INDUCTIVE TIME-SPACE LOWER BOUNDS FOR SAT AND RELATED PROBLEMS

Ryan Williams

Abstract. We improve upon indirect diagonalization arguments for lower bounds on explicit problems within the polynomial hierarchy. Our contributions are summarized as follows.

1. We present a technique that uniformly improves upon most known nonlinear time lower bounds for nondeterminism andalternating computation, on both subpolynomial ($n^{o(1)}$) space RAMs and sequential one-tape machines with random access to the input. We obtain improved lower bounds for Boolean satisfiability (SAT), as well as all NP-complete problems that have efficient reductions from SAT, and Σ_k-SAT, for constant $k \geq 2$. For example, SAT cannot be solved by random access machines using $n^{\sqrt{3}}$ time and subpolynomial space.

2. We show how indirect diagonalization leads to time-space lower bounds for computation with *bounded* nondeterminism. For both the random access and multitape Turing machine models, we prove that for all $k \geq 1$, there is a constant $c_k > 1$ such that linear time with $n^{1/k}$ nondeterministic bits is not contained in deterministic n^{c_k} time with subpolynomial space. This is used to prove that satisfiability of Boolean circuits with n inputs and n^k size cannot be solved by deterministic multitape Turing machines running in $n^{k \cdot c_k}$ time and subpolynomial space.

Keywords. time-space tradeoffs, lower bounds, polynomial-time hierarchy, satisfiability, diagonalization, bounded nondeterminism

Subject classification. 68Q17

1. Introduction

We study the power of indirect diagonalization in proving class separations and lower bounds on explicit problems in NP and the polynomial hierarchy,

deriving several improvements on existing lower bounds and some brand new bounds as well.

1.1. Indirect Diagonalization. Put bluntly, a separation by "indirect diagonalization" is a proof-by-contradiction simulation. Suppose we wish to show $\mathcal{C} \not\subseteq \mathcal{D}$, for classes $\mathcal{C}$ and $\mathcal{D}$. An indirect diagonalization argument begins by assuming $\mathcal{C} \subseteq \mathcal{D}$. If sufficiently strong, this assumption allows us to derive new complexity class inclusions— in particular, we use the algorithms guaranteed to exist by the assumption to develop new algorithms. After some iterations, the new algorithms/inclusions become so wonderful that they are not only unlikely but are also provably impossible, contradicting a known separation result, *e.g.*, a time hierarchy theorem. As a lower bound method, this approach is appealing in that it allows one to employ one's algorithmic intuitions towards the task of proving lower bounds, and it is non-relativizing, in the sense that one can potentially invoke non-relativizing inclusions in the derivations that lead to a contradiction.

One limitation to this sort of attack is that, at present, there are not too many known class separations to begin with, so a number of strong assumptions are sometimes necessary to reach a contradiction. The major goal of our work is to investigate how one might circumvent this difficulty, by employing a host of known class separations in some sophisticated way. In this paper, we focus on the particular cases of lower bounds on nondeterministic linear time, alternating linear time, and bounded nondeterminism. The lower bounds for nondeterministic linear time extend to lower bounds for natural NP-complete problems, such as Boolean satisfiability. Similarly, the lower bounds for alternating linear time extend to lower bounds for a class of quantified Boolean formulas, and the bounded nondeterminism results imply a lower bound for circuit satisfiability.

1.2. A Four-Step Scheme for Indirect Diagonalization. Let $\mathcal{D}[t]$ denote a class of sets recognized by some deterministic machine model running in time t. One family of prior lower bound approaches for nondeterministic time (in general, alternating time) follows a certain high-level schematic:

1. Assume (for contradiction) that $\mathsf{NTIME}[n] \subseteq \mathcal{D}[n^c]$.
2. Prove that $\mathcal{D}[t]$ can be "sped up" by alternating machines. More precisely, $\mathcal{D}[t] \subseteq \Sigma_\ell \mathsf{TIME}[f(t)]$ for some $\ell \geq 1$ and $f(n) = o(n)$.
3. Prove using (1) that alternations in an alternating computation can be "removed", at the cost of a small "slowdown" in time. For example, (1)

might imply $\Sigma_2\mathsf{TIME}[n] \subseteq \mathsf{NTIME}[n^c]$, by converting the conondeterministic part of the Σ_2 computation into a deterministic computation.

4. If the amount of speedup in (2) sufficiently exceeds the amount of slowdown in (3), conclude a contradiction to some known time hierarchy theorem.

Introduced in a paper by Kannan (12) in 1983 (he used it to prove time lower bounds on one-tape Turing machines), this four-step scheme has been quite successful, leading to a number of lower bounds on nondeterminism and alternation in several machine models over the years (6; 7; 14; 15; 20; 22; 25). For example, the celebrated result of Paul, Pippenger, Szemeredi, and Trotter (20) that $\mathsf{NTIME}[n] \neq \mathsf{DTIME}[n]$ for multitape machines can be said to follow the above:

1. Assume $\mathsf{NTIME}[n] = \mathsf{DTIME}[n]$.
2. Paul-Pippenger-Szemeredi-Trotter prove $\mathsf{DTIME}[t] \subseteq \Sigma_4\mathsf{TIME}[t/\log^* t]$, for $t(n) \geq n \log^* n$.
3. Item (1) implies that $\Pi_k\mathsf{TIME}[n] = \mathsf{coNTIME}[n] = \mathsf{DTIME}[n]$ for all k.
4. Therefore by padding, $\Pi_4\mathsf{TIME}[t] = \mathsf{DTIME}[t] \subseteq \Sigma_4\mathsf{TIME}[t/\log^* t]$, a contradiction with the alternating time hierarchy.

We extend the above four-step argument in a significant way, leading to improved lower bounds in several computational models.

1.3. Lower Bounds for SAT on Subpolynomial-Space RAMs. Rooted in work of Kannan (12; 13) from the early 80's, and initiated by Fortnow (6) in 1997, an intriguing thread of lower bound research has opened up that seeks to prove a "poor man's" version of $\mathsf{L} \neq \mathsf{NP}$ using indirect diagonalization.

More precisely, while it is well-known that $\mathsf{L} \neq \mathsf{NP}$ is equivalent to the statement "nondeterministic linear time is not contained in deterministic n^k time on logspace machines, for all $k \geq 1$" (*i.e.*, $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^k, \log n]$), proving such a large lower bound on nondeterministic linear time currently appears out of our reach. Nevertheless, it is of course still interesting to ask what we *can* prove about the matter—to find the largest k for which the separation provably holds. Naturally, when one starts to consider fixed time bounds, the model of computation becomes a possible issue. We use the random access Turing machine model discussed in Section 2.1, which is time-equivalent to other

random access models within polylogarithmic factors. SAT has strong completeness properties under this model, in the sense that a separation of the form $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^k, \log n]$ implies that SAT is not in $\mathsf{DTISP}[n^{k-o(1)}, \log n]$. Therefore, proving such a non-containment also proves a lower bound for an explicit and natural problem in NP.

Lipton and Viglas (14) gave an indirect diagonalization argument (which implicitly follows the four-step scheme) to obtain the lower bound

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^{\sqrt{2}-\varepsilon}, n^{o(1)}].$$

By Corollary 2.2 for subpolynomial (*i.e.* $n^{o(1)}$) space RAMs, this implies that SAT cannot be solved in $n^{\sqrt{2}-\varepsilon}$ time and subpolynomial space, for all $\varepsilon > 0$. The best lower bound known prior to our work was $n^{\phi-\varepsilon}$ time and $n^{o(1)}$ space by Fortnow and Van Melkebeek (7) in 2000, where $\phi \approx 1.618$ is the golden ratio. Their proof was also an indirect diagonalization.

Improvements on Deterministic Subpolynomial-Space SAT Lower Bounds. We utilize tools developed in the aforementioned work with an inductive method to improve the SAT time lower bound to $\Omega(n^{1.6616\ldots})$ time.

THEOREM 1.1. *SAT is not solvable by deterministic RAMs using $n^{1.6616}$ time and $n^{o(1)}$ space. The lower bound holds for any NP-complete problem that is (simultaneously) quasi-linear time and polylogspace reducible from SAT, where each bit in the reduction is computable in $n^{o(1)}$ time.*

An informal outline of the inductive method is provided in Section 1.7, and the theorem is proved in Section 3.2. We then boost the lower bound to $n^{1.7327}$, which is slightly larger than $n^{\sqrt{3}}$.

THEOREM 1.2. *SAT is not solvable by deterministic RAMs using $n^{1.7327}$ time and $n^{o(1)}$ space.*

To achieve this bound, we introduce a new tool that *improves upon item (2)* in the four-step scheme. That is, we give an improved speedup of DTISP in $\Sigma_2\mathsf{TIME}$. The key behind our speedup is that it relies heavily on item (1), *i.e.* the initial assumption of the indirect diagonalization argument. Previous speedup results of this kind (Lipton-Viglas and Fortnow-Van Melkebeek) did not utilize this assumption— their containments of DTISP in $\Sigma_k\mathsf{TIME}$ hold unconditionally.

1.4. Improvement on Alternating Time Lower Bounds. We also improve upon lower bounds for solving Σ_k-SAT (deciding the truth of Σ_k sentences in first-order Boolean logic) with deterministic and co-nondeterministic machines, by using the inductive method introduced in the $n^{1.661}$ lower bound for SAT. In their golden-ratio lower bound work, Fortnow and Van Melkbeek (7) also proved that the problem Σ_k-SAT cannot be solved in $n^{k-\varepsilon}$ time on a $n^{o(1)}$-space random access machine. We improve upon this result for all values of k. For instance, in the case of Σ_2-SAT, the lower bound goes from n^2 time to $n^{2.761}$. As k increases, the lower bound becomes closer to n^{k+1}: for example, the bound for Σ_{100}-SAT is $n^{100.99}$ time. The following lower bound for all Σ_ℓ is proved in Section 4.1.

COROLLARY 1.3. $\Sigma_\ell\mathsf{TIME}[n] \not\subseteq \mathsf{DTISP}[n^{\ell(1+1/\ell)^{1/2}}, n^{o(1)}]$.

1.5. Improved Lower Bounds for SAT on Off-Line One-Tape TMs. The above results all hold for random-access machine models. Our inductive method also works for lower bounds on a machine model that is a hybrid between a RAM and a off-line one-tape TM: in particular, the model has random access to its input, but only sequential access to its worktape. For more details on the model, see Section 2.1.

Proving time lower bounds on these machines is not as easy as one might first think, *e.g.* such machines can recognize PALINDROMES in linear time and logarithmic space.[1] Maass and Schorr (15) and Van Melkebeek and Raz (16) independently showed the following result for off-line one-tape TMs, where $\mathsf{DTIME}_1[t]$ denotes the time class for this machine model:

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTIME}_1[n^{1.22}].$$

Theorem 1.4 increases this lower bound by a modest amount.

THEOREM 1.4. $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTIME}_1[n^{1.268}]$.

COROLLARY 1.5. *SAT is not solvable by any hybrid TM that runs in* $O(n^{1.268})$ *time.*

[1] Contrast this machine model with the standard multitape Turing machine, where a $O(\log n)$ space bound implies a $\Omega(n^2)$ time lower bound for recognizing PALINDROMES (4). Santhanam (22) gave an efficient reduction from PALINDROMES to SAT, resulting in an $\Omega(n^2/(poly(\log n)))$ time lower bound on the time-space product for solving SAT on multitape machines.

1.6. Lower Bounds for Bounded Nondeterminism on Small-Space Machines. In the remainder of the paper, we show how indirect diagonalization ideas using alternation can be further extended to prove lower bounds on *bounded* nondeterministic computation. Define $\mathsf{NTIBI}[t(n), b(n)]$ to be the class of languages recognized by $t(n)$ *time* (the TI) random access Turing machines that use at most $b(n)$ nondeterministic *bits* (the BI). More precisely, when given an input x, a characteristic machine for this class guesses $b(|x|)$ bits on a special tape and then runs deterministically for $t(|x|)$ time using the input tape, the special tape, and some number of worktapes.[2] We prove the separation:

THEOREM 1.6. *For all $\varepsilon > 0$, there is a $c_\varepsilon > 1$ such that $\mathsf{NTIBI}[n, n^\varepsilon] \not\subseteq \mathsf{DTISP}[n^{c_\varepsilon}, n^{o(1)}]$. The theorem also holds when the classes are defined with respect to multitape Turing machines.*

That is, even with only $n^{1/1000}$ nondeterministic moves, there is still a nonlinear time separation of nondeterminism from deterministic small space. Some examples of concrete values of c_ε are:

- $\mathsf{NTIBI}[n, n^{\frac{1}{\sqrt{2}}-\varepsilon}] \not\subseteq \mathsf{DTISP}[n^{1.414}, n^{o(1)}]$
- $\mathsf{NTIBI}[n, n^{0.451}] \not\subseteq \mathsf{DTISP}[n^{1.25}, n^{o(1)}]$
- $\mathsf{NTIBI}[n, n^{0.0199}] \not\subseteq \mathsf{DTISP}[n^{1.01}, n^{o(1)}]$

As might be expected, $c_\varepsilon \to 1$ as $\varepsilon \to 1$. Theorem 1.6 has an interesting corollary.

COROLLARY 1.7. *For all $k \geq 1$, there exists $c_k > 1$ such that Boolean satisfiability on circuits with n inputs and n^k gates requires $n^{k \cdot c_k}$ time on a deterministic multitape Turing machine using $n^{o(1)}$ space.*

Remark on Derivations. While the lower bound exponents in the Lipton-Viglas and Fortnow-Van Melkebeek SAT lower bounds have simple, symbolic formulations, the best representations we know for our exponents are rather complicated expressions, whose complexity increases with each inductive step. As a consequence, the values we state for these exponents were obtained by computer analysis. We do not yet know nice closed-form expressions for the exponents we have derived.

[2] Note $\mathsf{NTIBI}[t, b] = \mathsf{GC}[b, \mathsf{DTIME}[t]]$, where GC is the guess-and-check model of Cai and Chen (1). However, we found the NTIBI notation more convenient for our purposes.

1.7. Our Strategy at a High Level. We propose an inductive strategy which elaborates upon the four-step scheme described in the introduction, and takes better advantage of the polynomial hierarchy in obtaining a contradiction.

The main idea is to derive a sequence of "switching lemma" style inclusions (but only vaguely related to Håstad (10)), where each new lemma invokes all of the previous lemmas in its derivation. We start by using items (2) and (3) from the four-step scheme (*i.e.* "speedup" and "alternation removal") to derive

$$\Sigma_2\mathsf{TIME}[n] \subseteq \Pi_2\mathsf{TIME}[n^{f_2}]$$

for some small constant f_2. Essentially this means that an "OR of ANDs" at the bottom of the configuration tree of an alternating machine can be switched with an "AND of ORs" of polynomial size. Under the right conditions, this switching lemma can be invoked to prove an even better relationship between Σ_3 and Π_3, namely

$$\Sigma_3\mathsf{TIME}[n] \subseteq \Pi_3\mathsf{TIME}[n^{f_3}]$$

for some $f_3 < f_2$. In general, a relation between Σ_k and Π_k can be proved by using relations between all lower levels of the polynomial hierarchy. If the derived exponent f_k ever drops below 1, then we know the hypothesis assumed in item (1) of the four-step scheme must be false, as that contradicts a time hierarchy theorem.

Let's give a concrete illustration. In the deterministic random-access machine lower bounds for SAT, we suppose that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ for some $c > 1$ and in turn we derive an inclusion of the form

$$\Sigma_2\mathsf{TIME}[n] \subseteq \Pi_2\mathsf{TIME}[n^{f_2(c)}],$$

where f_2 is a function of c. If $f_2(c) < 1$, then the inclusion contradicts a known separation (cf. Theorem 2.4 in the following section). Otherwise, we may assume this inclusion as a *lemma*, to help us out in further derivations.[3] More precisely, we use this inclusion to derive $\Sigma_3\mathsf{TIME}[n] \subseteq \Pi_3\mathsf{TIME}[n^{f_3(c)}]$, where $f_3(c) < f_2(c)$ for appropriate c. Again if $f_3(c) < 1$ we are done, otherwise we proceed to get an inclusion for $\Sigma_4\mathsf{TIME}[n]$, $\Sigma_5\mathsf{TIME}[n]$, *etc.* Our construction and choice of c ensure that the sequence $f_2(c)$, $f_3(c)$, $f_4(c)$, ... is monotonically decreasing, and eventually drops below 1. Moreover, the value of c such that the sequence drops below 1 is larger than the lower bound exponents previously obtained. More dramatic improvements occur with higher levels of the polynomial hierarchy, cf. Table 1.1.[4]

[3] In fact, observe $f_2(c) = 1$ would imply $\Sigma_k\mathsf{TIME}[n] = \Sigma_2\mathsf{TIME}[n]$ for all $k \geq 2$. One can show that this would be also sufficient for a contradiction.

[4] We regrettably remark that the table in the conference version (26) mistakenly reported

	Model of computation		
Problem	*det. RAM*	*co-nondet. RAM*	*hybrid TM (one worktape)*
SAT	$n^{1.7327}$ $(n^{1.618})$	$(n^{1.414})$	$n^{1.268}$ $(n^{1.224})$
Σ_2-SAT	$n^{2.788}$ (n^2)	$n^{1.6616}$	$n^{1.609}$ $(n^{1.5})$
Σ_3-SAT	$n^{3.826}$ (n^3)	$n^{2.39}$	$n^{1.726}$
Σ_{100}-SAT	$n^{100.99}$ (n^{100})	$n^{50.49}$	$n^{1.99}$

Table 1.1: **Time lower bounds of this paper.** For RAMs, the given bounds hold assuming at most $n^{o(1)}$ workspace is used. The 'hybrid TM' model is discussed in Section 2.1. The previous bounds are in parentheses; each of them either directly appear in (7) (the RAM bounds), (16) (the TM bounds), or can be easily derived from that work.

Comparison With Fortnow-Van Melkebeek. Our inductive strategy appears to be somewhat different from that of Fortnow and Van Melkebeek. Although both strategies use indirect diagonalization and induction, Fortnow and Van Melkebeek obtain contradictions by inductively deriving containments of the form

$$\mathsf{NTIME}[n^{k_i}] \subseteq \mathsf{coNTIME}[n^{a_i k_i}],$$

for an increasing sequence $\{k_i\}$ and decreasing sequence $\{a_i\}$. When $a_i < 1$, a contradiction to a time hierarchy theorem is reached. In contrast, our induction derives

$$\Sigma_k\mathsf{TIME}[n] \subseteq \Pi_k\mathsf{TIME}[n^{b_k}]$$

for a decreasing sequence $\{b_k\}$. That is, Fortnow-Van Melkebeek derives relations between NTIME and $\mathsf{coNTIME}$ for larger and larger time functions, while our method derives relations for the same time bounds but for a larger and larger number of alternations. Furthermore, when a contradiction does not hold for us, we still obtain a relation between Σ_k and Π_k that is useful for deriving a relation between Σ_{k+1} and Π_{k+1} and higher levels of the polynomial hierarchy. The dichotomy of deriving either (a) a contradiction, or (b) a better inclusion than before, is the leverage that allows us to improve the known lower bounds. In particular, item (b) provides a way to remove alternations at a lower time cost. Thus one may say that this kind of argument helps us *improve upon item (3)* in the four-step scheme above.

incorrect lower bounds for Σ_3 and Σ_{100} on co-nondeterministic machines, of $n^{2.761}$ and $n^{99.98}$. Those lower bounds were for Σ_2 and Σ_{99} on deterministic machines. We thank an anonymous reviewer for pointing this out.

We note that the final results in this paper (those proving lower bounds on bounded nondeterminism) do not follow the inductive strategy we have outlined above. Instead, they exploit a property of the known simulations of DTISP in $\Sigma_2\mathsf{TIME}$. In particular, these $\Sigma_k\mathsf{TIME}$ and $\Pi_k\mathsf{TIME}$ speedups of DTISP are such that the kth quantifier only guesses $O(\log n)$ bits. Roughly speaking, we do not need the assumption $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ to eliminate an $O(\log n)$-bit quantifier, as the weaker assumption $\mathsf{NTIBI}[n, \log n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ will suffice. For more details, confer with Section 6.

2. Preliminaries

This section has two parts. First, we describe our notation and the machine models we use in the paper. Secondly, we recall some results from past work that shall be useful in developing our lower bounds.

2.1. Notation and the Machine Models Studied. We use ε to denote (as is standard) a non-zero quantity that is sufficiently small for the current context. As is typical with most such works, we implicitly assume floors and ceilings are applied to fractions wherever appropriate.

We assume familiarity with basic complexity notions such as alternation (2), and standard resource-bounded classes such as $\mathsf{DTIME}[t]$, $\mathsf{NTIME}[t]$, $\mathsf{SPACE}[s]$, $\mathsf{DTISP}[t, s]$ (*simultaneous* deterministic time t and space s), $\mathsf{NTISP}[t, s]$ (the nondeterministic version of the same class), and $\Sigma_k\mathsf{TIME}[t]$ and $\Pi_k\mathsf{TIME}[t]$ (time with $k-1$ alternations— Σ denotes starting in an existential state, whereas Π-machines start in a universal state). We also use the class $\mathsf{NTIBI}[t, b]$ (nondeterministic time and bits), which we define to be the class of languages recognized by $t(n)$ *time* (the TI) random access Turing machines that use at most $b(n)$ nondeterministic *bits* (the BI).

When we speak of a "quantifier" of an alternating machine, we are referring to a segment of the machine's computation whose first timestep is either in the initial state or in a state immediately following an alternation, and whose last timestep is either in a final state or in a state immediately prior to an alternation. So, no alternations occur during a quantifier, but they can occur just before and after it. For example, an alternating machine M uses exactly k quantifiers iff M is a Σ_k or Π_k machine.

Default Model. Our default machine model is the random access Turing machine, although our arguments work for RAMs as well and in general are more or less model-independent. When we specify a class without further qual-

ification, we are referring to classes defined with respect to this Turing machine model. By "random access Turing machine", we mean Turing machines with a read-only input tape, a full-access work tape, and two write-only index tapes (one for input, one for work). To access the ith cell of the input or work tape, one writes i to the respective index tape; hence an arbitrary access of a tape with t cells takes $O(\log t)$ time. After the ith cell is accessed, the respective index tape is reset to blanks.

Hybrid Off-Line TM Model. We also prove lower bounds for an off-line Turing machine model, referred to as "deterministic off-line TM" in Table 1.1. The machine model has:

- an input tape that is read-only, random access,
- a small storage of $n^{o(1)}$ bits that is read-write, random access, and
- an unbounded one-dimensional tape that is read-write with sequential (two-way) access.

To emphasize that there is only one unbounded read-write tape, we define $\mathsf{DTIME}_1[t]$ to be the relevant time class for this machine model.

2.2. Existing Tools. As our results build upon previous lower bound work, we apply several tools already available. The first one permits us to phrase our results as lower bounds for SAT.

A Useful Property of Satisfiability. It is known that satisfiability of Boolean formulas in conjunctive normal form is a complete problem under very tight reductions for a small nondeterministic complexity class. Define NQL as "nondeterministic quasi-linear time", *i.e.*

$$\mathsf{NQL} := \bigcup_{c \geq 0} \mathsf{NTIME}[n \cdot (\log n)^c] = \mathsf{NTIME}[n \cdot \mathit{poly}(\log n)].$$

Theorem 2.1. (Follows from Schnorr (24), Cook (5), Gurevich and Shelah (9), Fortnow and Van Melkebeek (7), Tourlakis (25)) *SAT is* NQL*-complete, under reductions in quasi-linear time and* $O(\log n)$ *space simultaneously, for both multitape and random access machine models. Moreover, each bit of the reduction can be computed in* $O(\mathit{poly}(\log n))$ *time and* $O(\log n)$ *space in both machine models.*

This theorem has a corollary significant for our purposes. Let $\mathcal{C}[t(n)]$ in the following represent a time $t(n)$ complexity class under one of the three models:

- Deterministic RAM using time t and $t^{o(1)}$ space,
- Co-nondeterministic RAM using time t and $t^{o(1)}$ space,
- Hybrid Off-Line TM using time t.

COROLLARY 2.2. *If* $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[t(n)]$, *then there is a* $c > 0$ *such that SAT is not contained in* $\mathcal{C}[t(n) \cdot (\log t(n))^c]$.

That is, if one can show $\mathsf{NTIME}[n]$ is not in $\mathcal{C}[t]$, then one can name an *explicit problem* (SAT) that is not in $\mathcal{C}[t]$, modulo polylog factors. Our proofs establish that $\mathsf{NTIME}[n]$ is not in $\mathcal{C}[n^c]$ for particular $c > 1$, implying that SAT is not solvable in $n^{c-o(1)}$ time with respect to the particular machine model denoted by $\mathcal{C}$. For example, we shall show a lower bound of the form $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, which means that SAT is not in $\mathsf{DTISP}[n^{c-o(1)}, n^{o(1)}]$.

Furthermore, as observed by Van Melkebeek and Raz, the results of this work apply to any problem Π such that SAT reduces to Π under highly efficient reductions. Examples of such problems include VERTEX COVER, INDEPENDENT SET, TRAVELLING SALESPERSON, 3-SAT, and MAX-2-SAT. (The typical reductions from SAT to these problems use gadgets, where there is an explicit and simple correspondence between each clause of the original formula and each gadget of the reduced instance.)

COROLLARY 2.3. (CF. VAN MELKEBEEK AND RAZ (16)) *The lower bounds of this paper apply to any problem* Π *such that SAT reduces to* Π, *where each bit of the reduction is computable in* $n^{o(1)}$ *time in the machine model for which one is proving the lower bound.*

Some Separation Results. We shall also require some well-known separation results, each of which are provable by straightforward diagonalization. The following result uses the fact that a random-access machine using k quantifiers in time t can be simulated by a two-tape machine using k quantifiers in time $O(t)$, found in Chandra and Stockmeyer's original conference paper on alternation (3).

THEOREM 2.4. ("NO COMPLEMENTARY SPEEDUP") *For all $k \geq 1$ and time constructible $t(n) \geq n$, $\Sigma_k\mathsf{TIME}[t] \not\subseteq \Pi_k\mathsf{TIME}[o(t)]$.*

We call it the "No Complementary Speedup" Theorem, as it intuitively says that not all bounded-alternation machines can be sped up by a "complementary" machine with the same number of alternations.

When proving lower bounds on bounded nondeterminism classes, we use a different hierarchy theorem, which holds for random access and multitape Turing machines.

THEOREM 2.5. *For all time constructible functions $t_2(n), t_2(n) \geq n$ such that $t_2(n)\log t_2(n) = o(t_1(n))$, and for all $\varepsilon \in (0,1)$,*

$$\mathsf{coNTIBI}[t_1, t_1^\varepsilon] \not\subseteq \mathsf{NTIBI}[t_2, t_2^\varepsilon].$$

PROOF. Standard diagonalization. First, observe one can enumerate the set of random access (or multitape) machines $\{M_i\}$ using $t_2(n)$ time and $t_2^\varepsilon(n)$ nondeterministic bits in a standard way (perhaps the only difficulty in the proof is that $t_2^\varepsilon(n)$ is computable in $O(t_2(n))$ time). Define M' that on x determines M_x, universally guesses $t_1^\varepsilon(|x|)$ bits on a special tape, then simulates $M_x(x)$ with the special tape, returning the opposite answer. It takes $O(t_1^\varepsilon)$ time to write down the bits, and an arbitrary deterministic time t_2 machine can be simulated in $O(t_1)$ time. Since $M'(x) = \neg M_x(x)$, it is clear that if M_x is nondeterministic and M' is co-nondeterministic, then $M_x(x)$ accepts iff $M'(x)$ rejects. □

"Alternations For Time" Lemma. Another useful proposition says that we can reduce alternations in a computation while slightly increasing the runtime, provided that there is a close time relationship between classes with fewer alternations. We do not know of a reference for this lemma, but its proof is elementary.

LEMMA 2.6. (ALTERNATIONS FOR TIME LEMMA) *Let $d > 1$, let k and ℓ be non-negative integers with $k > \ell$, and let $t(n) \geq n$ be time constructible. If $\Sigma_\ell\mathsf{TIME}[n] \subseteq \Pi_\ell\mathsf{TIME}[n^d]$, then*

- $\Sigma_k\mathsf{TIME}[t] \subseteq \Sigma_{k-1}\mathsf{TIME}[t^d]$, *and*
- $\Pi_k\mathsf{TIME}[t] \subseteq \Pi_{k-1}\mathsf{TIME}[t^d]$.

PROOF. First, observe that $\Sigma_\ell\mathsf{TIME}[n] \subseteq \Pi_\ell\mathsf{TIME}[n^d]$ implies $\Pi_\ell\mathsf{TIME}[n] \subseteq \Sigma_\ell\mathsf{TIME}[n^d]$. So by padding, any k-quantifier machine M running in time t is

equivalent to some k-quantifier machine N that runs in time t^d, but if M begins with an $\exists$ (resp. $\forall$) quantifier, then N begins with a $\forall$ (resp. $\exists$) quantifier.

Let M be a Σ_k machine with $O(t)$ runtime. Without loss of generality, we may assume that the state of M at the start of the $(k-\ell)^{\text{th}}$ alternation is a special state q^*. Define a machine M^* whose input is an input x to M and a tape configuration C of M:

$$M^*(x, C): \quad \text{Simulate } M(x), \text{ starting from } C \text{ and state } q^*.$$

By assumption, M^* has ℓ quantifiers, since M has k quantifiers and q^* starts at the $(k-\ell)^{th}$ alternation (the beginning of the $(k-\ell+1)^{th}$ quantifier). M^* runs in $O(t)$ time and takes inputs of $O(t)$ size.

The hypothesis implies that there is a machine N^* that is equivalent to M^*, runs in $O(t^d)$ time, uses ℓ quantifiers, but begins each computation with the quantifier opposite to that with which M^* begins. We now define a machine N:

$N(x)$: Simulate $M(x)$ until q^* is reached.
Let C be the configuration of $M(x)$ at this point. Simulate $N^*(x, C)$.

It is easy to verify that $L(N) = L(M)$, and that N runs in $O(t^d)$ time. We claim that N uses only $k-1$ quantifiers. This follows from the fact that the last quantifier of $M(x)$ prior to q^* and the first quantifier of N^* are the *same*. Therefore, in N, no alternation occurs at state q^*. But N and M still have the same number of alternations occurring prior to q^* and after q^*, so N^* has one less alternation than M. □

Fortnow and Van Melkebeek's Speedup Simulations. We also require speedup simulations of DTISP of NTISP with alternating machines. In this paper, we build upon Fortnow and Van Melkebeek's simulations. Their proofs are crucial to some of our arguments, so we include sketches of them for completeness.

Lemma 2.7. (Fortnow and Van Melkebeek (7), Theorem 5.1) *For every natural number $k \geq 2$, and time constructible t, space constructible s, and $b(n)$ such that $1 \leq b(n) \leq t(n)$,*

$$\mathsf{DTISP}[t, s] \subseteq \Sigma_k\mathsf{TIME}[k \cdot b \cdot s + t/b^{k-1}].$$

In particular, the first $(k-1)$ quantifier stages run in $O(b \cdot s)$ time each, and the last quantifier stage runs in $O(t/b^{k-1})$ time.

Note the case $k = 2$ was essentially proved by Kannan (13). The key idea is to simulate the proof in (2) that $\mathsf{DTISP}[t, s] \subseteq \mathsf{ATIME}[s \log t]$, which is essentially Savitch's theorem (23) tailored to the language of alternating machines. In the proof of $\mathsf{DTISP}[t, s] \subseteq \mathsf{ATIME}[s \log t]$, the alternating simulation of a $\mathsf{DTISP}[t, s]$ machine M works by repeatedly guessing configurations "in the middle" of the computation. Let us think of the input to an alternating simulation A as a triple $\langle k, C, C' \rangle$, where k is a positive integer and C and C' are configurations of M. A wishes to output *yes* iff, when M is executed from C for 2^k steps, its configuration becomes C'. (Without loss of generality, the runtime t is a power of two.) To do this, if $k = 0$ then A just simulates M from C for one step, and checks if its configuration equals C'. For $k > 0$, A existentially guesses a configuration C'', which is supposed to be the configuration of M that occurs $2^{k-1} = 2^k/2$ steps after starting from C; let us call this configuration C''. A universal quantifier then guesses a 0 or a 1. Finally, A calls itself on $\langle C, C'', k-1 \rangle$ if 0 was written, and calls itself on $\langle C'', C', k-1 \rangle$ if 1 was written. It is easy to see that this simulation works; a little analysis shows that its runtime is $O(s \log t) = O(s^2)$.

The above machine A uses many alternations during its execution ($O(\log t)$, as a matter of fact). To obtain a fast simulation of M that uses a constant number of alternations, we can guess many configurations at once, and universally check each one we guessed. This sacrifices the $O(s \log t)$ runtime, but uses vastly fewer alternations.

Proof of Lemma 2.7. (Sketch) Fix a deterministic machine M using time $t(n)$ and space $s(n)$. We first show how to simulate M in $kbs + t/b^{k-1}$ time with $2k$ quantifiers (a $\Sigma_{2k}\mathsf{TIME}[kbs + t/b^{k-1}]$ machine). Next, we show how the construction can be modified to use only k quantifiers, *i.e.* we make a simulation in $\Sigma_k\mathsf{TIME}[kbs + t/b^{k-1}]$.

We describe a machine N using $2k$ quantifiers that simulates M below.

$N(x)$: Let C_0 and C_{t+1} be the unique initial and accept configurations of $M(x)$.
Return *Simulate*(C_0, C_{t+1}, k).

Simulate(C_i, C_j, j):
If $j = 0$ then *accept* iff C_i leads to C_j in at most t/b^{k-1} steps.

- *Existentially* guess machine configurations C_1^j, ..., C_b^j of $M(x)$.
  If $C_1^j \neq C_i$ then *reject*.
  If $C_b^j \neq C_j$ then *reject*.

- *Universally* choose $i_j \in \{1, \ldots, b\}$ and return *Simulate*$(C^j_{i_j}, C^j_{i_j+1}, j-1)$.

It is straightforward to verify that the procedure *Simulate* works analogously to the proof of $\mathsf{DTISP}[t,s] \subseteq \mathsf{ATIME}[s \cdot \log t]$, except that instead of guessing just the "midpoint" configuration C'', we are guessing b "midpoint" configurations of $M(x)$ before each recursive call. This increases the runtime, but lowers the number of required alternations.

Simulate with $j := k$ clearly has $2k$ quantifiers, guessing $O(b \cdot s)$ bits existentially and $O(\log t)$ bits universally between each recursive call, and running for $O(t/b^{k-1})$ deterministic time in the base case (on a RAM). Thus the procedure takes $O(k \cdot b \cdot s + t/b^{k-1})$ time overall.

How can we reduce the number of quantifiers from $2k$ to k? We exploit the fact that the computation is deterministic, and therefore closed under complement. Rewrite *Simulate* to be a "negation" of the above:

Simulate2(C_i, C_j, j):
If $j = 0$ then *accept* iff C_i leads to C_j in at most t/b^{k-1} steps.

- *Universally* choose configurations $C^j_1, \ldots, C^j_b$ of $M(x)$.
  If $C^j_1 \neq C_i$ then *accept*.
  If $C^j_b = C_j$ then *accept*.

- *Existentially* choose $i_j \in \{1, \ldots, b\}$.
  Return $\neg$*Simulate2*$(C^j_{i_j}, C^j_{i_j+1}, j-1)$.

Intuitively, *Simulate2* verifies that C_i leads to C_j by considering all sequences of configurations where the first configuration is C_i, but the last one is *not* C_j. *Simulate2* verifies that for any such sequence, there are two adjacent configurations that do *not* lead from one to the other. Since all sequences from C_i to some $C'_j \neq C_j$ fail to work, it must be that C_i leads to C_j. Clearly, *Simulate2* runs in the same time bound and number of alternations as *Simulate*, but the quantifiers start with a $\forall$ instead.

Our final algorithm makes the two procedures mutually recursive: rewrite *Simulate* so that it calls *Simulate2*, and vice-versa. Then, the number of quantifiers in $N(x)$ becomes exactly k, where the first $(k-1)$ quantifiers guess $O(b \cdot s)$ bits each, and the last quantifier runs in $O(t/b^{k-1})$ time. □

Remark 2.8. *Note that* $\mathsf{DTISP}[t,s] \subseteq \Pi_k\mathsf{TIME}[k \cdot b \cdot s + t/b^{k-1}]$ *follows immediately from the closure of* DTISP *under complementation.*

A significant instantiation of Lemma 2.7 is the following important corollary.

COROLLARY 2.9. *For all integers $k \geq 2$,* $\mathsf{DTISP}[t, s] \subseteq \Sigma_k\mathsf{TIME}[(ts^{k-1})^{1/k}]$.

PROOF. When $b = (t/s)^{1/k}$, the overall runtime of the Lemma 2.7 simulation is minimized, resulting in the corollary. □

For lower bounds involving co-nondeterministic machines, we use an analogous simulation, which was also observed by Fortnow and Van Melkebeek. Similar to with Lemma 2.7, one can show that for $k \geq 2$, $\mathsf{coNTISP}[n^c, n^{o(1)}] \subseteq \Pi_{2k-1}\mathsf{TIME}[n^{c/k}]$: $(k-1)$ pairs of two alternations (universally guessing $n^{c/k}$ configurations, then existentially picking one) are used to cut down the simulation time of a block of the computation by a kth root, and a final $\forall$ quantifier guesses the $n^{c/k}$ co-nondeterministic steps taken in a block.

LEMMA 2.10. (FORTNOW AND VAN MELKEBEEK (7)) *For time constructible $t(n) \geq n$ and $b(n)$ such that $1 \leq b(n) \leq t(n)$, and all $k \geq 1$,* $\mathsf{coNTISP}[t, t^{o(1)}] \subseteq \Pi_{2k-1}\mathsf{TIME}[t^{1/k+o(1)}]$.

3. New Time-Space Lower Bounds for SAT

We begin by showing a $n^{1.6616}$ time lower bound for deterministic RAMs using $n^{o(1)}$ space. The proof of this lower bound does not use any new tools, but rather relies solely on a new style of argument, introduced in Section 1.7 and outlined in the next section.

3.1. Intuition. As mentioned in Section 1.7, the idea behind our new lower bounds is to derive a sequence of "switching lemmas", where each new lemma invokes all of the previous lemmas in its derivation. Here we make that idea more precise, in the specific context of time-space lower bounds on nondeterminism.

If nondeterministic time n can be simulated in deterministic time n^c, then by Lemma 2.6 it follows that $\Sigma_k\mathsf{TIME}[n] \subseteq \Sigma_{k-1}\mathsf{TIME}[n^c]$. A fundamental observation behind our results is that, if we further assume nondeterministic time n is in deterministic time n^c and space $n^{o(1)}$ for $c < 2$, this not only implies that $\Sigma_k\mathsf{TIME}[n]$ can be efficiently simulated by a Σ_{k-1} machine, but also that the runtime for this simulation is *faster* than n^c. Moreover, as k increases, the implied simulation gets faster for appropriately small c. This simulation can be used to improve item (3) from the four-step scheme (the "alternation removal").

For an example of the idea, we give a simple proof that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ for $c < \sqrt{2}$. First, assume the contrary. Then by the ma-

chinery presented in the previous section,

$$\begin{aligned}\Sigma_2\mathsf{TIME}[n] \subseteq \mathsf{NTIME}[n^c] &\subseteq \mathsf{DTISP}[n^{c^2}, n^{o(1)}] \\ &\subseteq \Pi_2\mathsf{TIME}[n^{\frac{c^2}{2}+o(1)}] \subseteq \Pi_2\mathsf{TIME}[o(n)],\end{aligned}$$

a contradiction. (The first inclusion follows from the Alternations for Time Lemma, the second follows by assumption, and the third by Fortnow and Van Melkebeek's simulation.)

Consider if we let $c \geq \sqrt{2}$. Then the resulting derivation

$$\Sigma_2\mathsf{TIME}[n] \subseteq \Pi_2\mathsf{TIME}[n^{\frac{c^2}{2}+o(1)}]$$

is not quite a contradiction, but it is at the very least a *lemma* that, in conjunction with Lemma 2.6, implies $\Sigma_k\mathsf{TIME}[n] \subseteq \Sigma_{k\text{-}1}\mathsf{TIME}[n^{\frac{c^2}{2}+o(1)}]$, for all $k \geq 3$. Provided that $c < 2$, this is stronger than $\Sigma_k\mathsf{TIME}[n] \subseteq \Sigma_{k\text{-}1}\mathsf{TIME}[n^c]$. The lemma can then be used to get an even tighter inclusion for Σ_3 in Π_3, in particular

$$\Sigma_3\mathsf{TIME}[n] \subseteq \Sigma_2\mathsf{TIME}[n^{\frac{c^2}{2}+o(1)}] \subseteq \mathsf{DTISP}[n^{\frac{c^4}{2}+o(1)}, n^{o(1)}] \subseteq \Pi_3\mathsf{TIME}[n^{\frac{c^4}{6}+o(1)}].$$

If $c^4 < 6$, we have a contradiction. Otherwise, the above inclusion between Σ_3 and Π_3 can be used to prove a relation between Σ_4 and Π_4. An inductive strategy for improving lower bounds naturally arises: derive increasingly better Π_k simulations of Σ_k using the previous simulations obtained, and take c to be the largest constant that implies $\Pi_k\mathsf{TIME}[n] \subseteq \Sigma_k\mathsf{TIME}[o(n)]$ for some k. In many cases, this particular attack yields better lower bounds than previous approaches, as we shall see throughout the paper.

3.2. First SAT Lower Bound. We now present a more formal exposition of the above ideas, proving a $n^{1.6616}$ time lower bound for SAT on $n^{o(1)}$ space machines. The main theorem of this section is the following.

THEOREM 3.1. *For every integer $k \geq 2$ and $c \geq 1$ such that $c < f(k)$, $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, where $f(k) := \prod_{j=1}^{k-1}(1+1/j)^{1/2^j}$.*

Let us first observe some properties of the f function.

LEMMA 3.2. *$f(k)$ is monotone increasing and converges to a value greater than* 1.6616.

PROOF OF LEMMA 3.2. As $(1+1/j)^{1/2^j} > 1$ for all j, it is evident that $f(k)$ is monotone increasing. Observe that $(1+1/j)^{1/2^j} \leq \exp(\frac{1}{j \cdot 2^j})$, so $f(k) \leq \exp(\sum_{j=1}^{k-1} \frac{1}{j \cdot 2^j})$. As this sum converges, $f(k)$ converges also. Computation of $f(12)$ suffices to show $f(k) > 1.6616$. □

Corollary 2.2, Theorem 3.1, and Lemma 3.2 immediately imply Theorem 1.1 from the Introduction, *i.e.* the $n^{1.6166}$ time-space lower bound for SAT.

We use the inductive argument described in previous sections to prove a relation between Σ_k and Π_k for all $k \geq 2$, from which the theorem follows. Define an expression e by the inductive definition

$$e(2) := \frac{c^2}{2}, e(k) := \frac{c^2}{k}\left(\prod_{i=1}^{k-1} e(i)\right). \tag{3.3}$$

LEMMA 3.4. *Assume*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}] \qquad (*)$$

holds for some $c \geq 1$, and let $k \geq 2$ be an integer. If $e(i) \geq 1$ for all $i \in \{2, \ldots, k-1\}$, then

$$\Sigma_k\mathsf{TIME}[n] \subseteq \Pi_k\mathsf{TIME}[n^{e(k)+o(1)}].$$

PROOF. By induction. The case $k = 2$ is exactly the $n^{\sqrt{2}}$ lower bound of Section 3.1. We revisit it for completeness. Assume $(*)$ holds for $c > 1$; then

$$\Sigma_2\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^{c^2}, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{c^2/2+o(1)}] = \Pi_2\mathsf{TIME}[n^{e(2)+o(1)}],$$

where the last inclusion follows from Corollary 2.9. Observe this is precisely what we derived earlier in Section 3.1.

Induction Hypothesis: Assume for all $i \in \{2, \ldots, k-1\}$ that $e(i) \geq 1$ and $\Sigma_i\mathsf{TIME}[n] \subseteq \Pi_i\mathsf{TIME}[n^{e(i)+o(1)}]$.

We now prove the theorem for general k. The "Alternations For Time" Lemma (Lemma 2.6) and induction hypothesis imply that

$$\Sigma_\ell\mathsf{TIME}[n] \subseteq \Sigma_{\ell-1}\mathsf{TIME}[n^{e(\ell-1)+o(1)}], \text{ for } \ell \in \{2, \ldots, k-1\}.$$

We therefore have by padding (which is possible since each $e(i) \geq 1$)

$$\begin{aligned}\Sigma_k\mathsf{TIME}[n] &\subseteq \Sigma_{k-1}\mathsf{TIME}[n^{e(k-1)+o(1)}] \\ &\subseteq \Sigma_{k-2}\mathsf{TIME}[n^{(e(k-1)+o(1))(e(k-2)+o(1))}] \subseteq \cdots \subseteq \Sigma_2\mathsf{TIME}[n^{\prod_{i=2}^{k-1}(e(i)+o(1))}].\end{aligned}$$

But we also have

$$\begin{aligned}\Sigma_2\mathsf{TIME}[n^{\prod_{i=2}^{k-1}(e(i)+o(1))}] &\subseteq \mathsf{NTIME}[n^{c\prod_{i=2}^{k-1}(e(i)+o(1))}]\\ &\subseteq \mathsf{DTISP}[n^{c^2\prod_{i=2}^{k-1}(e(i)+o(1))}, n^{o(1)}] \subseteq \Pi_k\mathsf{TIME}[n^{\frac{c^2}{k}\prod_{i=2}^{k-1}(e(i)+o(1))}]\\ &\subseteq \Pi_k\mathsf{TIME}[n^{e(k)+o(1)}],\end{aligned}$$

where the penultimate inclusion follows from Corollary 2.9 (Fortnow and Van Melkebeek's simulation), and the last inclusion follows by definition of $e(k)$. □

Proof of Theorem 3.1. Let k' be the smallest integer such that $c < f(k')$; such a k' exists since f is monotonically increasing (Lemma 3.2). Consider when $k' = 2$. If we assume $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, then Lemma 3.4 implies

$$\Sigma_2\mathsf{TIME}[n] \subseteq \Pi_2\mathsf{TIME}[n^{e(2)+o(1)}] = \Pi_2\mathsf{TIME}[n^{c^2/2+o(1)}]. \quad (*)$$

Now if $c < 2^{1/2} = (1+1/1)^{1/2} = f(2)$, then $e(2) < 1$. Therefore $(*)$ contradicts the "No Complementary Speedup" Theorem (Theorem 2.4), and this concludes the base case. Otherwise, observe that $c \geq 2^{1/2}$ implies $e(2) \geq 1$. In fact, we have the following arithmetic relationship between the expressions e and f.

Claim 3.5. *For all i, $e(i) \geq 1 \iff c \geq f(i)$.*

Proof of Claim 3.5. Recall that $f(k) := \prod_{j=1}^{k-1}(1+1/j)^{1/2^j}$, and $e(1) := 1$, $e(k) := \frac{c^2}{k}\left(\prod_{i=1}^{k-1} e(i)\right)$.

First, we claim that $e(i) = \frac{c^{2^{i-1}}}{i!\prod_{j=2}^{i-2}(j!)^{2^{i-j-2}}}$ follows from a proof by induction. The denominator can be simplified further to get $e(i) = \frac{c^{2^{i-1}}}{i\cdot\prod_{j=2}^{i-1} j^{2^{i-j-1}}}$.

For all i, let c_i be the unique number in $(1,2)$ such that $e(i) = 1$ when $c = c_i$, *i.e.* $(c_i)^{2^{i-1}} = i\cdot\prod_{j=2}^{i-1} j^{2^{i-j-1}}$. It suffices for us to show that $c_i = f(i)$. Observe

$$(c_{i-1})^{2^{i-1}} = ((c_{i-1})^{2^{i-2}})^2 = (i-1)^2\cdot\prod_{j=2}^{i-2} j^{2^{i-j-1}}$$

by definition of c_{i-1}. Hence

$$\left(\frac{c_i}{c_{i-1}}\right)^{2^{i-1}} = \frac{i\cdot\prod_{j=2}^{i-1} j^{2^{i-j-1}}}{(i-1)^2\prod_{j=2}^{i-2} j^{2^{i-j-1}}} = i/(i-1),$$

so $\frac{c_i}{c_{i-1}} = \left(\frac{i}{i-1}\right)^{1/2^{i-1}}$. Therefore,

$$c_i = (c_i/c_{i-1})(c_{i-1}/c_{i-2}) \cdots (c_3/c_2)c_2$$
$$= \prod_{j=2}^{i} \left(1 + \frac{1}{j-1}\right)^{\frac{1}{2^{j-1}}} = \prod_{j=1}^{i-1} \left(1 + \frac{1}{j}\right)^{\frac{1}{2^j}} = f(i).$$

□

Claim 3.5 and our choice of k' implies that k' is the smallest integer such that $e(k') < 1$. Therefore for all $i \leq k' - 1$ we have that $e(i) \geq 1$, so Lemma 3.4 applies. Namely,

$$\Sigma_{k'}\mathsf{TIME}[n] \subseteq \Pi_{k'}\mathsf{TIME}[n^{e(k')+o(1)}]. \quad (**)$$

However, $(**)$ contradicts the "No Complementary Speedup" Theorem (Theorem 2.4), since $e(k') < 1$. This completes the proof of Theorem 3.1. □

The mechanics of the above proof also demonstrate a new time-space tradeoff for SAT.

Corollary 3.6. *For all $c < 1.66$ there is a $d \in (0,1)$ such that SAT is not in $\mathsf{DTISP}[n^c, n^d]$.*

Proof. (Sketch) In the above proof, one can replace the $n^{o(1)}$ space bound by n^d for a sufficiently small $d > 0$. The time bounds of the alternating simulations increase only by an additive factor of $q_k d$ in the exponents, where q_k is a constant that depends on the number of alternations. □

3.3. From $n^{1.661}$ to $n^{1.732}$: Conditional Speedups of DTISP. Fortnow and Van Melkebeek's result (Corollary 2.9, of Lemma 2.7) that $\mathsf{DTISP}[t, t^{o(1)}] \subseteq \Pi_k\mathsf{TIME}[t^{1/k+o(1)}]$ is an *unconditional* one, whereas all other inclusions we derived in the above actually depended on the assumption that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$. Here we show how to carefully exploit this assumption to get

$$\mathsf{DTISP}[t, t^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[t^{1/(2+\delta)+o(1)}]$$

for some $\delta > 0$ that depends on the constant $c < 2$. This new containment allows us to push the SAT lower bound above $n^{\sqrt{3}}$.

LEMMA 3.7. *Let $c \in (1,2)$. Define $d(1) := 2$, $d(k) := 1 + \frac{d(k-1)}{c}$. If $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ then for all $k \in \mathbb{N}$, $\mathsf{DTISP}[n^{d(k)}, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$.*

Let us briefly outline how the proof of the lemma goes. As the statement of the lemma suggests, it is an inductive argument, but of a different kind than before. First we use the containment of nondeterministic linear time in $\mathsf{DTISP}[n^c, n^{o(1)}]$ to obtain

$$\mathsf{NTIME}[n^\ell] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$$

for some $\ell > 1$. We show that this containment can be used to get a *better* simulation of DTISP in $\Pi_2\mathsf{TIME}$ than the known one. In particular,

$$\mathsf{DTISP}[n^d, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}],$$

for some $d > 2$. This new simulation can in turn be used to obtain

$$\mathsf{NTIME}[n^{\ell'}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$$

for some $\ell' > \ell$. That is, the two containments of DTISP in $\Pi_2\mathsf{TIME}$ and NTIME in $\Pi_2\mathsf{TIME}$ can mutually improve upon each other, and the amount of improvement that can be achieved depends on the constant c.

PROOF OF LEMMA 3.7. By induction on k. The $k = 1$ case is trivial since $\mathsf{DTISP}[n^2, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$ follows unconditionally.

Suppose that both $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ and $\mathsf{DTISP}[n^{d(k)}, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$. By padding and the inductive hypothesis (note $d(k) \geq 2$ for all k) we have

$$\mathsf{NTIME}[n^{d(k)/c}] \subseteq \mathsf{DTISP}[n^{d(k)}, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]. \quad (*)$$

We use this inclusion to get a better speedup of DTISP in $\Pi_2\mathsf{TIME}[n^{1+o(1)}]$. Consider a Π_2 simulation of $\mathsf{DTISP}[n^{1+d(k)/c}, n^{o(1)}]$, where we only guess $O(n)$ bits (*i.e.* $n^{1-o(1)}$ configurations) in the universal quantifier. (Formally, we are invoking Lemma 2.7, with $b = n^{1-o(1)}$.) Written as a first-order logic sentence, the Π_2 simulation looks like:

$$\left(\begin{array}{c} \forall \text{ configurations } C_1, \ldots, C_{n^{1-o(1)}} \text{ of } M \text{ on } x \\ \text{s.t. } C_{n^{1-o(1)}} \text{ is rejecting} \end{array} \right) (\exists i \in \{1, \ldots, n^{1-o(1)} - 1\})$$

$$[C_i \text{ does not lead to } C_{i+1} \text{ in } n^{d(k)/c+o(1)} \text{ time}].$$

The $(\exists i \cdots)[\cdots]$ part in the above sentence corresponds to an NTIME computation that takes an input of $O(n)$ bits (the input x, plus the list of configurations) and runs in $n^{d(k)/c+o(1)}$ time. Hence, by inclusion $(*)$, this nondeterministic computation can be replaced with a $\Pi_2\mathsf{TIME}[n^{1+o(1)}]$ computation. Therefore $\mathsf{DTISP}[n^{1+d(k)/c}, n^{o(1)}]$ can be simulated by a computation of the form:

$$\left(\begin{array}{c} \forall \text{ configurations } C_1, \ldots, C_{n^{1-o(1)}} \text{ of } M \text{ on } x \\ \text{s.t. } C_{n^{1-o(1)}} \text{ is rejecting} \end{array} \right) (\forall y, |y| = a|x|^{1+o(1)})$$

$$(\exists z, |z| = a|z|^{1+o(1)})[R(C_1, \ldots, C_{n^{1-o(1)}}, x, y, z)],$$

for some deterministic linear time relation R and constant $a > 0$. That is, $\mathsf{DTISP}[n^{d(k+1)}, n^{o(1)}]$ is in $\Pi_2\mathsf{TIME}[n^{1+o(1)}]$. □

Corollary 3.8. *Let $c \in (1,2)$. If $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ then for all $\varepsilon \in (0, \frac{1}{c-1})$, $\mathsf{DTISP}[n^{\frac{c}{c-1}-\varepsilon}, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$.*

(Note we need $\varepsilon \leq \frac{1}{c-1}$, since ε must satisfy $\frac{c}{c-1} - \varepsilon \geq 1$.)

Proof. For any $c < 2$, the sequence $\{d(k)\}_{k\in\mathbb{N}}$ is monotone non-decreasing, and converges to a constant given by $d_\infty = 1 + \frac{d_\infty}{c}$, which is $d_\infty = c/(c-1)$. Hence for all $\varepsilon > 0$, there is a finite K such that $d(K) \geq \frac{c}{c-1} - \varepsilon$. □

Notice that Corollary 3.8 can be padded in a standard way.

Corollary 3.9. *Let $c \in (1,2)$. If $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, then for all time constructible $t(n) \geq n$ and all $\varepsilon > 0$, $\mathsf{DTISP}[t(n)^{\frac{c}{c-1}-\varepsilon}, t(n)^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[t(n)^{1+o(1)}]$.*

We are now armed with an additional lower bound tool. If Corollary 3.9 is combined with the inductive argument from Section 3.2, we see an interesting result: instead of having Lipton-Viglas' $n^{\sqrt{2}}$ lower bound as a base case, we now have something resembling Fortnow-Van Melkebeek's n^ϕ lower bound as a base case, with ϕ being the golden ratio.

More precisely, we know that if $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ then $c \geq \phi$, hence $c^2 \geq c/(c-1) - \varepsilon$ for all $\varepsilon > 0$. Then, for all $\varepsilon > 0$ and sufficiently small $\varepsilon_2 > 0$, there is an $\varepsilon_1 > 0$ such that

$$\begin{aligned} \Sigma_2\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^{c^2}, n^{o(1)}] &\subseteq \mathsf{DTISP}[\left(n^{c^2 \cdot \left(\frac{c-1}{c}+\varepsilon_1\right)}\right)^{c/(c-1)-\varepsilon}, n^{o(1)}] \\ &\subseteq \mathsf{DTISP}[\left(n^{c(c-1)+\varepsilon_2}\right)^{c/(c-1)-\varepsilon}, n^{o(1)}] \\ &\subseteq \Pi_2\mathsf{TIME}[n^{c\cdot(c-1)+\varepsilon_2+o(1)}], \end{aligned}$$

where the penultimate inclusion follows by taking $\varepsilon_1 = \varepsilon_2/c^2$, and the last inclusion follows from Corollary 3.9. Observe that this new inclusion of Σ_2 linear time in Π_2 time is *superior* to the previously derived $\Sigma_2\mathsf{TIME}[n] \subseteq \Pi_2\mathsf{TIME}[n^{c^2/2+o(1)}]$ inclusion, for all $c < 2$. More precisely, $c(c-1) < c^2/2$ for all $c \in (1,2)$, and this is the source of our lower bound improvement.

Proceeding inductively as before, we can derive for all sufficiently small $\varepsilon_3 > 0$ that

$$\begin{aligned}\Sigma_3\mathsf{TIME}[n] \subseteq \Sigma_2\mathsf{TIME}[n^{c\cdot(c-1)+\varepsilon_2+o(1)}] &\subseteq \mathsf{DTISP}[n^{c^3\cdot(c-1)+\varepsilon_2 c^2+o(1)}, n^{o(1)}]\\ &\subseteq \Pi_3\mathsf{TIME}[n^{\frac{c^3\cdot(c-1)}{3}+\varepsilon_3+o(1)}],\end{aligned}$$

by setting $\varepsilon_2 = 3\varepsilon_3/c^2$.

Similarly, for all sufficiently small $\varepsilon_4 > 0$ we can derive

$$\begin{aligned}\Sigma_4\mathsf{TIME}[n] \subseteq \Sigma_3\mathsf{TIME}[n^{\frac{c^3\cdot(c-1)}{3}+\varepsilon_3+o(1)}] &\subseteq \Sigma_2\mathsf{TIME}[n^{\frac{c^4\cdot(c-1)^2}{3}+c(c-1)\varepsilon_3+o(1)}]\\ &\subseteq \mathsf{DTISP}[n^{\frac{c^6\cdot(c-1)^2}{3}+c^3(c-1)\varepsilon_3+o(1)}, n^{o(1)}]\\ &\subseteq \Pi_4\mathsf{TIME}[n^{\frac{c^6\cdot(c-1)^2}{12}+\varepsilon_4+o(1)}],\end{aligned}$$

by setting $\varepsilon_4 = 4\varepsilon_3/(c^3(c-1))$.

We can formally state the relation between Σ_k and Π_k for general k as follows. Define $g(2) := c(c-1)$, and for $k \geq 3$,

$$g(k) := \frac{c^{3\cdot 2^{k-3}}(c-1)^{2^{k-3}}}{k\cdot(\prod_{i=3}^{k-1} i^{2^{(k-1)-i}})}.$$

Observe that $g(3) = c^3(c-1)/3$ and $g(4) = c^{3\cdot 2}(c-1)^2/(4\cdot 3) = c^6(c-1)^2/12$.

Lemma 3.10. *Assume* $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ *holds for some* $c \geq 1$*, and let* $k \geq 2$ *be an integer. If* $g(i) \geq 1$ *for all* $i \in \{2,\ldots,k-1\}$*, then*

$$\Sigma_k\mathsf{TIME}[n] \subseteq \Pi_k\mathsf{TIME}[n^{g(k)+o(1)}].$$

Proof. By induction on k. □

Finally, we are in position to improve our previous lower bound for SAT (Theorem 1.2) to $\Omega(n^{1.7327})$ time on subpolynomial space machines.

Proof of Theorem 1.2. By Corollary 2.2, it suffices to show $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^{1.7327}, n^{o(1)}]$. Assuming $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, we wish to find the largest c possible such that $g(k) < 1$ for some k, while $g(k') \geq 1$ for all

$k' < k$. (Note that, as with the function f, the function g is also monotone decreasing.) By Lemma 3.10, such a c implies that $\Sigma_k\mathsf{TIME}[n] \subseteq \Pi_k\mathsf{TIME}[o(n)]$, and therefore is a contradiction. Observe that the function $g(k)$ can be simplified to

$$\begin{aligned} g(k) &= \frac{c^{3\cdot 2^{k-3}}(c-1)^{2^{k-3}}}{k\cdot(3^{2^{k-4}}\cdot 4^{2^{k-5}}\cdot 5^{2^{k-6}}\cdots(k-1))} \\ &= \left(\frac{c^3(c-1)}{k^{2^{-k+3}}\cdot(3^{2^{-1}}\cdot 4^{2^{-2}}\cdot 5^{2^{-3}}\cdots(k-1)^{2^{-k+3}})}\right)^{2^{k-3}}. \end{aligned}$$

Now, $g(k) < 1$ if and only if

$$g'(k) := \frac{c^3(c-1)}{k^{2^{-k+3}}\cdot(3^{2^{-1}}\cdot 4^{2^{-2}}\cdot 5^{2^{-3}}\cdots(k-1)^{2^{-k+3}})} < 1,$$

so it suffices to analyze the latter expression.

The denominator of $g'(k)$ numerically converges to $3.81213\cdots$ as $k \to \infty$. Therefore the calculation of c reduces to finding the positive root of $c^3\cdot(c-1) = 3.81213$, or $c \approx 1.7327$. □

A straightforward application of the methods of Tourlakis (25) further yields a lower bound on non-uniform machines for SAT (we use the standard notation of $\mathcal{C}/f(n)$ to denote class $\mathcal{C}$ augmented with advice strings of length $f(n)$ on inputs of length n). We omit the details.

Corollary 3.11. $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^{1.7327}, n^{o(1)}]/n^{o(1)}$.

4. Better Lower Bounds for Alternating Time

The method can also be used to improve known lower bounds for alternating linear time. Here, we just show how the inductive argument extends to goes for Σ_2 and Σ_{100}— the other cases are extremely similar.

4.1. Deterministic RAMs. We begin by showing the Σ_k time lower bounds for small space deterministic RAMs. An argument similar to the one of Section 4.2 can be applied by proving a simple generalization of Lemma 3.7.

Lemma 4.1. *Let $\ell > 0$ be an integer and let $c < \ell + 1$. Define $d'(1) := \ell + 1$, $d'(k) := \ell + \frac{d'(k-1)}{c}$. If $\Sigma_\ell\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, then for all $k \in \mathbb{N}$, $\mathsf{DTISP}[n^{d'(k)}, n^{o(1)}] \subseteq \Pi_{\ell+1}\mathsf{TIME}[n^{1+o(1)}]$.*

Observe when $\ell = 1$, the lemma is equivalent to Lemma 3.7. The proof for arbitrary ℓ is analogous. We sketch it for completeness.

Proof of Lemma 4.1. (Sketch) The case $k = 1$ follows from Lemma 2.7. For $k > 1$, the induction hypothesis implies

$$\Sigma_\ell \mathsf{TIME}[n^{d'(k-1)/c}] \subseteq \mathsf{DTISP}[n^{d'(k)}, n^{o(1)}] \subseteq \Pi_{\ell+1}\mathsf{TIME}[n^{1+o(1)}]. \tag{4.2}$$

Essentially, we rely on the flexibility of the alternating simulation of DTISP from Lemma 2.7. Consider a $\mathsf{DTISP}[n^{\ell+\frac{d'(k-1)}{c}}, n^{o(1)}]$ computation simulated in $\Sigma_{\ell+1}$ along the lines of Lemma 2.7, where the first ℓ quantifiers each guess n configurations (*i.e.* $b = n$), the $(\ell+1)^{\text{th}}$ quantifier picks a configuration from the ℓ^{th} quantifier, and the remaining computation is a $\mathsf{DTISP}[n^{d'(k-1)/c}, n^{o(1)}]$ computation:

$$\begin{aligned}&(\exists\ n \text{ configurations})(\forall i_1 \in [n])\\ &(\forall\ n \text{ configurations})(\exists i_2 \in [n]) \quad \cdots \quad (\mathsf{Q} i_\ell \in [n]))D(x, \ldots),\end{aligned}$$

where D runs in $n^{d'(k-1)/c}$ time and $n^{o(1)}$ space, and $\mathsf{Q} = \exists$ if ℓ is even and $\mathsf{Q} = \forall$ if ℓ is odd.

But (4.2) implies that we can replace the $(\forall i_1 \cdots) \cdots D(x, \cdots)$ part of the above with a $\Pi_{\ell+1}\mathsf{TIME}[n^{1+o(1)}]$ computation. Doing so, we find that $\mathsf{DTISP}[n^{\ell+\frac{d'(k-1)}{c}}, n^{o(1)}] \subseteq \Pi_{\ell+1}\mathsf{TIME}[n^{1+o(1)}]$. □

Taking the limit of the sequence $\{d'(k)\}_{k\in\mathbb{N}}$ yields the following corollary.

Corollary 4.3. *For ℓ and c as the above, If $\Sigma_\ell\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, then for all sufficiently small $\varepsilon > 0$, $\mathsf{DTISP}[n^{\ell\cdot\frac{c-\varepsilon}{c-1}}, n^{o(1)}] \subseteq \Pi_{\ell+1}\mathsf{TIME}[n^{1+o(1)}]$.*

Theorem 4.4. *For all $k \geq 2$, $\Sigma_2\mathsf{TIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$, for $c < 2.788$.*

We briefly sketch the argument. First observe that if we assume the contrary, then $\Sigma_2\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}] \subseteq \Pi_2\mathsf{TIME}[n^{c/2+o(1)}]$. Thus $c \geq 2$, and by Corollary 4.3, $\mathsf{DTISP}[n^{2\cdot\frac{c-\varepsilon}{c-1}}] \subseteq \Pi_3\mathsf{TIME}[n^{1+o(1)}]$ for small $\varepsilon > 0$.

Hence

$$\begin{aligned}\Sigma_3\mathsf{TIME}[n] \subseteq \Sigma_2\mathsf{TIME}[n^{c/2+o(1)}] &\subseteq \mathsf{DTIME}[n^{c^2/2+o(1)}]\\ &\subseteq \mathsf{DTIME}[\left(n^{\frac{c(c-1)}{4}+\varepsilon'}\right)^{2\frac{c}{c-1}-\varepsilon}]\\ &\subseteq \Pi_3\mathsf{TIME}[n^{\frac{c(c-1)}{4}+\varepsilon'}]\end{aligned}$$

for arbitrarily small $\varepsilon' > 0$ and appropriate $\varepsilon > 0$. Therefore, to avoid a contradiction, it must be that $c(c-1) \geq 4$, or $c \geq 2.56$. Continuing with Σ_4, Σ_5, *etc.*, one can numerically derive that $c > 2.788$ is needed to avoid a contradiction.

The columns of Table 1 for deterministic RAMs can be completed in similar fashion. A general lower bound result can be stated as follows.

THEOREM 4.5. *Let $\ell \geq 2$ be an integer. Let $c > 1$ be such that for some integer $k \geq 2$,*

$$c^{2^k} < (\ell + k) \prod_{i=0}^{k-1} (\ell + i)^{2^{k-i-1}}.$$

Then $\Sigma_\ell \mathsf{TIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$.

PROOF. (Sketch) The induction begins with:

$$\Sigma_\ell \mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}] \subseteq \Pi_\ell \mathsf{TIME}[n^{c/\ell + o(1)}]$$

and

$$\begin{aligned} \Sigma_{\ell+1} \mathsf{TIME}[n] \subseteq \Sigma_\ell \mathsf{TIME}[n^{\frac{c}{\ell}+o(1)}] &\subseteq \mathsf{DTISP}[n^{\frac{c^2}{\ell}+o(1)}, n^{o(1)}] \\ &\subseteq \Pi_{\ell+1} \mathsf{TIME}[n^{\frac{c^2}{\ell \cdot (\ell+1)}+o(1)}]. \end{aligned}$$

In general, one can prove that for $k \geq \ell$,

$$\Sigma_{k+\ell} \mathsf{TIME}[n] \subseteq \Pi_{k+\ell} \mathsf{TIME}[n^{e_\ell(k)}]$$

where $e_\ell(0) = c/\ell, e_\ell(k) = (e_\ell(k-1))^2 \cdot (k-1)/k$. An explicit expression for e_ℓ is given by

$$e_\ell(k) = \frac{c^{2^k}}{(\ell + k) \cdot \prod_{i=0}^{k-1} (\ell + i)^{2^{k-1-i}}}.$$

□

For example, the $n^{100.99}$ time lower bound for Σ_{100} can be derived using Theorem 4.5. When $k > 12$ and $c < 100.99$, the expression $\frac{c^{2^k}}{(100+k) \cdot \prod_{i=0}^{k-1} (100+i)^{2^{k-1-i}}}$ is less than 1.

A simple expression for a lower bound on $\Sigma_\ell \mathsf{TIME}[n]$ is given by the following corollary, which is already an improvement over Fortnow and Van Melkebeek's $n^{\ell-\varepsilon}$ lower bound for $\Sigma_\ell \mathsf{TIME}[n]$. In the following, we prove Corollary 1.3 from Section 1.4, which claims that $\Sigma_\ell \mathsf{TIME}[n] \not\subseteq \mathsf{DTISP}[n^{\ell(1+1/\ell)^{1/2}}, n^{o(1)}]$.

PROOF OF COROLLARY 1.3. First, we claim that

$$\ell^{2^k}\left(1+\frac{1}{\ell}\right)^{2^{k-1}} < (\ell+k)\cdot\prod_{i=0}^{k-1}(\ell+i)^{2^{k-1-i}}. \tag{4.6}$$

Observe that the corollary follows from (4.6), due to Theorem 4.5 and the fact that

$$\begin{aligned} c = \ell\cdot\left(1+\frac{1}{\ell}\right)^{1/2} \implies c^{2^k} &= \ell^{2^k}\left(1+\frac{1}{\ell}\right)^{2^{k-1}} \\ &< (\ell+k)\cdot\prod_{i=0}^{k-1}(\ell+i)^{2^{k-1-i}}. \end{aligned}$$

We now prove (4.6). For $k \geq 3$, the following sequence of inequalities holds:

$$\begin{aligned} (\ell+k)\cdot\prod_{i=0}^{k-1}(\ell+i)^{2^{k-1-i}} &= \ell(1+k/\ell)\cdot\prod_{i=0}^{k-1}\ell^{2^{k-1-i}}(1+i/\ell)^{2^{k-1-i}} \\ &= \ell^{1+\sum_{i=0}^{k-1}2^{k-1-i}}(1+k/\ell)\cdot\prod_{i=0}^{k-1}(1+i/\ell)^{2^{k-1-i}} \\ &> \ell^{1+2^{k-1}(1+1/2+1/4+\cdots+1/2^{k-1})}(1+1/\ell)^{1+2^{k-2}+\cdots+2+1} \\ &= \ell^{1+2^{k-1}(2-1/2^{k-1})}(1+1/\ell)^{2^{k-1}} = \ell^{2^k}(1+1/\ell)^{2^{k-1}}. \end{aligned}$$

□

While the above bound is in a simple form, we note that it is still somewhat weak for large values of ℓ. In particular, we know (empirically, from analysis of $\ell = 100$) that the lower bound of Theorem 4.5 approaches $\Omega(n^{\ell+1})$ for $\Sigma_\ell\mathsf{TIME}[n]$ when ℓ is large, while Corollary 1.3 only implies an $\Omega(n^{\ell+1/2-\varepsilon})$ lower bound for $\Sigma_\ell\mathsf{TIME}[n]$ when ℓ is sufficiently large.

4.2. Co-Nondeterministic RAMs. Our method can also be used to derive lower bounds for for $\Sigma_\ell\mathsf{TIME}[n]$ on co-nondeterministic machines using $n^{o(1)}$ space. We only sketch how these arguments go, since they are similar to the above. Our purpose here is not to prove the best possible bound, but to demonstrate the lines of reasoning. We shall invoke Lemma 2.10, Fortnow and Van Melkebeek's alternating simulation of $\mathsf{coNTISP}$.

First we look at the case when $\ell = 2$. Assume for contradiction that $\Sigma_2\mathsf{TIME}[n] \subseteq \mathsf{coNTISP}[n^c, n^{o(1)}]$. This implies that, for $k \geq 1$, two alternations can be removed from a Σ_{k+2} computation with cost c in the exponent

(the proof of this is similar to Lemma 2.6). We derive

$$\Sigma_4\mathsf{TIME}[n] \subseteq \Sigma_2\mathsf{TIME}[n^c] \subseteq \mathsf{coNTISP}[n^{c^2}, n^{o(1)}] \subseteq \Pi_3\mathsf{TIME}[n^{c^2/2+o(1)}].$$

Now assuming $c \geq \sqrt{2}$, this implies $\Sigma_5\mathsf{TIME}[n] \subseteq \Sigma_3\mathsf{TIME}[n^{c^2/2+o(1)}]$. In particular, the $\Pi_4\mathsf{TIME}[n]$ part of a $\Sigma_5\mathsf{TIME}[n]$ computation can be replaced with a $\Sigma_3\mathsf{TIME}[n^{c/2+o(1)}]$ computation, leaving a $\Sigma_3\mathsf{TIME}[n^{c/2+o(1)}]$ computation. Thus

$$\begin{aligned}\Sigma_6\mathsf{TIME}[n] \subseteq \Sigma_4\mathsf{TIME}[n^{\frac{c^2}{2}+o(1)}] &\subseteq \Sigma_2\mathsf{TIME}[n^{\frac{c^3}{2}+o(1)}] \\ &\subseteq \mathsf{coNTISP}[n^{\frac{c^4}{2}+o(1)}, n^{o(1)}] \\ &\subseteq \Pi_5\mathsf{TIME}[n^{\frac{c^4}{6}+o(1)}].\end{aligned}$$

Hence when $c^4 \geq 6$, $\Sigma_7\mathsf{TIME}[n] \subseteq \Sigma_5\mathsf{TIME}[n^{c^4/6+o(1)}]$ by similar argument as above, and

$$\begin{aligned}\Sigma_8\mathsf{TIME}[n] \subseteq \Sigma_6\mathsf{TIME}[n^{\frac{c^4}{6}+o(1)}] &\subseteq \Sigma_4\mathsf{TIME}[n^{\frac{c^2}{2}\cdot\frac{c^4}{6}+o(1)}] \\ &\subseteq \mathsf{coNTISP}[n^{\frac{c^8}{12}+o(1)}, n^{o(1)}] \\ &\subseteq \Pi_7\mathsf{TIME}[n^{\frac{c^8}{48}+o(1)}],\end{aligned}$$

which implies $\Sigma_9\mathsf{TIME}[n] \subseteq \Sigma_7\mathsf{TIME}[n^{c^8/48+o(1)}]$. The acute reader will notice that the sequence $c^2/2$, $c^4/6$, $c^8/48$ corresponds precisely to the expressions $e(2)$, $e(3)$, and $e(4)$ where e is defined by equation (3.3) in the $n^{1.6616}$ SAT lower bound of Section 3.2. Indeed, it can be proved inductively that

$$\Sigma_{2k+1}\mathsf{TIME}[n] \subseteq \Sigma_{2k-1}\mathsf{TIME}[n^{e(k)+o(1)}].$$

Therefore if $c < 1.6616$, we obtain a contradiction with a time hierarchy theorem.

Let us now look at a more general case: lower bounds on $\Sigma_\ell\mathsf{TIME}[n]$ when ℓ is odd. If $\ell = 2k-1$ and $\Sigma_\ell\mathsf{TIME}[n] \subseteq \mathsf{coNTISP}[n^c, n^{o(1)}]$, then $\ell+1$ alternations from an alternating time t computation can be removed at cost t^c, and

$$\Sigma_\ell\mathsf{TIME}[n] \subseteq \mathsf{coNTISP}[n^c, n^{o(1)}] \subseteq \Pi_\ell\mathsf{TIME}[n^{c/k+o(1)}].$$

Thus for $\ell + 2 = 2(k+1) - 1$,

$$\begin{aligned}\Sigma_{\ell+1}\mathsf{TIME}[n] \subseteq \Sigma_\ell\mathsf{TIME}[n^{(c/k)+o(1)}] &\subseteq \mathsf{coNTISP}[n^{c(c/k)+o(1)}, n^{o(1)}] \\ &\subseteq \Pi_{\ell+2}\mathsf{TIME}\left[n^{\frac{c^2}{k(k+1)}+o(1)}\right],\end{aligned}$$

and similarly

$$\begin{aligned}
\Sigma_{\ell+4}\mathsf{TIME}[n] &\subseteq \Sigma_{\ell+2}\mathsf{TIME}\left[n^{\left(\frac{c^3}{k^2(k+1)}\right)^2+o(1)}\right] \\
&\subseteq \mathsf{coNTISP}\left[n^{c(c/k)^2\left(\frac{c^3}{k^2(k+1)}\right)^2+o(1)}, n^{o(1)}\right] \\
&\subseteq \Pi_{\ell+4}\mathsf{TIME}\left[n^{\frac{c^9}{k^6(k+1)^2(k+2)}+o(1)}\right].
\end{aligned}$$

In general, the expression for the exponent derived in the jth inclusion satisfies the recurrence $r(1) = c/k$, $r(j) = \frac{c}{k+j-1} \cdot \prod_{i=1}^{j-1} r(i)^2$. We can simplify r to

$$\begin{aligned}
r(j) &= r(j-1)^2 \cdot \frac{c}{k+j-1} \cdot \prod_{i=1}^{j-2} r(i)^2 \\
&= \left(\frac{c^2}{(k+j-2)^2} \cdot \prod_{i=1}^{j-2} r(i)^4\right) \cdot \frac{c}{k+j-1} \cdot \prod_{i=1}^{j-2} r(i)^2 \\
&= \frac{c^3}{(k+j-2)^2(k+j-1)} \cdot \prod_{i=1}^{j-2} r(i)^6 = r(j-1)^3 \cdot \frac{(k+j-2)}{(k+j-1)}.
\end{aligned}$$

For $\ell = 3$ ($k = 1$) and $\ell = 99$ ($k = 50$), the recurrence r implies a lower bound of $n^{2.390}$ and $n^{50.49}$, respectively.

In principle, one can improve further upon these bounds by developing conditional speedups of $\mathsf{coNTISP}$ in $\Pi_k\mathsf{TIME}$, along the lines of Section 3.3. However, we do not feel that this exercise yields enough new insight into our techniques to be worth the reader's trouble— at best, the resulting improvements in the lower bound exponent are fractional.

4.3. Remark on Solving Tautologies With Nondeterminism. To end this section, we briefly discuss the problem of solving Boolean tautologies on nondeterministic machines with small space. Here, our methods do not appear as effective as what is already known. The best lower bound known[5] says that this problem requires at least $n^{\sqrt{2}-\varepsilon}$ time given $n^{o(1)}$ space, and is due to Fortnow and Van Melkebeek (7). We have found an alternative inductive

[5]The preliminary version of this paper in CCC'05 stated that its lower bound of $n^{1.337}$ was the best known. At the time, we did not realize that Fortnow and Van Melkbeek had proven a better result.

argument that also attains the $n^{\sqrt{2}}$ bound, but it does not take the form of the above work and so we omit it here.

After much effort, we have not found a way to improve upon the $n^{\sqrt{2}}$ lower bound with our techniques, although we strongly believe this to be possible. The proof of the result in (7) appears to rely critically on the hierarchy result $\mathsf{NTIME}[p(n)] \not\subseteq \mathsf{coNTIME}[o(p(n))]$, for *infinitely many polynomials* $p(n)$. However, our style of argument looks to be only truly effective when can formulate the contradictions in terms of inclusions among classes running in *linear* (or nearly linear) time.

5. Improving Time Lower Bounds for a Strong Form of Off-Line Turing Machine

The inductive method makes it also possible to strengthen time lower bounds for a type of Turing machine that is a hybrid between a random access machine and a one-tape machine, as described in Section 2.1. Recall this model has read-only random access to the input, an $n^{o(1)}$-space random-access storage, and an unbounded one-dimensional sequential-access worktape.

Previously, an $n^{\sqrt{3/2}-\varepsilon} \approx n^{1.22}$ time lower bound for SAT was provable for this machine model, using a simulation due to Maass and Schorr (15) (independently rediscovered recently by Van Melkebeek and Raz (16)). An $n^{\sqrt[4]{3/2}-\varepsilon} \approx n^{1.1}$ bound proved by Kannan (12) in 1983 for a more restricted machine model can be easily seen to hold for the above as well. Our improvement pushes the lower bound to greater than $n^{5/4}$. More precisely, we prove Theorem 1.4, which establishes

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTIME}_1[n^{1.268}].$$

Our main tool is a "speed up" simulation of DTIME_1, implicitly proven by Van Melkebeek and Raz. In the context of lower bounds, this speedup simulation represents item (2) in the four-step scheme described in Section 1.2. A speedup lemma of the kind we need was first proved by Maass and Schorr for sequential tapes, but examination of the proof shows that the result is not affected by the addition of a random access $n^{o(1)}$-space storage and adding random-access to the input tape.

Lemma 5.1. (Maass and Schorr (15))
For all $k \geq 1$, $\mathsf{DTIME}_1[t] \subseteq \Sigma_{2k}\mathsf{TIME}[t^{(k+1)/(2k+1)+o(1)}]$.

Later, Van Melkebeek and Raz independently proved a tighter version of this speedup. Note when $k = 1$, the two results are the same.

Lemma 5.2. (Implicitly in Van Melkbeek and Raz (16))
For all $k \geq 1$, $\mathsf{DTIME}_1[t] \subseteq \Sigma_{k+1}\mathsf{TIME}[t^{(k+1)/(2k+1)+o(1)}]$.

At a high level, the simulation of DTIME_1 works by guessing and verifying carefully chosen crossing sequences of the one-dimensional read-write tape, which can be described in fewer bits than the entire tape's content. Note by definition of the machine model, each crossing in a sequence (that is, each configuration of the rest of the machine) can be described in $n^{o(1)}$ bits, since the rest of the machine can be described by the tape heads, the state, and the content of the small-space storage. The speedup of DTIME_1 in Σ_{k+1} is used to prove a time lower bound for nondeterminism, just as the speedup of DTISP in Σ_k was used earlier.

Proof of Theorem 1.4. Suppose $\mathsf{NTIME}(n) \subseteq \mathsf{DTIME}_1[n^c]$. Then

$$\Pi_2\mathsf{TIME}[n] \subseteq \mathsf{coNTIME}[n^c] \subseteq \mathsf{DTIME}_1[n^{c^2}] \subseteq \Sigma_2\mathsf{TIME}[n^{\frac{2}{3}c^2+o(1)}].$$

Clearly there is a contradiction when $c < \sqrt{\frac{3}{2}}$ (the previously known lower bound). If this is not the case, then as before we can use the derived inclusion $\Pi_2\mathsf{TIME}[n] \subseteq \Sigma_2\mathsf{TIME}[n^{\frac{2}{3}c^2+o(1)}]$. The induction hypothesis becomes

$$\textit{For all } i = 2, \ldots, k-1, \;\; \Pi_i\mathsf{TIME}[n] \subseteq \Sigma_i\mathsf{TIME}[n^{h(i)+o(1)}],$$

where $h(1) := c$, $h(k+1) := \left(\frac{k+1}{2k+1}c^2\right)\prod_{i=1}^{k} h(i)$. Thus, applying Lemma 5.2,

$$\begin{aligned}\Pi_{k+1}\mathsf{TIME}[n] &\subseteq \Pi_k\mathsf{TIME}[n^{h(k+1)+o(1)}] \subseteq \cdots \subseteq \Pi_2\mathsf{TIME}[n^{\prod_{i=2}^{k} h(i)+o(1)}] \\ &\subseteq \mathsf{DTIME}_1[n^{c^2\prod_{i=2}^{k} h(i)+o(1)}] \\ &\subseteq \Sigma_{k+1}\mathsf{TIME}[n^{\frac{(k+1)}{2k+1}c^2\prod_{i=2}^{k} h(i)+o(1)}],\end{aligned}$$

so

$$\Pi_{k+1}\mathsf{TIME}[n] \subseteq \Sigma_{k+1}\mathsf{TIME}[n^{h(k+1)+o(1)}].$$

Simplifying as in previous cases, one can rewrite the recurrence for h to be

$$h(2) = \frac{2}{3}\cdot c^2, h(k+1) = \frac{2k-1}{k}\cdot\frac{k+1}{2k+1}\cdot h(k-1)^2.$$

Let $c_k \in (1,2)$ be such that $h(k) = 1$ when $c = c_k$. As $k \to \infty$, one can verify that $c_k > 1.2684$. □

To complete the rest of Table 1 for this machine model, we simply observe that the same analysis holds, except that the "base case" for e changes. For example, the proof of the lower bound for $\Sigma_2\mathsf{TIME}[n]$ results in the expression $h'(2) = \frac{2c}{3}$, $h'(k+1) = \frac{k+1}{k} \cdot \frac{2k-1}{2k+1} \cdot h'(k)^2$, yielding a $n^{1.609}$ lower bound.

As a final note to this section, it is not clear if one can obtain better conditional speedups of DTIME_1 in alternating time, similar to that in Section 3.3. The alternating simulation of DTIME_1 in Lemma 5.2 does not seem to admit as nice of a tradeoff between the sizes of quantifier segments.

Remark 5.3. *Van Melkebeek and Raz gave similar lower bounds for SAT on co-nondeterministic machines under the same model, as well as lower bounds when the sequential access tape is k-dimensional for constant k. Our method can in principle be applied to improve those bounds as well, due to existence of speedups via alternation.*

For example, when the sequential tape is two-dimensional, Van Melkebeek and Raz prove a $n^{\sqrt{4/3}-\varepsilon} = n^{1.154\ldots}$ time lower bound for solving SAT. The above argument can be used to modestly improve this lower bound to $n^{1.173\ldots}$, using the fact (again, derivable from Van Melkebeek and Raz's paper) that time t in the two-dimensional tape model can be simulated with $k+1$ alternations in $t^{\frac{2k+1}{3k+1}+o(1)}$ time.

6. Lower Bounds for Bounded Nondeterminism

We now turn to the problem of proving time-space lower bounds for machines with bounded nondeterminism. In particular, we prove Theorem 1.6 from the introduction, which states that for any $\varepsilon > 0$, there is a $c_\varepsilon > 1$ whereby $\mathsf{NTIBI}[n, n^\varepsilon]$ is not contained in $\mathsf{DTISP}[n^{c_\varepsilon}, n^{o(1)}]$. More precisely, the separation holds when $c_\varepsilon < (\varepsilon + \sqrt{\varepsilon^2+4})/2$.

Theorem 1.6 also holds when the classes are defined with respect to multitape Turing machines. For this reason, we define the classes $\mathsf{NTIBI}_M[t(n), b(n)]$ and $\mathsf{DTISP}_M[t(n), s(n)]$ to be the corresponding complexity classes for multitape Turing machines. Like the separation results for NTIME, Theorem 1.6 is also tied to lower bounds on an explicit problem, that of satisfying Boolean circuits with a relatively small number of inputs.

Corollary 1.7. *For all $k \geq 1$, there exists $c_k > 1$ such that Boolean satisfiability on circuits with n inputs and n^k gates requires $n^{k \cdot c_k}$ time on a deterministic multitape Turing machine using $n^{o(1)}$ space.*

Our results here do not use an inductive argument. Instead, they hinge primarily on the fact that an alternating simulation of DTISP (such as that appearing in Lipton-Viglas and Fortnow-Van Melkebeek) has the property that its last quantifier does not guess many bits. An anonymous reviewer pointed out that this particular property has been exploited before in a different manner– Fortnow *et al.* (8) used it to prove lower bounds on NTISP, rather than NTIBI. Their proofs are similar in nature to ours; the reader is referred to their paper for details.

Proof of Theorem 1.6. Assume there is an $\varepsilon > 0$, such that for all $c > 1$, $\mathsf{NTIBI}[n, n^\varepsilon] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$. Fix such a $c \in (1, 2)$, to be defined later. Choose $\varepsilon' \in (0, \min\{\varepsilon, c/2\})$ and $t(n) \geq n^{1/\varepsilon'}$.

By padding, $\mathsf{NTIBI}[t, t^\varepsilon] \subseteq \mathsf{DTISP}[t^c, t^{o(1)}]$ for all $c > 1$ and $t(n) \geq n$. Since DTISP is closed under complement, we have $\mathsf{coNTIBI}[t, t^\varepsilon] \subseteq \mathsf{DTISP}[t^c, t^{o(1)}]$ as well. By the proof of Lemma 2.7, a time t^c and space $t^{o(1)}$ machine M can be simulated by a Σ_2 machine that is defined by the sentence:

$$(\exists \text{ configurations } C_1, \ldots, C_{t^{\varepsilon'}} \text{ of } M \text{ on } x)(\forall i \in \{1, \ldots, t^{c-\varepsilon'} + 1\}) \\ [C_{i-1} \text{ leads to } C_i \text{ in } t^{c-\varepsilon'} \text{ time}],$$

where C_0 and $C_{t^{\varepsilon'}+1}$ are the initial and accept configurations, respectively.The $(\forall i \cdots)[\cdots]$ part of the Σ_2 sentence corresponds to a $\mathsf{coNTIBI}[N^{(c-\varepsilon')/\varepsilon'}, \log N]$ computation, where $N = t^{\varepsilon'} + n \in O(t^{\varepsilon'})$. Therefore the $(\forall i \cdots)[\cdots]$ part can be replaced by a $\mathsf{DTIME}[N^{c(c-\varepsilon')/\varepsilon'}]$ computation, by assumption (and the fact that $c - \varepsilon' > \varepsilon'$, since $\varepsilon' < c/2$).

Thus the computation represented by the Σ_2 sentence above is equivalent to a computation represented by the sentence:

$$(\exists \text{ configurations } C_1, \ldots, C_{t^{\varepsilon'}} \text{ of } M \text{ on } x)\ D(x, C_1, \ldots, C_{t^{\varepsilon'}}),$$

where D is a deterministic computation running in $t^{c(c-\varepsilon')}$ time. However, the above sentence corresponds to an $\mathsf{NTIBI}[t^{c(c-\varepsilon')} + t^{\varepsilon'}, t^{\varepsilon'+o(1)}]$ computation.

Hence we obtain

$$\mathsf{coNTIBI}[t, t^\varepsilon] \subseteq \mathsf{NTIBI}[t^{c(c-\varepsilon')} + t^{\varepsilon'}, t^{\varepsilon'+o(1)}].$$

Note that $\varepsilon' < c/2 < 1$, for $c < 2$. If $c(c-\varepsilon') < 1$, then we have a contradiction with Theorem 2.5, as this implies $\mathsf{coNTIBI}[t, t^\varepsilon] \subseteq \mathsf{NTIBI}[t^{1-\gamma}, t^{\varepsilon-\gamma}]$ for some $\gamma > 0$.

It remains for us to show that there is a $c > 1$ such that $c(c - \varepsilon') < 1$. Note $d(d - \varepsilon') = 1$ has the unique positive solution $d = (\varepsilon' + \sqrt{(\varepsilon')^2 + 4})/2$.

Therefore, for any $\varepsilon' \in (0,1)$ it follows that $d \in (1,2)$, so any $c \in (1,d)$ suffices. □

Notice in the above we needed that $c(c-\varepsilon') < 1$, $\varepsilon' < \varepsilon$, and $\varepsilon' < c/2$. Thus the maximum possible ε and c are $\varepsilon < 1/\sqrt{2}$ and $c < \sqrt{2}$. (To see this, note that c is maximized by making ε' as large as possible; change the inequalities to equations, and solve.)

The separation of NTIBI from DTISP has application to real satisfiability problems, as we now demonstrate.

LEMMA 6.1. *Fix $k \geq 1$. If Boolean satisfiability on circuits with N inputs and N^k gates and wires is in $\mathsf{DTISP}_M[n^c, n^{o(1)}]$ for some $c \geq 1$, then $\mathsf{NTIBI}_M[n, n^{1/k}]$ is contained in $\mathsf{DTISP}_M[n^c \cdot poly(\log n), n^{o(1)}]$.*

PROOF. (Sketch) Suppose the problem stated in the lemma is solvable in n^c time and $n^{o(1)}$ space, by multitape TM M_C. Let M be an arbitrary multitape TM using $n^{1/k}$ nondeterministic bits running in $O(n)$ time. Then there exists a deterministic linear time M' such that $M(x)$ accepts iff there is a y of length $|x|^{1/k}$ such that $M'(x,y)$ accepts. By the oblivious TM simulation of Pippenger and Fischer (21), there is a Boolean circuit $C_{M'}$ of $O(n \log n)$ gates such that $M'(x,y)$ accepts iff $C_{M'}(x,y) = 1$.

Let $C_{M',x}$ be $C_{M'}$ with the input bits of x hard-coded. An arbitrary bit in the description of the Pippenger-Fischer circuit $C_{M',x}$ can be computed in $O(poly(\log n))$ time and $O(\log n)$ space on a multitape machine, when the input tape head is on the proper bit of x (for a reference, cf. Fortnow *et al.* (8)). Thus $M(x)$ accepts if and only if there is an input y of length $|x|^{1/k}$ such that $C_{M',x}(y) = 1$.

Now we define a multitape M'' that, on input x, simulates $M_C(C_{M',x})$ and accepts if and only if the simulation does. More precisely, when the simulation of M_C requests a bit of $C_{M',x}$, it is computed in $O(poly(\log n))$ time and $O(\log n)$ space. By assumption on M_C and the fact that $C_{M',x}$ has $O(n \log n)$ gates, M'' runs in $O(n^c \cdot poly(\log n))$ time and $n^{o(1)}$ space. M'' is correct since

$$\begin{aligned} M(x) \text{ accepts} &\iff (\exists y : |y| \leq |x|^{1/k})[M'(x,y) \text{ accepts}] \\ &\iff (\exists y : |y| \leq |x|^{1/k})[C_{M',x}(y) = 1] \\ &\iff M_C(C_{M',x}) \text{ accepts} \iff M''(x) \text{ accepts.} \end{aligned}$$

□

Corollary 1.7 follows immediately from Theorem 1.6 and Lemma 6.1.

7. Conclusion

We have demonstrated an inductive method for proving lower bounds on concrete NP-complete problems, such as SAT, that utilizes properties of the polynomial hierarchy and existing tools. We also showed how existing lower bound arguments can get lower bounds on bounded nondeterminism classes, due to the $O(\log n)$-bit universal quantifier inside of a Σ_2TIME simulation of DTISP.

Our approach is extremely general. It is applicable to essentially any lower bound on Σ_kTIME where the class $\mathcal{C}$ (shown to be weak) can be sped up using a constant number of alternations, and this speedup improves when more alternations are used— a tradeoff between alternations and speedup must be possible. To illustrate the importance of the last point, observe that our approach cannot currently be used to improve Paul, Pippenger, Szemeredi, and Trotter's NTIME$[n] \neq$ DTIME$[n]$ result, since we do not know a generalization of DTIME$[t] \subseteq \Sigma_4$TIME$[t/\log^* t]$ where a machine using more than four alternations improves upon the $t/\log^* t$ time simulation of DTIME$[t]$. (However, we do know that DTIME$[t] \subseteq$ ATIME$[t/\log t]$ (11; 19), so such a tradeoff may be possible.)

Concerning further work in this area, it would be interesting to unify our techniques with those of Fortnow and Van Melkebeek. Their approach seems to be different from ours in both its capabilities and limitations. We can improve many of their results— however, they are able to prove an $\Omega(n^{\sqrt{2}})$ time lower bound for SAT on co-nondeterministic subpolynomial space machines and it seems we cannot do that with our method.

In closing, we believe that we have not yet completely exploited the full "power of inductive thinking" in our results. The conditional speedup of DTISP (Lemma 3.7) only works when we assume a subquadratic time algorithm. Thus it appears that the best time lower bound for SAT that we can obtain with the tools of this paper is $\Omega(n^2)$ (assuming subpolynomial space), but this is not totally clear. It may very well be that a proof of L $\neq$ NP could take a form similar to the kind of arguments we presented here. Results supporting or contradicting this suggestion would be of great interest.

Acknowledgements

This work was supported by the NSF ALADDIN Center (NSF Grant No. CCR-0122581), and appeared in preliminary form as (26). The author thanks his advisor Manuel Blum for his boundless encouragement and demand for clarity. Anupam Gupta, Richard Statman, Jon Kleinberg, Virginia Vassilevska, Maverick Woo, and anonymous referees gave extremely helpful comments that

significantly improved the paper's presentation. Finally, this work owes a debt to Dieter van Melkebeek, whose commentary and enthusiasm for this project were inspirational.

References

[1] L. Cai and J. Chen. On the Amount of Nondeterminism and the Power of Verifying. *SIAM J. Comput.* 26(3):733–750, 1997.

[2] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM* 28(1):114–133, 1981.

[3] A. K. Chandra and L. J. Stockmeyer. Alternation. In *Proceedings of IEEE FOCS* 98–108, 1976.

[4] A. Cobham. The recognition problem for the set of perfect squares. In *IEEE Symposium on Switching and Automata Theory*, 78–87, 1966.

[5] S. A. Cook. Short Propositional Formulas Represent Nondeterministic Computations. *Inf. Process. Lett.* 26(5): 269-270 (1988)

[6] L. Fortnow. Nondeterministic Polynomial Time Versus Nondeterministic Logarithmic Space: Time-Space Tradeoffs for Satisfiability. In *Proceedings of IEEE Conference on Computational Complexity*, 52–60, 1997.

[7] L. Fortnow and D. Van Melkebeek. Time-Space Tradeoffs for Nondeterministic Computation. In *Proceedings of the IEEE Conference on Computational Complexity*, 2–13, 2000.

[8] L. Fortnow, R. Lipton, D. Van Melkebeek, and A. Viglas. Time-Space Lower Bounds for Satisfiability. *Journal of the ACM* 52(6):835–865, 2005.

[9] Y. Gurevich and S. Shelah. Nearly Linear Time. In *Logic at Botik'89, Symposium on Logical Foundations of Computer Science*, Springer LNCS 363:108–118, 1989.

[10] J. Håstad. *Computational limitations for small depth circuits.* PhD Thesis, MIT Press, 1986.

[11] J. Hopcroft, W. J. Paul, L. Valiant. On Time vs. Space. *Journal of the ACM* 24(2):332–337, 1977.

[12] R. Kannan. Alternation and the power of nondeterminism. In *Proceedings of ACM STOC*, 344–346, 1983.

[13] R. Kannan. Towards Separating Nondeterminism from Determinism. *Mathematical Systems Theory* 17(1):29–45, 1984.

[14] R. J. Lipton and A. Viglas. On the Complexity of SAT. In *Proceedings of IEEE FOCS*, 459–464, 1999.

[15] W. Maass and A. Schorr. Speed-Up of Turing Machines with One Work Tape and a Two-Way Input Tape. *SIAM J. Comput.* 16(1):195–202, 1987.

[16] D. van Melkebeek and R. Raz. A Time Lower Bound for Satisfiability. In *Proceedings of ICALP*, 971–982, 2004.

[17] D. van Melkebeek. Time-Space Lower Bounds for NP-Complete Problems. In G. Paun, G. Rozenberg, and A. Salomaa (eds.), *Current Trends in Theoretical Computer Science*, 265–291, World Scientific, 2004.

[18] V. Nepomnjaŝciĭ. Rudimentary predicates and Turing calculations. *Soviet Mathematics Doklady*, 11:1462-1465, 1970.

[19] W. J. Paul and R. Reischuk. On Time versus Space II. (Turing Machines). *J. Comput. Syst. Sci.* 22(3):312–327, 1981.

[20] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On determinism versus nondeterminism and related problems. In *Proceedings of IEEE FOCS*, 429–438, 1983.

[21] N. Pippenger and M. J. Fischer. Relations Among Complexity Measures. *Journal of the ACM* 26(2):361–381, 1979.

[22] R. Santhanam. Lower bounds on the complexity of recognizing SAT by Turing machines. *Info. Proc. Lett.* 79(5):243–247, 2001.

[23] W. J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.* 4(2):177–192, 1970.

[24] C. P. Schnorr. Satisfiability is Quasilinear Complete in NQL. *J. of the ACM* 25(1):136–145, 1978.

[25] I. Tourlakis. Time-Space Tradeoffs for SAT on Nonuniform Machines. *J. Comput. Syst. Sci.* 63(2): 268–287, 2001.

[26] R. Williams. Better Time-Space Lower Bounds for SAT and Related Problems. In *Proceedings of the IEEE Conference on Computational Complexity*, 40–49, 2005.

Manuscript received October 12, 2006

Ryan Williams
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
ryanw@cs.cmu.edu