any accept. We can choose δ so that these procedures all take less than 2^t time.

Proof. Let $\delta > 0$ be a parameter. Let σ be the cardinality of the tape alphabet of M, and let k be the number of tapes of M. For all possible strings y of $(1-\delta)t$ bits, simulate M for its first $(1-\delta)t$ steps, assuming the nondeterministic tape has y written on it. For each y, save the *configuration* C_y of M after $(1-\delta)t$ steps, where a configuration consists of all tape cells that are within δt cells of some tape head. For each of the k tapes, there are $2\delta t + 1$ such cells. Note that for the nondeterministic tape in C_y, some tape cells may be *undetermined*, as they refer to bits of the nondeterministic tape that have not yet been read by M. We denote those cells with a fresh symbol s that is not already in the alphabet of M.

Note the total number of such configurations is $(\sigma+1)^{k(2\delta t+1)}$. Hence in order to remember all possible configurations of M, it suffices to store a bit array A of only $(\sigma+1)^{k(2\delta t+1)}$ size.

Now for every configuration C marked in array A, and for all $2^{2\delta t}$ ways to fill in the undetermined symbols of the nondeterministic tape in C with $2\delta t$ bits, simulate M for δt steps. If any simulation of M results in acceptance, then accept.

To optimize the running time, we want to set δ to minimize $2^{(1-\delta)t} + 2^{2\delta t}(\sigma+1)^{k(2\delta t+1)}$. (Note that polynomial factors in t do not matter in the optimization, as they can be subsumed by simply adding an ε factor to the exponents.) Routine calculation shows that $\delta = 1/(3 + 2k\log(\sigma+1))$ suffices. □

C A Remark About Barriers

Certainly, a slightly faster Circuit SAT or CAPP algorithm will need significantly new ideas. Let us remark why this strategy may be viable for proving lower bounds. Any potential approach to proving strong lower bounds must pass certain litmus tests that complexity theory has developed. The three primary barriers are relativization, natural proofs, and algebrization. We believe our work may provide a path that circumvents all three (provided the appropriate algorithms exist, of course!). The natural proofs barrier does not apply due to our use of diagonalization, which avoids the "largeness" of natural proofs. We believe the relativization and algebrization barriers would be avoided, because *all nontrivial SAT algorithms that we know do not relativize or algebrize*. That is, the interesting SAT algorithms in the literature cannot be used to solve SAT^A when the algorithm is given oracle access to an arbitrary oracle A (or its algebraic extension $\tilde{A}$).[15]

[15] Recall that SAT^A is the problem of satisfying CNF formulas with clauses of the form $(\ell_1 \vee \cdots \vee \ell_k \vee A(\ell_{k+1}, \ldots, \ell_{k+k'}))$, where the ℓ_i are literals, and the predicate $A(\ell_{k+1}, \ldots, \ell_{k+k'})$ is true if and only if $\ell_{k+1} \cdots \ell_{k+k'} \in A$. It is the natural variant on SAT obtained by applying the Cook-Levin theorem to a language in NP^A.

We invite the reader to examine their favorite (nontrivial) SAT algorithm to understand why it does not extend to SAT^A. As a simple example, consider the branching algorithm which looks for clauses of length one, and sets the variable in that clause accordingly. (If there is no length-one clause, say it branches on an arbitrary variable.) In an instance of SAT^A, a length-one clause may have the form $A(\ell_{k+1}, \ldots, \ell_{k+k'})$, with possibly some 0-1 values substituted for some of the literals. Determining a correct assignment for the variables in this predicate requires at least an oracle for "Satisfiability of A." More precisely, we would need to efficiently solve the problem: *given a pattern string $p \in \{0, 1, \star\}^n$, determine if there is a binary assignment to the $\star$'s in p which yields a string in A.* This would require *nondeterministic* access to the oracle A. Furthermore, if the A-predicate has more than one satisfying assignment, then the correct assignment to A may simply be *underdetermined*. Access to the algebraic extension $\tilde{A}$ would not suffice for simulating such queries to A.

Given the above, we believe it is not outrageous to think that one might design slightly better CIRCUIT SAT or CAPP algorithms that circumvent all the known lower bound barriers in complexity theory.