Automated Proofs of Time Lower Bounds

R. Ryan Williams*
Carnegie Mellon University

Abstract

A fertile area of recent research has demonstrated concrete polynomial time lower bounds for solving natural hard problems on restricted computational models. Among these problems are Satisfiability, Vertex Cover, Hamilton Path, MOD_6-SAT, Majority-of-Majority-SAT, and Tautologies, to name a few. These lower bound proofs all follow a certain diagonalization-based proof-by-contradiction strategy. A pressing open problem has been to determine how powerful such proofs can possibly be.

We propose an automated theorem-proving methodology for studying these lower bound problems. In particular, we prove that the search for better lower bounds can often be turned into a problem of solving a large series of linear programming instances. We describe an implementation of a small-scale theorem prover and discover surprising experimental results. In some settings, our program provides strong evidence that the best known lower bound proofs are already optimal for the current framework, contradicting the consensus intuition; in others, the program guides us to improved lower bounds where none had been known for years.

*Supported in part by the ALADDIN Center under NSF grant CCR-0122581. Email: `ryanw@cs.cmu.edu`.

1 Introduction

In this work, we show how to prove new limitations on computers by exploiting their capabilities. Many time lower bounds for hard problems follow a certain pattern that we call a *resource-trading scheme.* At an abstract level, the scheme consists of diagonalization-style arguments that use four basic steps:

1. Assume a hard problem Π can be solved in n^c time with resources R. (Let's abbreviate the class of such problems as $R[n^c]$.) We wish to obtain a contradiction. For example, R could represent algorithms using $\text{poly}(\log n)$ space, and Π could be the satisfiability problem.

2. Prove a *Speedup Lemma* that "trades time for resources", where $R[t]$ is shown to be in a class $S[o(t)]$, for a more powerful resource S. For example, S could represent alternating Turing machines. Fortnow-Van Melkebeek [FvM00] showed that $\text{poly}(\log n)$ space algorithms running in n^k time can be simulated by a Σ_k machine in $\tilde{O}(n)$ time.

3. Prove a *Slowdown Lemma* that "trades resources for time", where $S[t]$ is shown to be in $R[t^d]$, for small $d \geq 1$. A slowdown typically uses the assumption that Π is in $R[n^c]$. For example, if SAT has an n^c time, $\text{poly}(\log n)$ space algorithm, then all of $\mathsf{NTIME}[t]$ has $t^{c+o(1)}$ time, $\text{poly}(\log t)$ space algorithms, and as a consequence $\Sigma_k\mathsf{TIME}[t]$ has $t^{c^k+o(1)}$ time, $\text{poly}(\log t)$ space algorithms.

4. Combine (2) and (3) to show $\mathcal{C}[t] \subseteq \mathcal{C}[t^{1-\varepsilon}]$, for some $\varepsilon > 0$ and complexity class $\mathcal{C}$ parameterized by t, implying a contradiction with a time hierarchy theorem for $\mathcal{C}$. For example, we can derive $\Sigma_2\mathsf{TIME}[t] \subseteq \mathsf{DTISP}[t^{c^2+o(1)}, \text{poly}(\log t)] \subseteq \Pi_2\mathsf{TIME}[t^{c^2/2}]$,[1] where the first inclusion holds by (3) and the second holds by (2). This contradicts the alternating time hierarchy if $c^2 < 2$. The above is the $n^{\sqrt{2}-\varepsilon}$ lower bound of Lipton and Viglas [LV99].

This scheme has been applied to prove lower bounds in a enormity of settings, dating back to the seventies. A non-exhaustive list includes:

- **Time Versus Space.** Hopcroft, Paul, and Valiant [HPV77] proved that $\mathsf{SPACE}[n] \not\subseteq \mathsf{DTIME}[o(n \log n)]$ for multitape Turing machines, by proving the "speedup lemma" that $\mathsf{DTIME}[t(n)] \subseteq \mathsf{SPACE}[t(n)/\log t(n)]$ and invoking a diagonalization argument. (Note their result has since been extended to more general models of computation.)

- **Determinism versus Nondeterminism with Multitape TMs.** A celebrated result of Paul-Pippenger-Szemeredi-Trotter [PPST83] shows that $\mathsf{NTIME}[n] \subsetneq \mathsf{DTIME}[n(\log^* n)^{1/4}]$ in the multitape Turing machine setting. The key component of their proof is the "speedup lemma" that $\mathsf{DTIME}[t] \subseteq \Sigma_4\mathsf{TIME}[t/\log^* t]$, which exploits the limitations of sequential tapes.

- **Deterministic and Nondeterministic Space-Bounded Algorithms.** Here the model of computation is a general purpose random access machine (determininstic or nondeterministic) using small workspace (where $n/\text{poly}(\log n)$, $n^{1-\varepsilon}$, and $n^{o(1)}$ are the typical values) and the time lower bounds are for solving satisfiability and other NP-complete problems, as well as

[1] $\mathsf{DTISP}[t, s]$ is the class of sets recognizable by algorithms running in time t and space s, simultaneously.

problems higher up in the polynomial hierarchy [Kan84, For97, LV99, FvM00, DvM06, Wil06, Wil07a, vM07]. The best known deterministic time lower bound for solving satisfiability (and similar NP-complete problems) with $n^{o(1)}$ space algorithms is $n^{2\cos(\pi/7)-o(1)} \geq n^{1.801}$ [Wil07a], and the bound also holds for the counting problem MODm-SAT where m is a composite that is not a prime power. For complete problems higher up the polytime hierarchy, k-QBF (the problem of satisfying quantified Boolean formulas with at most k quantifier blocks) is known to require $\Omega(n^{k+1-\delta_k})$ time for deterministic $n^{o(1)}$ space algorithms [Wil07b], where $\delta_k < 0.2$ rapidly converges towards 0 as k grows. In the setting of nondeterministic algorithms using $n^{o(1)}$ space, the best known time lower bound known for natural co-NP problems (such as TAUTOLOGY) has been $n^{\sqrt{2}-o(1)}$, by Fortnow and Van Melkebeek [FvM00].

- **Probabilistic and Quantum Space-Bounded Algorithms.** Allender *et al.* [AKRRV01] showed that MAJ-MAJ-SAT requires $n^{1+\Omega(1)}$ time to solve on unbounded error machines that use $n^{1-\varepsilon}$ space, for $\varepsilon > 0$. Diehl and Van Melkebeek [DvM06] proved that for $k \geq 2$, k-QBF requires $\Omega(n^{k-o(1)})$ time with randomized two-sided error algorithms using $n^{o(1)}$ space. Viola [Vio07] has shown that 3-QBF requires $n^{1+\Omega(1)}$ time on Turing machines with a random access input tape and two-way read-write access to a random bit tape. Van Melkebeek and Watson [vMW07, vM07] have very recently shown how to adapt the result of Adleman *et al.* [ADH97] that $\mathsf{BQP} \subseteq \mathsf{PP}$ to extend Allender *et al.* to a time lower bound for solving MAJ-MAJ-SAT with quantum algorithms.

- **Hybrid One-Tape TMs.** Here the model of interest has a random access both a read-only input tape and a $n^{o(1)}$ read-write store, along with read-write access to a sequential tape. As this model generalizes both the widely studied off-line one-tape model and the $n^{o(1)}$ space-bounded RAM, it holds importance as the most powerful computational model known where we can still prove non-trivial lower bounds for SAT. A series of work [Kan83, MS87, vMR05, Wil06] has established time lower bounds for solving $\mathsf{NTIME}[n]$ problems in this model, the best bound being $\Omega(n^{1.26})$ for SAT.

While the overall strategy behind these results is certainly principled, the proofs of lower bounds under this framework have had an *ad hoc* feel. "Speedups" and "slowdowns" are often applied in carefully contrived ways, making it hard for one to build intuition about the proofs. After gaining experience with these problems, one gets a sense that the space of all possible proofs might be difficult to systematically explore.

The primary contribution of this work is to develop a meta-proof framework for alternation-trading proofs that is also practically implementable. We show that many proofs following the above scheme can be reformulated in such a way that the search for new lower bounds becomes a feasible problem that computers themselves can help us attack. Informally speaking, the "hard work" in these lower bound arguments can often be replaced by an automated search that solves a series of linear programming problems, each in polynomial time [Kha79, Kar84]. Additionally, the formalization of alternation-trading proofs also allows us to prove meta-results about how proofs relate to each other, and their limitations.

1.1 Main Results

This paper is a case study that describes an automated theorem-proving approach to lower bounds in three interesting scenarios. In all three cases, the resource being "traded" is that of alternations, so for the purposes of this paper we say that the proof objects are *alternation-trading proofs*. We formalize the components used in prior work and their relevant properties, with the following results.

1. **Deterministic Time-Space Lower Bounds for Satisfiability.** For several years it has been a folklore conjecture that one could use the current known tools to prove a quadratic time lower bound for SAT, against algorithms using $n^{o(1)}$ space. In Section 3, using a computer program that searches over all possible alternation-trading proofs up to a fixed length, we give empirical evidence that our time lower bound of $n^{2\cos(\pi/7)-o(1)}$ from previous work [Wil07a] is already optimal for the current framework. Thus it appears we have exhausted the power of existing techniques, and that the curious constant $2\cos(\pi/7)$ represents a challenging barrier to further progress. With our formalism, we also give a simple meta-proof (sans computer) that it will be impossible to push the lower bound to quadratic time with the current alternation-trading proof framework.

2. **Nondeterministic Time-Space Lower Bounds for Tautologies.** Tailoring our computer program to fit this setting (described in Section 4), it found a very short (eleven line) proof that improves upon Fortnow-Van Melkebeek's seven-year old bound, and the optimal proofs found were strongly suggestive of a particular inductive pattern. Extrapolating from this pattern, we are able to prove (on paper) that an $n^{4^{1/3}-o(1)} \geq n^{1.587}$ time lower bound holds, and this appears to be the best possible within the framework. After hearing of our eleven-liner, Diehl and Van Melkebeek also found the same lower bound along with a generalization, cf. [DvMW07]. Thus an automated theorem-proving approach can not only tell us when we have reached a dead end, but it can also rapidly guide us towards new improvements.

3. **Time Lower Bounds for Hybrid One-Tape TMs.** In this setting (described in Section 5), another adaptation of our automated approach uncovers highly regular behavior in the optimal proofs. From the empirical results we readily extract an $\Omega(n^r)$ time lower bound, where r is the root of the polynomial $12x^5 - 12x^4 - x^3 - 13x + 4x^2 + 2$ in the interval $(1, 2)$. Numerical approximations indicate that $r \approx 1.3009$. Again, this appears to the best possible with the current tools.

The lower bounds above also hold for many other natural NP-complete problems as well, since the only property of satisfiability (or tautologies) required is that every set in $\mathsf{NTIME}[n]$ (respectively, $\mathsf{coNTIME}[n]$) has sufficiently efficient reductions to the problem. Also, the linear programming approach is not limited to the above three lower bound scenarios; in principle, it can be applied to the league of lower bound problems discussed in Van Melkebeek's surveys [vM04, vM07]. We have chosen to present these particular cases because they are among the most interesting lower bound settings, and the results illustrate a diversity of structure in alternation-trading proofs.

The key to our automated approach is that we separate the *discrete choices* in an alternation-trading proof from the *real-valued choices*. Discrete choices consist of choosing whether to apply a speedup or slowdown at a given step, and which complexity class $\mathcal{C}[t]$ to try to use in the contradiction. In all three cases above, we prove that one can restrict the number of possible discrete choices that need to be made by writing proofs in a certain *normal form.* The significant consequence of normal form is that it lets us consider a fixed type of class for $\mathcal{C}[t]$. The real-valued choices come from the choices of classes to prove a contradiction with, and the speedup step: one must guess how to split up the runtime of a class using quantifiers. However, once all the discrete choices are made, it turns out that with some care we can formulate the remaining real-valued problem as an instance of linear programming, which can then be efficiently solved.

Unfortunately, we cannot efficiently search over *all* possible proofs, as the number of discrete choices increases exponentially with the number of lines in the proof (the number is $2^n/n^{\Omega(1)}$ for n-line proofs, with small hidden constants). Still, with normal form simplifications and the linear programming reduction, we can search a substantial portion of the proof space– it is a feasible task to search over *all* proofs of up to 20-something lines for the best possible lower bounds. From clear patterns found in those proofs, we can restrict the search space much further and look for longer proofs using a heuristic search. For example, the best 424 line proof of a time lower bound for SAT on $n^{o(1)}$ space algorithms gives an $\Omega(n^{1.8018})$ time lower bound, matching the $2\cos(\pi/7)$ exponent to within $n^{0.0002}$.

2 Preliminaries

We assume familiarity with standard concepts in complexity theory [Pap94], especially alternation [CKS81]. Our default computational model is the random access machine, construed broadly; the particular variants do not affect our results. All functions used to bound runtime and space are assumed to be time constructible within the appropriate space bounds.

We also use some non-standard notation that turns out to be very useful in our formalization. Recall that $\mathsf{DTISP}[t(n), s(n)]$ is the class of sets accepted by a RAM that runs in $t(n)$ time and $s(n)$ space, simultaneously [BM78]. For convenience, we use the notation

$$\mathsf{DTS}[t(n)] := \mathsf{DTISP}[t(n)^{1+o(1)}, n^{o(1)}]$$

to avoid negligible $o(1)$ factors in the exponent.

To properly formalize alternation-trading proofs, we introduce some notation for alternating complexity classes which include *input constraints* between alternations. Define $(\exists\ f(n))^b\mathcal{C}$ to be the class of languages recognized by some machine N that, on input x, writes a $f(n)^{1+o(1)}$ bit string y nondeterministically, copies at most $n^{b+o(1)}$ bits z of the tuple $\langle x, y\rangle$ deterministically (in $O(n^{b+o(1)})$ time), and finally feeds z as input to a machine from class $\mathcal{C}$. We refer to this behavior by saying that the *class $\mathcal{C}$ is constrained to n^b input.* We also define $(\exists\ f(n))\mathcal{C} := (\exists\ f(n))^{\max\{1,(\log f(n))/(\log n)\}}\mathcal{C}$; that is, the default is $n^b \leq O(f(n)^{1+o(1)} + n^{1+o(1)})$. The class $(\forall\ f(n))^b\mathcal{C}$ is defined similarly (with respect to co-nondeterministic machines). We say that the existential and universal phases of an alternating computation are *quantifier blocks*, to reflect the notation. When a machine recognizing a language in class $(\exists\ f(n))\mathcal{C}$ is guessing the $O(f(n))$ bits, we say that it is *in an existential quantifier.* Similarly, we define being *in a universal quantifier* for $(\forall\ f(n))\mathcal{C}$.

Therefore, an alternating class of the form

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]$$

with $Q_i \in \{\exists, \forall\}$ means that the input to the computation starting at the ith quantifier block is of length $n^{b_i+o(1)}$ for all $i = 1, \ldots, k$, and the input to the DTS computation has length $n^{b_{k+1}+o(1)}$. (Of course, the first quantifier block always has an input of length n.) It is crucial to keep track of the input lengths to quantifier blocks, since the $2\cos(\pi/7)$ time lower bound and prior lower bounds rely heavily on the fact that these inputs can be small in some cases.

2.1 A Very Short Introduction to Time-Space Lower Bounds

Here we give a brief overview of the tools that have been used to prove time-space tradeoff lower bounds. For now we focus on deterministic time lower bounds for satisfiability for algorithms using $n^{o(1)}$ space, as the other relevant lower bound problems use analogous tools and the $n^{o(1)}$ space case is especially simple to work with.

A SAT Fact. It is known that satisfiability of Boolean formulas in conjunctive normal form (Sat) 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 (Cook [Coo88], Schnorr [Sch78], Tourlakis [Tou01], Fortnow *et al.* [FLvMV05]) *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.*

Let $\mathcal{C}[t(n)]$ 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 one-tape Turing machine using time t.

The above theorem implies that if we can prove $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[t]$, then Sat is not in $\mathcal{C}[t]$ modulo polylogarithmic factors.

Corollary 2.1 *If $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[t(n)]$, then there is a $c > 0$ such that Sat is not contained in $\mathcal{C}[t(n) \cdot (\log t(n))^c]$.*

Hence we want to prove $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[n^c]$ for a large constant $c > 1$. Van Melkebeek and Raz observed that a similar corollary holds for any problem Π such that Sat reduces to Π under highly efficient reductions, *e.g.* Vertex Cover, Hamilton Path, 3-SAT, and Max-2-Sat. It follows that identical time lower bounds hold for these problems as well.

Speedups, Slowdowns, and Contradictions. Given that our goal is to prove $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$, how can we go about this? In an alternation-trading proof, we assume that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and attempt to establish a contradiction, by applying two lemmas in such a way that a time hierarchy is violated. One lemma (called the "speedup lemma") takes a $\mathsf{DTS}[t]$ class and places it in an alternating class with runtime $o(t)$; the other (called the "slowdown lemma") takes an alternating class with runtime t and places it in a class with one less alternation and runtime $O(t^c)$.

Lemma 2.1 (Speedup Lemma) *Let $a \geq 1$, $e \geq 0$ and $0 \leq x \leq a$. Then*

$$\mathsf{DTISP}[n^a, n^e] \subseteq (Q_1\ n^{x+e})^{\max\{1,x+e\}}(Q_2\ \log n)^{\max\{1,e\}}\mathsf{DTISP}[n^{a-x}, n^e],$$

for $Q_i \in \{\exists, \forall\}$ where $Q_1 \neq Q_2$. In particular,

$$\mathsf{DTS}[n^a] \subseteq (Q_1\ n^x)^{\max\{1,x\}}(Q_2\ \log n)^1\mathsf{DTS}[n^{a-x}].$$

Proof. Let M be a random access machine using n^a time and n^e space. To get a simulation of M having type $(\exists\ n^{x+e})^{\max\{1,x+e\}}(\forall\ \log n)^{\max\{1,e\}}\mathsf{DTISP}[n^{a-x}, n^e]$, the simulation $N(x)$ existentially guesses a sequence of configurations $C_1, \ldots, C_{n^x}$ of $M(x)$. It then appends the initial configuration to the beginning of the sequence and the accepting configuration to the end of the sequence. Then $N(x)$ universally guesses an integer $i \in \{0, \ldots, n^x\}$, 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-x} steps. It is easy to see that the simulation is correct. The input constraints on the quantifier blocks are satisfied, since after the universal guess, the input is only x, C_i, and C_{i+1}, which is of size $n + 2n^e \leq n^{\max\{1,e\}+o(1)}$. □

The Speedup Lemma dates back to work of Nepomnjascii [Nep70] and Kannan [Kan84]. Note that in the above alternating simulation, the input to the final DTISP computation is linear in $n + n^e$, *regardless of the choice of x.* This is a surprisingly subtle property that is exploited heavily in alternation-trading proofs. The Slowdown Lemma is straightforward:

Lemma 2.2 (Slowdown Lemma) *Let $a \geq 1$, $e \geq 0$, $a' \geq 0$, and $b \geq 1$. If $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^e]$, then for both $Q \in \{\exists, \forall\}$,*

$$(Q\ n^{a'})^b\mathsf{DTIME}[n^a] \subseteq \mathsf{DTISP}[n^{c\cdot\max\{a,a',b\}}, n^{e\cdot\max\{a,a',b\}}].$$

In particular, if $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$, then

$$(Q\ n^{a'})^b\mathsf{DTIME}[n^a] \subseteq \mathsf{DTS}[n^{c\cdot\max\{a,a',b\}}].$$

Proof. Let L be a problem in $(Q\ n^{a'})^b\mathsf{DTIME}[n^a]$, and let A be an algorithm satisfying $L(A) = L$. On an input x of length n, A guesses a string y of length $n^{a'+o(1)}$, then feeds an $n^{b+o(1)}$ bit string z to $A'(z)$, where A' is a deterministic algorithm that runs in n^a time. Since $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^e]$ and DTISP is closed under complement, by padding we have that $\mathsf{NTIME}[p(n)] \cup \mathsf{coNTIME}[p(n)] \subseteq \mathsf{DTISP}[p(n)^c, p(n)^e]$ for polynomials $p(n) \geq n$. Therefore A can be simulated by a deterministic algorithm B. Since the total runtime of A is $n^{a'+o(1)} + n^{b+o(1)} + n^a$, the runtime of B is $n^{c\cdot\max\{a,a',b\}+o(1)}$ and the space usage is similar. □

The final component of an alternation-trading proof is a time hierarchy theorem, the most general of which is the following, provable by a simple diagonalization.

Theorem 2.2 (Alternating Time Hierarchy) *For $k \geq 0$, for all $Q_i \in \{\exists, \forall\}$, $a'_i > a_i \geq 1$, and $b'_i > b_i \geq 1$,*

$$(Q_1\ n^{a_1})^{b_2} \cdots^{b_k} (Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}] \nsubseteq (R_1\ n^{a'_1})^{b'_2} \cdots^{b'_k} (R_k\ n^{a'_k})^{b'_{k+1}}\mathsf{DTS}[n^{a'_{k+1}}],$$

where $R_i \in \{\exists, \forall\}$ and $R_i \neq Q_i$.

Remark 1 *Are alternation-trading proofs relativizing? The Slowdown Lemma clearly relativizes, but the Speedup Lemma does not relativize in most oracle models, for the simple reason that the original machine runs longer than the (sped-up) host machine, and can therefore ask longer queries. This is typically the case in this area of work. For example, Paul-Pippenger-Szemeredi-Trotter's result that $\mathsf{NTIME}[n] \neq \mathsf{DTIME}[n]$ is non-relativizing: a powerful enough oracle makes the two classes equal. Therefore, we consider alternation-trading proofs to be in that rare genre of non-relativizing and non-naturalizing lower bounds.*[2]

Two Instructive Examples. In order to really understand alternation-trading proofs, it is necessary to consider some examples. The art behind their construction consists of finding the proper sequence of rules to apply, and the right settings of the parameter x in the Speedup Lemma.

1. In FOCS'99, Lipton and Viglas proved that SAT cannot be solved by algorithms running in $n^{\sqrt{2}-\varepsilon}$ time and $n^{o(1)}$ space, for all $\varepsilon > 0$. Their proof can be summarized as follows: by Theorem 2.1, the assumption that there is such a SAT algorithm implies that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ with $c^2 < 2$. Then

$$\begin{aligned}(\exists\ n^{2/c^2})(\forall\ n^{2/c^2})\mathsf{DTS}[n^{2/c^2}] &\subseteq (\exists\ n^{2/c^2})\mathsf{DTS}[n^{2/c}] \quad \text{(Slowdown Lemma)}\\ &\subseteq \mathsf{DTS}[n^2] \quad \text{(Slowdown Lemma)}\\ &\subseteq (\forall\ n)(\exists\ \log n)\mathsf{DTS}[n] \quad \text{(Speedup Lemma, with} x = 1.\end{aligned}$$

   But $(\exists\ n^{2/c^2})(\forall\ n^{2/c^2})\mathsf{DTS}[n^{2/c^2}] \subseteq (\forall\ n)(\exists\ \log n)\mathsf{DTS}[n]$ contradicts Theorem 2.2. In fact, one can show that if $c^2 = 2$, we still have a contradiction with $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$, so we can remove the ε from our previous statement and state that SAT cannot be solved in $n^{\sqrt{2}}$ time and $n^{o(1)}$ exactly.[3]

2. Improving on the previous example, one can show SAT is not in $\mathsf{DTS}[n^{1.6004}]$. If $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and $\sqrt{2} \leq c < 2$, then by applying the Speedup and Slowdown Lemmas appropriately,

[2]In fact, they are "non-algebrizing" as well.

[3]Suppose $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and $\mathsf{\Sigma_2TIME}[n] \subseteq \mathsf{\Pi_2TIME}[n^{1+o(1)}]$. The first assumption, along with the Speedup and Slowdown Lemmas, implies that for every k there's a K satisfying $\mathsf{\Sigma_2TIME}[n^k] \subseteq \mathsf{NTIME}[n^{kc}] \subseteq \mathsf{\Sigma_K TIME}[n]$. But the second assumption implies that $\mathsf{\Sigma_K TIME}[n] = \mathsf{\Sigma_2TIME}[n^{1+o(1)}]$. Hence $\mathsf{\Sigma_2TIME}[n^k] \subseteq \mathsf{\Sigma_2TIME}[n^{1+o(1)}]$, which contradicts the time hierarchy for $\mathsf{\Sigma_2TIME}$.

one can derive:

$$\begin{aligned}
\mathsf{DTS}[n^{c^2/2+2}] &\subseteq (\exists\ n^{c^2/2})(\forall\ \log n)\mathsf{DTS}[n^2] \\
&\subseteq (\exists\ n^{c^2/2})(\forall\ \log n)(\forall\ n)(\exists\ \log n)\mathsf{DTS}[n] \\
&= (\exists\ n^{c^2/2})(\forall\ n)(\exists\ \log n)\mathsf{DTS}[n] \\
&\subseteq (\exists\ n^{c^2/2})(\forall\ n)\mathsf{DTS}[n^c] \\
&\subseteq (\exists\ n^{c^2/2})\mathsf{DTS}[n^{c^2}] \\
&\subseteq (\exists\ n^{c^2/2})(\exists\ n^{c^2/2})(\forall\ \log n)\mathsf{DTS}[n^{c^2/2}] \\
&= (\exists\ n^{c^2/2})(\forall\ \log n)\mathsf{DTS}[n^{c^2/2}] \\
&\subseteq (\exists\ n^{c^2/2})\mathsf{DTS}[n^{c^3/2}] \\
&\subseteq \mathsf{DTS}[n^{c^4/2}]
\end{aligned}$$

Now, when $c^2/2 + 2 > c^4/2$ (which happens for $c < 1.6004$) we have $\mathsf{DTS}[n^a] \subseteq \mathsf{DTS}[n^{a'}]$ for some $a > a'$. Notice that we do not know if $\mathsf{DTS}[n^a] \not\subseteq \mathsf{DTS}[n^{a'}]$ when $a' > a$, as the space bounds on both sides of the inequality are the same. However one can still show by a translation argument (similar to the footnote) that either $\mathsf{DTS}[n^a] \not\subseteq \mathsf{DTS}[n^{a'}]$ or $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$, concluding the proof.

While the second example is somewhat clever in its structure, the most interesting aspect of this proof is that it was *found by a computer program.* By "found", we mean that the program applied the Speedup and Slowdown Lemmas in precisely the same order and with precisely the same parameter settings, having only minimum knowledge of these Lemmas along with a way to check the validity of the parameters. Moreover, the program verified that the above proof is the *best possible* alternation-trading proof that applies the Speedup and Slowdown Lemmas for at most seven times. A precise definition of "alternation-trading proof" will be given in the next section, along with a description of the program and its experimental results.

3 Automated Time-Space Lower Bounds for Satisfiability

We start our study with time-space lower bounds for nondeterministic linear time problems, such as satisfiability. In this setting we shall give a highly detailed description of our methodology. As all three settings have similar features, our treatment of the other two is far briefer and assumes some knowledge of this section.

As mentioned earlier, every lower bound proof by alternation-trading applies a sequence of "speedups" and "slowdowns" in some order. We formalize the notion of an alternation-trading proof in the following definition.

Definition 3.1 *Let $c > 1$. An* alternation-trading proof for c *is a list of complexity classes of the form*

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}],$$

where k is a non-negative integer, $Q_i \in \{\exists, \forall\}$, $Q_i \neq Q_{i+1}$, $a_i > 0$, and $b_i \geq 1$, for all i. (When $k = 0$, the class is deterministic.) The items of the list are called lines of the proof. *Each line* *is obtained from the previous line by applying either a* speedup rule *or a* slowdown rule. *More precisely, if the ith line is*

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}],$$

then the $(i+1)$*st line has one of three possible forms:*

1. (Speedup Rule 0) *If* $k=0$ *(*i.e. *the ith line is a deterministic class), then*
$$(Q_k\ n^x)^{\max\{x,1\}}(Q_{k+1}\ n)^1\mathsf{DTS}[n^{a_{k+1}-x}],$$
for some $x \in (0, a_{k+1})$.[4]

2. (Speedup Rule 1) *If* $k>0$*, then*
$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x\}})^{\max\{x,b_{k+1}\}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}],$$
for some $x \in (0, a_{k+1})$.

3. (Speedup Rule 2) *If* $k>0$*, then*
$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^x)^{\max\{x,b_{k+1}\}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}],$$
for some $x \in (0, a_{k+1})$.

4. (Slowdown Rule) *If* $k>0$*, then*
$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_{k-1}}(Q_{k-1}\ n^{a_{k-1}})^{b_k}\mathsf{DTS}[n^{c\cdot\max\{a_{k+1},a_k,b_k,b_{k+1}\}}].$$

We say that an alternation-trading proof shows $(\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies A_1 \subseteq A_2)$ *if its first line is* A_1 *and its last line is* A_2.

The justification for the definition comes directly from the proofs of the Speedup Lemma (Lemma 2.1) and the Slowdown Lemma (Lemma 2.2). For instance, Speedup Rule 1 holds, since

$$\begin{aligned}
&(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]\\
\subseteq\ &(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_k\ n^x)^{\max\{b_{k+1},x\}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]\\
\subseteq\ &(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x\}})^{\max\{b_{k+1},x\}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}],
\end{aligned}$$

since two quantifier blocks of the same type can always be "merged." Rule 2 is akin to Rule 1, except that it uses opposite quantifiers in its invocation of the Speedup Lemma. The Slowdown Rule works analogously to the Slowdown Lemma (Lemma 2.2). It follows that alternation-trading proofs are *sound*, in that an alternation-trading proof of $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies A_1 \subseteq A_2$ is indeed a proof that the implication is true.

Note that Speedup Rules 0 and 2 add two quantifier blocks to the line, while Speedup Rule 1 adds only one new quantifier block, and all of them introduce a new real-valued parameter x that must be determined. We can prove that it is actually never beneficial to apply Speedup Rule 2, and so it does not hurt us to remove it from consideration. To prove this we first need a lemma relating the a_i's and b_i's of an alternating class.

[4]The astute reader will notice that the $(k+1)$th quantifier block in the new class is larger than what the Speedup Lemma needs: here it is n, while for the Speedup Lemma it was only $\log n$. With the rules we are using, this distinction does not matter.

Definition 3.2 *A class* $(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]$ *is* orderly *if it is either (a) a* $\mathsf{DTS}[n^a]$ *class, or (b) for all* $i = 1, \ldots, k$, $a_i \le b_{i+1}$.

Lemma 3.1 *Suppose A_1 is orderly. Then every alternation-trading proof beginning with A_1 consists of only orderly classes.*

Proof. By induction on the number of lines. The base case is trivial. We hypothesize that the ℓth line given by

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]$$

is orderly. For the $(\ell+1)$th line, we consider the rules in turn:

- (Speedup Rule 0) Clearly $(Q_k\ n^x)^{\max\{x,1\}}(Q_{k+1}\ n)^1\mathsf{DTS}[n^{a_{k+1}-x}]$ is orderly.
- (Speedup Rule 1) Suppose the line is

  $$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x\}})^{\max\{x,b_{k+1}\}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

  Then $a_k \le b_{k+1}$ by the induction hypothesis, so $\max\{a_k, x\} \le \max\{x, b_{k+1}\}$, $1 \le b_{k+1}$, thus the class is orderly.
- (Speedup Rule 2) This case is clear, as the line is:

  $$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^x)^{\max\{x,b_{k+1}\}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$
- (Slowdown Rule) Obvious, given the hypothesis.

This concludes all the cases. □

Lemma 3.2 *Let A_1 be orderly. For every alternation-trading proof of* $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow A_1 \subseteq A_2$*, there is another alternation-trading proof of the same that does not use Speedup Rule 2.*

Proof. Consider a proof P that applies Speedup Rule 2 at some line. By definition, that line has the form

$$A = (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^x)^{\max\{x,b_{k+1}\}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

We consider two cases:

1. If $x \le a_k$, then $x \le b_{k+1}$ by Lemma 3.1. By applying Speedup Rule 2, one obtains

   $$\begin{aligned} A &= (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^x)^{\max\{x,b_{k+1}\}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}] \\ &= (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^x)^{b_{k+1}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}]. \end{aligned}$$

   If we instead apply Speedup Rule 1 with $x' = x$, the class is

   $$\begin{aligned} B &= (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x'\}})^{\max\{x',b_{k+1}\}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x'}] \\ &= (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}]. \end{aligned}$$

Then by applying Speedup Rule 1 with $x' = 0$, the above class is in

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n)^{b_{k+1}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

It is clear that $B \subseteq A$: every parameter in B is at most the corresponding parameter in A. Thus any inclusion derived with Rule 2 could only be made stronger by applying Rule 1 twice instead.

2. If $x \geq a_k$, then Speedup Rule 2 gives

$$A = (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^{x})^{\max\{x,b_{k+1}\}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

Speedup Rule 1 with $x' = a_k$ gives

$$B = (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x'\}})^{\max\{x',b_{k+1}\}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x'}].$$
$$= (Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-a_k}].$$

where we used the fact that $x' = a_k \leq b_{k+1}$ (Lemma 3.1). Applying Speedup Rule 1 again with $x' = x - a_k$, B is contained in

$$(Q_1\ n^{a_1})^{b_2}\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^{\max\{x-a_k,1\}})^{\max\{x-a_k,b_{k+1}\}}(Q_{k+2}\ n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

Again, observe that $B \subseteq A$ in this case, and every parameter in B is at most the corresponding parameter in A.

This completes the proof. □

In the next section, we shall actually show that every alternation-trading proof can in fact be rewritten so that the first line of the proof is a DTS class, and therefore all lines of the constituent proof are orderly. Because of this, from now on we just refer to the *Speedup Rule* as Speedup Rule 0 or Speedup Rule 1, depending on which applies to the situation.

3.1 A normal form for alternation-trading proofs

Lower bound proofs using the alternation-trading scheme apply the assumption $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ to derive a contradiction to a time hierarchy theorem along the lines of Theorem 2.2. To consider all possible ways to derive a time hierarchy contradiction (at least, between alternating, nondeterministic, and deterministic classes), we consider the most general setting of deriving a contradiction from *complementary alternating classes.*

Definition 3.3 *We say that A_1 and A_2 are* complementary alternating classes *if A_1 is the complement of A_2.*

Every known time-space lower bound for SAT proves $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $A_1 \subseteq A_2$, for some complementary alternating classes A_1 and A_2. A similar claim holds for nondeterministic time-space lower bounds against tautologies (all of which prove $\mathsf{NTIME}[n] \subseteq \mathsf{coNTS}[n^c]$ implies $A_1 \subseteq A_2$), and lower bounds for hybrid one-tape machines (which prove $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_h[n^c]$ implies $A_1 \subseteq A_2$). We now introduce a restriction to the space of alternation-trading proofs we consider, calling it a "normal form." Our intent is to show that any lower bound provable with complementary alternating classes can also be proved with a normal form proof, so it suffices for us to explore normal form proofs. This simplification will greatly reduce our proof search space in the future.

Definition 3.4 *Let $c \geq 1$. An alternation-trading proof for c is in* normal form *if*

- *The first and last lines are* $\mathsf{DTS}[n^a]$ *and* $\mathsf{DTS}[n^{a'}]$ *respectively, for some $a \geq a'$.*
- *No other lines are* DTS *classes.*

First we show that a normal form proof for c implies that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$.

Lemma 3.3 *Let $c \geq 1$. If there is an alternation-trading proof for c in normal form having at least two lines, then* $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$*.*

Proof. Let P be an alternation-trading proof for c in normal form. We consider two cases.

- Suppose $a > a'$. In this case, $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $\mathsf{DTS}[n^a] \subseteq \mathsf{DTS}[n^{a-\delta}]$ for some $\delta > 0$. By translation, $\mathsf{DTS}[n^a] \subseteq \mathsf{DTS}[n^{a-\delta}]$ implies

  $$\mathsf{DTS}[n^{a^2/(a-\delta)}] \subseteq \mathsf{DTS}[n^a] \subseteq \mathsf{DTS}[n^{a-\delta}],$$

  and $\mathsf{DTS}[n^{a\cdot(a/(a-\delta))^i}] \subseteq \mathsf{DTS}[n^{a-\delta}]$ for all $i \geq 0$. Since $\delta > 0$, this implies $\mathsf{DTS}[n^L] \subseteq \mathsf{DTS}[n^{a-\delta}]$ for all $L \geq a - \delta$. Therefore, if $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ then for all $L \geq a$,

  $$\mathsf{NTIME}[n^L] \subseteq \mathsf{DTS}[n^{Lc}] \subseteq \mathsf{DTS}[n^{a-\delta}] \subseteq \mathsf{coNTIME}[n^{a-\delta}],$$

  a contradiction to the time hierarchy (Theorem 2.2).

- Suppose $a = a'$. Let A be a line in P with a positive number of alternations. (Such a line must exist since P has at least two lines.) The proof P shows that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $\mathsf{DTS}[n^a] \subseteq A \subseteq \mathsf{DTS}[n^{a'}]$, so $A = \mathsf{DTS}[n^a]$.

  Since $\mathsf{DTS}[n^a]$ is closed under complement,

  $$A = A', \tag{1}$$

  where A' is the complement of A. Without loss of generality, assume $A = (\exists\ n^\delta)B$ and $A' = (\forall\ n^\delta)B'$ for some $\delta > 0$ and complementary classes B and B'. It is easy to see that

  $$A' = (\forall\ n^\delta)A' \text{ and } A = (\exists\ n^\delta)A. \tag{2}$$

  Now consider the class $\mathsf{DTS}[n^{\delta\lceil\frac{k}{\delta}\rceil}] \supseteq \mathsf{DTS}[n^k]$, for arbitrary $k \geq 1$. By the Speedup Lemma (Lemma 2.1) and the fact that $\mathsf{DTS}[n^\varepsilon] \subseteq A'$ for some $\varepsilon > 0$,

  $$\mathsf{DTS}[n^k] \subseteq \mathsf{DTS}[n^{\delta\lceil\frac{k}{\delta}\rceil}] \subseteq \underbrace{(\exists\ n^\delta)(\forall\ n^\delta)\cdots(\exists\ n^\delta)(\forall\ n^\delta)}_{\lceil k/\delta\rceil} A'.$$

Applying equations (1) and (2), we have

$$\begin{gathered}(\exists\ n^{\delta})(\forall\ n^{\delta})\cdots(\exists\ n^{\delta})(\forall\ n^{\delta})A' \\ = (\exists\ n^{\delta})(\forall\ n^{\delta})\cdots(\exists\ n^{\delta})A' \\ = (\exists\ n^{\delta})(\forall\ n^{\delta})\cdots(\exists\ n^{\delta})A \\ = (\exists\ n^{\delta})(\forall\ n^{\delta})\cdots A \\ = \cdots = (\exists\ n^{\delta})(\forall\ n^{\delta})A' = (\exists\ n^{\delta})A' = (\exists\ n^{\delta})A = A.\end{gathered}$$

Therefore $\mathsf{DTS}[n^k] \subseteq A$, for *every* $k \geq 1$. Hence $\mathsf{NP} \subseteq \mathsf{DTS}[n^{O(1)}, n^{o(1)}] \subseteq A$. But by applying a slowdown step for a finite number of times to A, there is an alternation-trading proof that $A \subseteq \mathsf{DTS}[n^K]$ for a constant K. It follows that $\mathsf{NP} \subseteq A \subseteq \mathsf{DTS}[n^K] \subseteq \mathsf{coNTIME}[n^K]$, contradicting the time hierarchy (Theorem 2.2). So $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$ in this case as well.

□

We now prove that any alternation-trading proof that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies A_1 \subseteq A_2$ for complementary alternating classes A_1 and A_2 can be converted into an analogous normal form proof. Therefore, to find good lower bounds it suffices for us to find good normal form proofs.

Theorem 3.1 *Let A_1 and A_2 be complementary. If there is an alternation-trading proof P for c that shows* ($\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies A_1 \subseteq A_2$)*, then there is a normal form proof for c, of length at most that of P.*

Proof. Consider an alternation-trading proof P for c, written as

$$P = A_1, C_1, \ldots, C_k, A_2.$$

Define the *dual proof* P' by

$$P' = A_2, \neg C_1, \ldots, \neg C_k, A_1,$$

where the notation $\neg C$ denotes the unique complementary alternating class for C, *i.e.* every '$\forall$' in C is replaced with '$\exists$', and vice-versa. Note that P' is an alternation-trading proof if and only if P is one.

Since the quantifiers of the first and last line of P are different, there must be a line $C_i = \mathsf{DTS}[n^a]$ for some a.

- Suppose there is only one deterministic class in P; call it C_i. Then

$$P'' = C_i, C_{i+1}, \ldots C_k, A_2, \neg C_1, \ldots, \neg C_i$$

  is also an alternation-trading proof, obtained by piecing together the appropriate lines from P and P'. However, $C_i = \neg C_i$, since $\mathsf{DTS}[n^a]$ is closed under complement. Hence P'' is in normal form: its first and last lines are DTS classes, and no intermediate class is a DTS class.

- Suppose there are $k \geq 2$ different DTS classes in P. Write P as:

$$P = A_1, \ldots, \mathsf{DTS}[n^{a_1}], \ldots, \mathsf{DTS}[n^{a_2}], \ldots, \ldots, \mathsf{DTS}[n^{a_k}], \ldots, A_2.$$

  There are two cases:

- If there is an $i \in [k]$ satisfying $a_i \geq a_{i+1}$, we are done: simply take P'' to be the sequence of lines from $\mathsf{DTS}[n^{a_i}]$ and $\mathsf{DTS}[n^{a_{i+1}}]$ to be the normal form proof.
- If $a_i < a_{i+1}$ for every i, then set $P'' = \mathsf{DTS}[n^{a_k}], \ldots, A_2, \ldots, \mathsf{DTS}[n^{a_1}]$, where the classes in the first "…" in P'' are taken directly from P, and the classes in the second "…" in P'' are gotten by taking the lines $A_2, \ldots, \mathsf{DTS}[n^{a_1}]$ in P'. P'' is in normal form since $a_k > a_1$.

□

N.B. For the remainder of this section, we assume that all alternation-trading proofs under discussion are in normal form.

3.2 Proof annotations

Different lower bound proofs can result in quite different sequences of speedups and slowdowns. A *proof annotation* represents such a sequence.

Definition 3.5 *A* proof annotation *for an alternation-trading proof of ℓ lines is the unique $(\ell-1)$-bit vector A that has $A[i] = 1$ (respectively, $A[i] = 0$) if the ith line is obtained by a Speedup Rule (respectively, a Slowdown Rule), for all $i = 1, \ldots, \ell - 1$.*

By definition, an $(\ell-1)$-bit proof annotation corresponds to a proof strategy for an ℓ-line proof. For a normal form alternation-trading proof with ℓ lines, its proof annotation A must have $A[1] = 1$, and $A[\ell-1] = 0$. In fact, $A[\ell-2] = 0$ as well – otherwise the last line would have at least one quantifier block. Thus every proof annotation begins with a 1 and ends with two 0's.

How many possible proof annotations are there? The number is closely related to the number of well-balanced strings over parentheses. For a string $x = x_1 \cdots x_{\ell-1}$ with $x_i \in \Sigma$, define $x[i..j] := x_i x_{i+1} \cdots x_j$ for $i \leq j$.

Definition 3.6 *Let $n > 0$ be an integer and $x \in \{\texttt{(},\ \texttt{)}\}^n$. x is* well-balanced *if*

- *for all $i = 1, \ldots, n-1$, the number of* (*'s in $x[1..i]$ is greater than the number of*) *'s, and*
- *the number of* (*'s in x equals the number of*) *'s in x.*

Recall that the kth Catalan number is $C(k) = \frac{1}{k+1}\binom{2k}{k}$. A well-known fact in combinatorics states that the number of well-balanced strings of a given length can be counted with the Catalan numbers.

Fact 3.1 *The number of well-balanced strings of length $2k$ is $C(k)$.*

Proposition 1 *Let $\ell > 3$ be even. The number of possible annotations for proofs of ℓ lines is $C(\ell/2 - 1)$.*

Proof. Consider an $(\ell-1)$-bit vector $A = [1, \ldots, 0]$. The first 1 introduces two quantifier blocks in line 1 of the corresponding proof. All subsequent 1's introduce only one quantifier block. All

0's remove one quantifier. So in order for A to count as a proof annotation, it must be that the number of 1's in any prefix of A is greater than (or equal to) the number of 0's, up to the last line, in which the number of 0's becomes the number of 1's plus one. (Recall that in normal form, only the first and last lines of a proof are DTS classes.)

Given these observations, it is not hard to see that every proof annotation can be expressed as a well-balanced string of the form $(x)y$, where x and y are well-balanced strings such that $|x| + |y| = \ell - 4$, corresponding to the bit vector $[1, x', 0, y', 0]$, where x' and y' are the strings gotten by replacing '(' with '1' and ')' with '0' in x and y, respectively. But the number of such well-balanced strings is $C(\ell/2 - 1)$, for even $\ell > 3$: *every* well-balanced string of length $\ell - 2$ can be written in the form $(x)y$, where $|x| + |y| = \ell - 4$, and by the fact, the number of such strings is precisely $C(\ell/2 - 1)$. $\square$

Corollary 3.1 *Let $\ell > 3$ be even. The number of possible annotations for proofs of ℓ lines is $\Theta(2^\ell/\ell^{3/2})$.*

Proof. By the previous proposition, the number is

$$C(((\ell/2) - 1) = \frac{2}{\ell}\binom{\ell-2}{\ell/2-1} = \Theta\left(\frac{1}{\ell}\cdot\frac{2^\ell}{\ell^{1/2}}\right),$$

where the last equality follows from a standard estimate. $\square$

That is, the number of n-bit proof annotations is only a poly(n)-fraction of the total number of n-bit strings.

Note that a proof annotation in itself does not determine the proof entirely– each application of a speedup rule introduces a new real-valued parameter x that must be set to some value. The problem of determining optimal values for these parameters shall be tackled in the next section.

To illustrate the proof annotation concept, let us consider four examples where the first two correspond to proofs that were discussed in the Preliminaries.

- The $n^{\sqrt{2}}$ lower bound of Lipton and Viglas uses the only possible 3-bit annotation: $[1, 0, 0]$.
- The $n^{1.6004}$ lower bound discussed in the Preliminaries has annotation $[1, 1, 0, 0, 1, 0, 0]$.
- The n^ϕ lower bound of Fortnow and Van Melkebeek is an inductive proof, corresponding to an infinite sequence of proof annotations. In normal form, the sequence looks like:

$$[1,0,0], [1,1,0,0,0], [1,1,1,0,0,0,0], [1,1,1,1,0,0,0,0,0], \ldots$$

- The $n^{2\cos(\pi/7)}$ lower bound contains two stages of induction, and this is reflected in its proof annotations. Let $A = 1, 0, 1, 0, \ldots, 1, 0, 0$, where the '$\ldots$' contain some arbitrarily large number of repetitions of $1, 0$. The sequence of annotations is given by:

$$[A], [1, A, A], [1, 1, A, A, A], [1, 1, 1, A, A, A, A], \ldots$$

  That is, the proof performs many speedups, then a sequence of many slowdown-speedup alternations, then two consecutive slowdowns, repeating this until all the quantifiers have been removed.

3.3 Translation Into Linear Programming

We have defined alternation-trading proofs, given a nice normal form for them, and have shown how the strategies of prior lower bounds can be expressed succinctly in terms of proof annotations. A potential plan for finding new lower bounds is to try searching over all proof annotations to see which is best. In order for this plan to be sensible, we need a way to efficiently determine the best proof that can be obtained with a given annotation. The primary issue here is that there are multiple real-valued parameters that need to be set in a particular way to yield a good lower bound. We describe how to take any proof annotation A and constant $c > 1$, and formulate an instance of linear programming that has a feasible solution if and only if there is an alternation-trading proof of $\mathsf{NTIME}[n] \nsubseteq \mathsf{DTS}[n^c]$ that follows the steps of annotation A.

Let A be a proof annotation of $\ell - 1$ bits and let $c > 1$. Let m be the maximum number of quantifier blocks in any line of A; note m can be computed in linear time by reading the bits of A. The corresponding linear programming instance has variables $a_{i,j}$, $b_{i,j}$, and x_i, for all $i = 0, \ldots, \ell-1$ and $j = 1, \ldots, m$.[5] These variables have the interpretation:

$a_{i,j}$ is the exponent for the runtime of the jth quantifier block in the class on the ith line,

$b_{i,j}$ is the exponent for the input to the jth quantifier block of the class on the ith line,

and for all lines i that use a Speedup Rule,

x_i is the choice of x in the use of the Speedup Rule.

For example:

- If the kth line of a proof is $\mathsf{DTS}[n^a]$, the corresponding constraints for that line are

$$a_{k,1} = a, \quad b_{k,1} = 1, \quad (\forall k > 0)\ a_{k,i} = b_{k,i} = 0.$$

- If the kth line of a proof is $(\exists\ n^{a'})^b \mathsf{DTS}[n^a]$, then the constraints are

$$a_{k,0} = a, \quad b_{k,1} = b, \quad a_{k,1} = a', \quad b_{k,1} = 1, \quad (\forall k > 1)\ a_{k,i} = b_{k,i} = 0.$$

The objective function of the linear program shall be to minimize $\sum_{i,j} a_{i,j} + b_{i,j}$. We now describe the constraints of the linear programming instance, along with justification for their correctness. In a sentence, the problem of determining valid x_i's can be viewed as a circuit evaluation with gates of type max, $+$, and multiplication by a fixed constant, and it is well known that such circuits can be evaluated using a linear program. For example, we can effectively replace $c = \max\{a, b\}$ by $c \geq a$, $c \geq b$, keeping in mind that our obtained value for c may be larger than necessary. For every line i, we include the constraints

$$(\forall j:\ 1 \leq j \leq m-1)\ a_{i,j} \leq b_{i,j+1}$$

to enforce the orderly constraint implied by Lemma 3.1.[6]

[5] We start the numbering of lines at 0, so that at the ith line it follows that i rules have been applied.

[6] These constraints are not strictly necessary, but they are useful in guiding an LP solver.

Initial constraints. For the 0th and $(\ell-1)$th lines, we have the constraints

$$a_{0,1} \geq 1,\ b_{0,1} = 1,\ \ (\forall\ k > 1)\ a_{0,k} = b_{0,k} = 0,\ \text{and}$$
$$a_{\ell,1} \geq 1,\ b_{\ell,1} = 1,\ \ (\forall k > 1)\ a_{\ell,k} = b_{\ell,k} = 0,$$

representing the classes $\mathsf{DTS}[n^{a_{0,1}}]$ and $\mathsf{DTS}[n^{a_{\ell-1,0}}]$, respectively. The constraint $a_{0,1} \geq a_{\ell-1,1}$ is also included, to ensure that the proof is in normal form. The first line of a proof is always an application of the first Speedup Rule, being of the form

$$(Q_1 n^x)^{\max\{x,1\}}(Q_2\ n)^1\mathsf{DTS}[n^{a-x}].$$

The corresponding LP constraints for the first line are therefore:

$$\begin{gathered} a_{1,1} = a_{0,1} - x_1,\ b_{1,1} = 1, \\ a_{1,2} = 1,\ b_{1,2} \geq x_1,\ b_{1,2} \geq 1, \\ a_{1,3} = x_3,\ b_{1,3} = 1 \\ (\forall\ k:\ 4 \leq k \leq m)\ a_{1,k} = b_{1,k} = 0. \end{gathered}$$

Speedup Rule constraints. For the ith line where $i > 1$ and $A[i] = 1$, the constraints are

$$\begin{gathered} a_{i,1} \geq 1,\ a_{i,1} \geq a_{i-1,1} - x_i,\ b_{i,1} = b_{i-1,1} \\ a_{i,2} = 1,\ b_{i,2} \geq x_i,\ b_{i,2} \geq b_{i-1,1}, \\ a_{i,3} \geq a_{i-1,2},\ a_{i,3} \geq x_i,\ b_{i,3} \geq b_{i-1,2} \\ (\forall\ k:\ 4 \leq k \leq m)\ a_{i,k} = a_{i-1,k-1}, b_{1,k} = b_{i-1,k-1}. \end{gathered}$$

These constraints express that

$$\cdots\ {}^{b_2}(Q_2\ n^{a_2})^{b_1}\mathsf{DTS}[n^{a_1}]$$

in the ith line is replaced with

$$\cdots\ {}^{b_2}(Q_2\ n^{\max\{a_2,x\}})^{\max\{x,b_1\}}(Q_1\ n)^{b_1}\mathsf{DTS}[n^{\max\{a_1-x,1\}}]$$

in the $(i+1)$st line, where Q_1 is opposite to Q_2.

Slowdown Rule constraints. For the ith line where $A[i] = 0$, the constraints are

$$\begin{gathered} a_{i,1} \geq c \cdot a_{i-1,1},\ a_{i,1} \geq c \cdot a_{i-1,2},\ a_{i,1} \geq c \cdot b_{i-1,1},\ a_{i,1} \geq c \cdot b_{i-1,2},\ b_{i,1} = b_{i-1,2} \\ (\forall\ k:\ 2 \leq k \leq m-1)\ a_{i,k} = a_{i-1,k+1},\ b_{i,k} = b_{i-1,k+1} \\ a_{i,m} = b_{i,m} = 0. \end{gathered}$$

These express the replacement of

$$\cdots\ {}^{b_2}(Q_1 n^{a_2})^{b_1}\mathsf{DTS}[n^{a_1}]$$

in the ith line with

$$\cdots\ {}^{b_2}\mathsf{DTS}[n^{c\cdot\max\{a_1,a_2,b_1,b_2\}}]$$

in the $(i+1)$st line.

This concludes the description of the linear program. The above discussion, combined with a binary search, yields the following theorem.

Theorem 3.2 *Given a proof annotation of n lines, the best possible lower bound proof following the annotation can be determined up to n digits of precision, in* $\text{poly}(n)$ *time.*

3.4 Experimental Results

Armed with the LP formulation, we wrote proof search routines in Maple 10 using the Optimization package. Thanks to built-in procedures, our code is quite short – under a few hundred lines. For large proof annotations (exceeding 100 lines), we used the freeware `lp_solve` package to solve the corresponding linear program.[7] We hope to make our code available for public use soon. Our routines include the following:

- `list2LPc`: Takes a proof annotation as input and outputs a set of constraints corresponding to the relevant LP instance. This routine performs the translation given in the previous section.
- `proofproduce`: Takes a solution to a LP instance (given as a set of assignments to variables) and prints a line-by-line human-readable proof.
- `binarysearch`: Takes a proof annotation and range (c_1, c_2) as input, and prints a human-readable proof as well as the largest $c \in (c_1, c_2)$ (within nine digits of precision) such that an n^c lower bound could be proved for the given annotation.
- `randomadmiss`: Takes an integer k and outputs a random k-bit proof annotation, drawn uniformly at random over all such annotations. (Random bits are obtained using the Blum-Blum-Shub generator [BBS86].) To perform the sampling, we adapted a simple method for producing random well-balanced strings, given by Arnold and Sleep [AS80].
- `heuristicsearch`: Takes an ordered list of proof annotations and lower bound exponent c as input and runs as follows: take the first annotation A on the list and try all possible annotations A' gotten by inserting a 1 and 0 in A. For each A', if `binarysearch`$(A', c, 2)$ succeeds and A' achieves the best exponent so far for proofs of its length, then add A' to the end of the list. (A tabu list is maintained to keep redundant comparisons at a minimum.) Examining the printed results of this heuristic method turns out to be an efficient way to find better proof annotations, starting from one that is known to be good. In fact, by calling `heuristicsearch([[1,0,0]],1.4)` one actually generates the whole table of results below.
- `writeLP`: Takes an LP instance and filename and writes the LP to the `*.lp` file format, used by `lp_solve`.

Our first objective was to verify some lower bounds that we knew to hold, such as the $n^{1.8019}$ result. In particular, we wished to ensure that our choice of parameters in the 1.8019 proof were tight. We tested a 300-line LP (consisting of a few hundred variables) corresponding to an annotation for our proof of the 1.8019 bound, and found that it was feasible for $c = 1.8017$ but infeasible for 1.8018; moreover, its choice of parameters mimicked our own.

Our second objective was to search as much of the space of proof annotations as we could, looking for interesting patterns. For all even-numbered k from 2 to 26, we conducted an exhaustive search over all valid proof annotations with k lines. The best proof annotations for each k are

[7]The `lp_solve` package is an open source simplex-based linear programming solver. It is maintained by a community on Yahoo Groups: `http://groups.yahoo.com/group/lp_solve`.

given in the below table. For $k > 26$ we have not exhaustively searched the space of all proofs, but we have searched by random uniform sampling ($> 40,000$ samples) over all proof annotations – these rows in the following table are marked with an asterisk ($*$). For those rows with multiple annotations, we checked the annotations to two more decimal places to further verify that the obtained lower bounds are the same. The Δ of a row is the difference between the lower bound exponent of that row and the exponent of the previous row.

#Lines	Best Proof Annotation(s)	L.B.	Δ
4	$[1, 0, 0]$	1.4142	0
6	$[1, 0, 1, 0, 0]$	1.5213	0.1071
	$[1, 1, 0, 0, 0]$		
8	$[1, 1, 0, 0, 1, 0, 0]$	1.6004	0.0791
10	$[1, 1, 0, 0, 1, 0, 1, 0, 0]$	1.633315	0.032915
	$[1, 1, 0, 1, 0, 0, 1, 0, 0]$		
	$[1, 1, 1, 0, 0, 0, 1, 0, 0]$		
12	$[1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0]$	1.6635	0.0302
14	$[1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0]$	1.6871	0.0236
16	$[1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0]$	1.699676	0.012576
	$[1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0]$		
	$[1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0]$		
18	$[1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0]$	1.7121	0.0125
20	$[1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0]$	1.7232	0.0111
22	$[1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$	1.7322	0.0090
24	$[1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$	1.737851	0.005651
	$[1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$		
	$[1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$		
26	$[1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$	1.7437	0.005849
28*	$[1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$	1.7491	0.0054
30*	$[1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]$	1.7537	0.0046
32*	$[1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]$	1.7577	0.0040
34*	$[1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]$	1.760632	0.002932
	$[1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]$		
	$[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]$		

Before we analyze the above table, let us first note that the proofs produced by the above annotations bear strong similarities to those in our 1.801 lower bound. To give a small example, the best 14-line proof (establishing an $\Omega(n^{1.6871})$ time lower bound) is output as:

```
0, DTS[n^5.275587925]
1, (E n^1.853485593)(A n^1.)DTS[n^3.422102331]
2, (E n^1.853485593)(A n^1.422102331)(E n^1.)DTS[n^2.000000001]
3, (E n^1.853485593)(A n^1.422102331)(E n^1.000000001)(A n^1.000000000)DTS[n^1.]
4, (E n^1.853485593)(A n^1.422102331)(E n^1.000000001)DTS[n^1.687100000]
5, (E n^1.853485593)(A n^1.422102331)DTS[n^2.846306408]
6, (E n^1.853485593)(A n^1.423153204)(E n^1.000000000)DTS[n^1.423153204]
7, (E n^1.853485593)(A n^1.423153204)DTS[n^2.401001771]
8, (E n^1.853485593)DTS[n^4.050730087]
```

```
9, (E n^1.853485593)(A n^1.000000000)DTS[n^2.197244494]
10, (E n^1.853485593)DTS[n^3.706971186]
11, (E n^1.853485593)(A n^1.000000000)DTS[n^1.853485593]
12, (E n^1.853485593)DTS[n^3.127015544]
13, DTS[n^5.275587925]
```

(Note that the LP actually returns numbers to 18 decimal places– the proof-printing routine truncates them to keep the presentation legible.)

Turning back to the results of the table, we see a strong correlation between later rows of the table and earlier ones. For example, there is a tie for best annotation at 10, 16, 24, and 34 lines, among three annotations that differ only in three of their bits. To develop a greater understanding of what is happening, let us introduce some abbreviations in the annotation. Where an annotation contains the string $(1\ 0)^k\ 0$, we put the symbol $\mathbf{k}$, for $k \geq 1$. Where an annotation contains the string 11000, we just put **0**. The following table emerges:

#Lines	Best Proof Annotation(s)	L.B.	Δ
4	**1**	1.4142	0
6	**2**	1.5213	0.1071
	0		
8	1 **2**	1.6004	0.0791
10	1 **1 2**	1.633315	0.032915
	1 **2 1**		
	1 **0 1**		
12	1 1 **1 1 1**	1.6635	0.0302
14	1 1 **1 1 2**	1.6871	0.0236
16	1 1 **1 2 2**	1.699676	0.012576
	1 1 **2 1 2**		
	1 1 **0 1 2**		
18	1 1 1 **1 1 1 2**	1.7121	0.0125
20	1 1 1 **1 1 2 2**	1.7232	0.0111
22	1 1 1 **1 1 2 3**	1.7322	0.0090
24	1 1 1 **1 2 2 3**	1.737851	0.005651
	1 1 1 **2 1 2 3**		
	1 1 1 **0 1 2 3**		
26	1 1 1 1 **1 1 1 2 3**	1.7437	0.005849
28*	1 1 1 1 **1 1 2 2 3**	1.7491	0.0054
30*	1 1 1 1 **1 1 2 3 3**	1.7537	0.0046
32*	1 1 1 1 **1 1 2 3** 4	1.7577	0.0040
34*	1 1 1 1 **1 2 2 3** 4	1.760632	0.002932
	1 1 1 1 **2 1 2 3** 4		
	1 1 1 1 **0 1 2 3** 4		

For an optimal annotation that ends with a non-zero **k**, a longer optimal annotation can be obtained by adding either a **k** or **k+1** to the end, and a 1 at the beginning. (There are of course some restrictions– there are no more than three consecutive **1**'s, no more than two consecutive **2**'s, *etc.*) Unfortunately, we do not yet know how to *prove* that all of the best proofs must have this behavior, but it would be rather extraordinary if this pattern deviated at some later point.

We solved the corresponding LP for many proof annotations, including all annotations used by previous time lower bounds, with no success beyond the $2\cos(\pi/7)$ exponent. The above table suggests that it should suffice for us to examine those proof annotations of the form $1 \cdots 1\,\mathbf{0}\;\mathbf{1}\;\mathbf{2}\;\mathbf{3}\;\mathbf{4}\cdots$; however, these annotations also do not lead to an improvement. To illustrate, for the 424 line proof annotation denoted by the sequence

$$1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;1\;\mathbf{0}\;\mathbf{1}\;\mathbf{2}\;\mathbf{3}\;\mathbf{4}\;\cdots\;\mathbf{17}\;\mathbf{18}\;\mathbf{19},$$

experiments with `lp_solve` revealed that the optimal exponent is only in the interval $[1.80175, 1.8018)$. These empirical results point strongly to a conjecture:

Conjecture 3.1 *There is no alternation-trading proof that* $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$*, for any* $c > 2\cos(\pi/7) \approx 1.8019$.

At present we do not know how to prove the conjecture, but we can give a partial result. It was already believed that alternation-trading proofs of lower bounds for SAT had limitations, in that a quadratic time lower bound appeared to be the best one could do. In our formalism it is relatively easy to prove that a quadratic lower bound is impossible to achieve:

Theorem 3.3 *There is no alternation-trading proof that* $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^2]$.

Proof. Suppose there is such a proof, and let A be an annotation for it. We assume $|A| > 3$, for in this case the only annotation is $[1, 0, 0]$ and that can only attain an $n^{\sqrt{2}}$ lower bound.[8] We claim that if the subsequence $1, 0$ occurs in the proof, then some occurrence of it can be removed and the proof can only be improved. This claim implies a contradiction, by the following reasoning. Note that *every* proof contains a sequence $1, 0$. Recall that proof annotations can be put in one-to-one correspondence with strings of balanced parentheses, so there is some occurrence of $1, 0$ in the proof that corresponds to an adjacent parentheses pair `( )` from the corresponding string. Let us remove that occurrence of $1, 0$ from the annotation, and the appropriate `( )` from the parentheses string. By repeatedly removing parentheses pairs, at some point we reach the single string `( )` corresponding to the annotation $[1, 0, 0]$, yielding a contradiction.

It remains to prove the claim. If $1, 0$ occurs, then the two lines in the proof corresponding to the sequence (including the previous line) look like:

$$\cdots^{b_{k-1}}(Q_{k-1}\ n^{a_{k-1}})^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}] \tag{3}$$

$$\cdots^{b_{k-1}}(Q_{k-1}\ n^{a_{k-1}})^{b_k}(Q_k\ n^{\max\{x,a_k\}})^{\max\{x,b_{k+1}\}}(Q_{k+1}\ \log n)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}] \tag{4}$$

$$\cdots^{b_{k-1}}(Q_{k-1}\ n^{a_{k-1}})^{b_k}(Q_k\ n^{\max\{x,a_k\}})^{\max\{x,b_{k+1}\}}\mathsf{DTS}[n^{\max\{c(a_{k+1}-x),cx,cb_{k+1}\}}] \tag{5}$$

It is clear that every parameter in the class (5) is at least the corresponding parameter in the class (3), except for possibly the runtime of the DTS computation. Therefore if $a_{k+1} \le c(a_{k+1} - x)$,

[8] While the computer verified this, it is also easy to prove on paper as well.

or $a_{k+1} \leq cx$, or $a_{k+1} \leq cb_{k+1}$, then the sequence $1, 0$ could be removed with no harm to the proof. So suppose for contradiction that $a_{k+1} > c(a_{k+1} - x)$ and $a_{k+1} > cx$. Then $2a_{k+1} > c(a_{k+1} - x) + cx$, a contradiction when $c \geq 2$. □

One could imagine a proof of Conjecture 3.1 along similar lines as the above, by ruling out so many particular subsequences in the proof annotation that the annotation is forced to take a form that we know is suboptimal. If the conjecture is true, it is very useful knowledge in that it forces us to rethink our whole approach. In order to significantly exceed the current best result, it appears that we cannot expect to rearrange the existing ingredients– we must find ways to mix in additional complexity-theoretic components. Perhaps we could use randomness as well as alternation to perform interesting "speed ups" of DTS, where the addition of randomness allows us to obtain new types of proof-by-contradiction. Perhaps we can find a better method for "slowing down", removing alternations more efficiently than the simple Slowdown Lemma. There are plenty of tools out there, and we believe the search is not over but instead is only beginning.

3.5 Some Extensions

Explicit Time-Space Tradeoffs. A tweak to the speedup rules in the LP formulation lets us prove some new time-space tradeoff lower bounds for solving SAT. In particular, we just have to translate the more general version of the Speedup Lemma (Lemma 2.1) instead of the special DTS version. This introduces a new parameter e (the exponent of the space bound) which we must set in advance to keep the constraint program linear.

The table below gives time-space function pairs for which our theorem-prover has shown that no SAT algorithm can satisfy both time and space requirements simultaneously. Experimentation suggested that the optimal annotations for the $n^{o(1)}$ space setting were still the best possible, independently of the space bound. In the results below, the proof annotation used is a sufficiently large one from the $2\cos(\pi/7)$ lower bound.

Time	Space
$n^{1.06}$	$n^{.9}$
$n^{1.17}$	$n^{.75}$
$n^{1.24}$	$n^{.666}$
$n^{1.36}$	$n^{.5}$
$n^{1.51}$	$n^{.333}$
$n^{1.58}$	$n^{.25}$
$n^{1.7}$	$n^{.1}$
$n^{1.75}$	$n^{.05}$

Extending to Quantified Boolean Formulas. The approach for SAT lower bounds can easily be extended to proving time lower bounds for k-QBF, the problem of solving quantified Boolean formulas with k quantifiers. Let us just sketch how the formalism changes. The rules behind these lower bounds are the same as with the SAT case, except that in the Slowdown Rules of the linear program, we now allow for k quantifier blocks to be removed in one application of the rule (instead of just one for SAT). The resulting computer program obtained very similar results to the lower bounds for the SAT setting, and did not find a better lower bound than the n^{c_k} time and $n^{o(1)}$ space bound of our earlier work [Wil07b], where c_k is the positive root in $(1, 2)$ of the polynomial $p(x) = x^3/k - x^2 - 2x + k$.

4 Nondeterministic Time-Space Lower Bounds for Solving Tautologies

Along with proving deterministic time-space lower bounds for NP problems, the problem of proving nondeterministic time-space lower bounds for $\mathsf{co\text{-}NP}$ has also been studied. However, the latter problem has seen less progress. Fortnow and Van Melkebeek [FvM00] first studied the problem and proved that TAUTOLOGY requires $\Omega(n^{\sqrt{2}-o(1)})$ time on a nondeterministic $n^{o(1)}$-space RAM, but since their result no further progress had been made on this problem. We shall sketch how to extend the LP-based approach of the previous section to this lower bound problem, and find that the best proof annotations actually look quite different from the previous setting. Here the approach turns out to be quite successful; in fact, the seven-year-old $n^{\sqrt{2}}$ lower bound can already be improved with a very short eleven-line proof. By analyzing the results further on paper, we uncover a new inductive argument that leads to a $\Omega(n^{\sqrt[3]{4}-o(1)})$ time lower bound for TAUTOLOGY.

4.1 The Framework

Similar to the class DTS, we define $\mathsf{NTS}[n^a] := \mathsf{NTISP}[n^a, n^{o(1)}]$ and $\mathsf{coNTS}[n^a] := \mathsf{coNTISP}[n^a, n^{o(1)}]$ for notational brevity. As in the previous setup, there are speedup and slowdown rules that are applied in a manner that contradicts a time hierarchy theorem. The Speedup Lemma is essentially the same as before, but the Slowdown Lemma is a little different. In the following, let Q represent any string of quantifier blocks of the form

$$Q = (Q_1\ n^{a_1})^{b_2} \cdots (Q_{k-1}\ n^{a_{k-1}}).$$

Lemma 4.1 *(Speedup) For $b \geq 1$, $a \geq 1$, $x \geq 0$, and $s \geq 0$,*

$$Q^b\mathsf{NTISP}[n^a, n^s] \subseteq Q^b(\exists\ n^{x+s})^{\max\{b,x+s\}}(\forall\ \log n)^{\max\{b,s\}}\mathsf{NTISP}[n^{a-x}, n^s].$$

In particular for $s = o(1)$ we have $\mathsf{NTS}[n^a] \subseteq (\exists\ n^x)^{\max\{1,x\}}(\forall\ \log n)^1\mathsf{NTS}[n^{a-x}]$.

Proof. The proof is analogous to Lemma 2.1 (the Speedup Lemma for DTISP). $\square$

Lemma 4.2 *(Slowdown) If TAUTOLOGY is in $\mathsf{NTS}[n^c]$ then*

1. $Q^b(\exists\ n^{a_k})^{b_{k+1}}\mathsf{NTS}[n^{a_{k+1}}] \subseteq Q^b\mathsf{coNTS}[n^{c\cdot\max\{a_k, a_{k+1}, b_{k+1}, b\}}$,
2. $Q^b(\forall\ n^{a_k})^{b_{k+1}}\mathsf{coNTS}[n^{a_{k+1}}] \subseteq Q^b\mathsf{NTS}[n^{c\cdot\max\{a_k, a_{k+1}, b_{k+1}, b\}}]$,
3. $Q^b(\exists\ n^{a_k})^{b_{k+1}}\mathsf{coNTS}[n^{a_{k+1}}] \subseteq Q^b(\exists\ n^{a_k})^{b_{k+1}}\mathsf{NTS}[n^{c\cdot\max\{a_{k+1}, b_{k+1}\}}]$, *and*
4. $Q^b(\forall\ n^{a_k})^{b_{k+1}}\mathsf{NTS}[n^{a_{k+1}}] \subseteq Q^b(\forall\ n^{a_k})^{b_{k+1}}\mathsf{coNTS}[n^{c\cdot\max\{a_{k+1}, b_{k+1}\}}]$.

Proof. To prove (2), observe that

$$(\forall\ n^{a_k})^{b_{k+1}}\mathsf{coNTS}[n^{a_{k+1}}] \subseteq \mathsf{coNTIME}[n^{\max\{a_k,a_{k+1},b_{k+1}\}}],$$

and the assumption along with Corollary 2.1 implies that this class is contained in $\mathsf{NTS}[n^{c\cdot\max\{a_k,a_{k+1},b_{k+1}\}}]$. The proof of (1) is analogous. To prove (3), use the fact that

$$(\exists\ n^{a_k})^{b_{k+1}}\mathsf{coNTS}[n^{a_{k+1}}] \subseteq (\exists\ n^{a_k})^{b_{k+1}}\mathsf{coNTIME}[n^{a_{k+1}}],$$

so the assumption implies the above is in $(\exists\ n^{a_k})^{b_{k+1}}\mathsf{NTS}[n^{c\cdot\max\{a_{k+1},b_{k+1}\}}]$. The proof of (4) is analogous. □

To yield contradictions, one can use the alternating time hierarchy (Theorem 2.2) just as in the deterministic case.

4.2 Translation Into Linear Programming

The above rules immediately lead to a natural definition of *alternation-trading proof that* $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c] \implies A_1 \subseteq A_2$, for classes A_1 and A_2. Such a definition is completely analogous to the one given for satisfiability time lower bounds.

Another way to yield a contradiction is to use a result that has nearly identical structure to Lemma 3.3, which showed that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $\mathsf{DTS}[n^a] \not\subseteq \mathsf{DTS}[n^{a'}]$ for $a > a'$.

Lemma 4.3 *If* $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c]$ *then* $\mathsf{NTS}[n^a] \not\subseteq \mathsf{coNTS}[n^{a'}]$ *for* $a > a'$.

Just as in the SAT lower bounds setting, we may use this lemma to motivate a definition of *normal form proof*, and prove that any alternation-trading proof can be converted into a normal form type.

Definition 4.1 *Let* $c \geq 1$. *An alternation-trading proof of* $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c] \implies A_1 \subseteq A_2$ *is in* normal form proof for c *if*

- $A_1 = \mathsf{NTS}[n^a]$ *and* $A_2 = \mathsf{coNTS}[n^{a'}]$, *for some* $a \geq a'$.
- *No other lines are* NTS *or* coNTS *classes.*

Lemma 4.4 *Let* $c \geq 1$. *If there is an alternation-trading proof that* $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c] \implies A_1 \subseteq A_2$ *in normal form that has at least two lines, then* $\mathsf{coNTIME}[n] \not\subseteq \mathsf{NTS}[n^c]$.

Theorem 4.1 *Let* A_1 *and* A_2 *be complementary. If there is an alternation-trading proof* P *that* $(\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies A_1 \subseteq A_2)$, *then there is a normal form proof for* c, *of length at most that of* P.

Example. If TAUTOLOGY is in $\mathsf{NTS}[n^c]$ then $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c]$ by Theorem 2.1. Thus

$$\begin{aligned}
\mathsf{NTS}[n^2] &\subseteq (\exists\ n)^1(\forall\ \log n)^1\mathsf{NTS}[n] \quad \text{NTISP Speedup (Lemma 4.1)}\\
&\subseteq (\exists\ n)^1(\forall\ \log n)^1\mathsf{coNTS}[n^c] \quad \text{NTISP Slowdown (Lemma 4.2)}\\
&\subseteq (\exists\ n)^1\mathsf{NTS}[n^{c^2}] \quad \text{(Slowdown)}\\
&\subseteq \mathsf{coNTS}[n^{c^3}]. \quad \text{(Slowdown)}
\end{aligned}$$

When $c \leq \sqrt[3]{2} \approx 1.25$ we have $\mathsf{NTS}[n^a] \subseteq \mathsf{coNTS}[n^{a'}]$ for some $a > a'$, which contradicts Lemma 4.3.

As earlier, we can define proof annotations for this setting, where

- the ith bit of a proof annotation is 1 if the ith line of the proof is a speedup lemma application, and
- the ith bit of a proof annotation is 0 if the ith line of the proof is a slowdown lemma application.

The bit vectors that correspond to valid proof annotations change a bit, due to differences in the Speedup and Slowdown Lemmas. Here is a procedure for checking if a given bit string is a proof annotation:

```
If $A[1] \neq 1$ then return *invalid*. Set $n \leftarrow 2$, $q \leftarrow 0$.
For all $i = 2, \dots, (\ell - 1)$,
    If $A[i] = 1$ and $\neg q$ then $n \leftarrow n + 2$.
    If $A[i] = 1$ and $q$ then $n \leftarrow n + 1$ and $q \leftarrow 0$.
    If $A[i] = 0$ and $\neg q$ then $q \leftarrow 1$.
    If $A[i] = 0$ and $q$ then $n \leftarrow n - 1$.
    If $n < 1$ then return *invalid*.
End for
If $n = 0$ then *valid* else *invalid*.
```

The parameter n corresponds to the number of quantifier blocks in the current line, and $q = 1$ iff the last quantifier of the current line has the same type as the final time-space class. That is, $q = 1$ precisely when the class is either of the form $\cdots(\exists)\mathsf{NTS}$ or $\cdots(\forall)\mathsf{coNTS}$. For examples, we note that $[1, 0, 0, 0]$, $[1, 1, 0, 0, 0, 0, 0]$, and $[1, 0, 1, 0, 0, 0, 0]$ are proof annotations for this setting, the first being the annotation for our small example above.

The translation of a proof annotation to linear programming for the "TAUTOLOGY versus NTS" problem follows exactly the same strategy as that for the "SAT versus DTS" problem: we define variables $a_{i,j}$, $b_{i,j}$, x_i for all lines i and possible quantifier blocks j, and replace components of the form $\max\{a, a'\} = a''$ with $a'' \geq a$, $a'' \geq a'$. It is clear from the Speedup and Slowdown Lemmas given above that this reduction suffices. In implementation, the reduction is more complicated due to the four possible cases for applying the Slowdown Lemma, but the overall idea is precisely the same.

4.3 Experimental Results

While the proofs of these new Speedup and Slowdown Lemmas are quite similar to those used for the "SAT versus DTISP" problem, the structure of good lower bound proofs for the "TAUTOLOGY versus NTISP" problem look rather different. We wrote similar routines to those in the deterministic setting, and the resulting program uncovered very interesting results. It turns out that Fortnow and Van Melkebeek's $n^{\sqrt{2}}$ lower bound is not optimal; in fact, the optimal eleven line proof already establishes a 1.419 exponent:

```
0, NTS[n^4.788956584]
1, (E n^2.369956583)(A n^1.)NTS[n^2.419]
2, (E n^2.369956583)(A n^1.)(E n^1.419)(A n^1.)NTS[n^1.]
3, (E n^2.369956583)(A n^1.)(E n^1.419)(A n^1.)coNTS[n^1.419]
4, (E n^2.369956583)(A n^1.)(E n^1.419)NTS[n^2.013561000]
5, (E n^2.369956583)(A n^1.)coNTS[n^2.857243059]
6, (E n^2.369956583)(A n^2.378351877)(E n^1.)coNTS[n^1.181167036]
7, (E n^2.369956583)(A n^2.378351877)(E n^1.)NTS[n^1.676076023]
8, (E n^2.369956583)(A n^2.378351877)coNTS[n^2.378351877]
9, (E n^2.369956583)NTS[n^3.374881314]
10, coNTS[n^4.788956584]
```

Below is a table of results found by exhaustive search over valid annotations.

#Lines	Best Proof Annotation(s)	L.B.
5	[1, 0, 0, 0]	1.323
8	[1, 1, 0, 0, 0, 0, 0]	1.380
	[1, 0, 1, 0, 0, 0, 0]	
11	[1, 1, 0, 0, 0, 1, 0, 0, 0, 0]	1.419
14	[1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]	1.433
	[1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0]	
	[1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]	
	[1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0]	
17	[1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0]	1.445
	[1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]	
20	[1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]	1.455
	[1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0]	
	[1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]	
	[1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]	
23	[1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]	1.465

Note how larger annotations are composed of smaller ones: for example, $[1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$ is $[1, A_7, A_4, 0]$, where A_4 and A_7 are optimal annotations for four and seven lines. In particular, observe that three optimal annotations from the table have a very distinctive pattern, namely

$$[1, 0, 0, 0], [1, 1, 0, 0, 0, 1, 0, 0, 0, 0], \text{ and } [1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0].$$

The pattern is: given a sequence, the next sequence is obtained by concatenating two copies of the current sequence, and placing a 1 at the beginning and 0 at the end. Thus the next annotation in the pattern would be

$$[1,1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,\\ 1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0],$$

a 47-line annotation that gives a 1.49 exponent. To determine if this annotation is indeed good, we implemented a heuristic search which starts by putting a given annotation on a queue, and noting the exponent obtained with that annotation. Then the search repeats in a loop where the head of the queue is removed, and all possible ways to insert at most 3 bits into that annotation are tried. If one of these new annotations is valid, obtains a better exponent than the original, and is at least as good as the best of that length found so far, the new annotation is added to the end of the queue.

For many line numbers ℓ, the heuristic search found a large number of proof annotations that achieve exactly the same lower bound. For example, there are eight such annotations of 26 lines. Each optimal annotation we found could be written as a concatenation of smaller optimal annotations along with an additional 1 and 0. After a long search, the heuristic search found the above 47-line annotation, and all other annotations found (with at most 47 lines) gave as good of a lower bound.

4.4 New Time Lower Bound for Solving Tautologies With Nondeterminism

After examining the optimal proof annotations closely, it was evident that the particular annotations

$$\begin{aligned} A_1 &= [1,0,0,0] \\ A_2 &= [1,1,0,0,0,1,0,0,0,0] \\ A_3 &= [1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0] \\ A_4 &= [1,1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0, \\ &\quad\ 1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0] \end{aligned}$$

enjoy a special relationship, namely $A_{i+1} = [1, A_i, A_i, 0]$ (with a slight abuse of notation). This observation strongly suggests that we should look for a new lower bound proof by induction, where the induction hypothesis is applied twice in the inductive step. Doing so, we arrived at the following result.

Theorem 4.2 *The* TAUTOLOGY *problem requires $n^{\sqrt[3]{4}-o(1)}$ time for nondeterministic algorithms that use $n^{o(1)}$ space.*

To prove Theorem 4.2, we need to establish an inductive lemma. For a given constant $c \geq 1$, define the sequence $x_1 := 1$, $x_2 := c$, $x_k := c^3(x_{k-1})^2/(\sum_{i=1}^{k-1} x_i)$.

Lemma 4.5 *If $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c]$ and $c^2(x_k)^2 \geq \sum_{i=1}^{k} x_i$, then for all $k \geq 2$,*

$$\mathsf{NTS}[n^{\sum_{i=1}^{k} x_i}] \subseteq (\exists\ n^{x_k})(\forall\ n^{x_k})\mathsf{coNTS}[n^{x_k}].$$

In the following, we do not bother with specifying the inputs to quantifier blocks, except where absolutely necessary for the argument.

Proof. For $k = 2$ we have $\mathsf{NTS}[n^{c+1}] \subseteq (\exists\ n^c)(\forall\ \log n)\mathsf{NTS}[n] \subseteq (\exists\ n^c)(\forall\ \log n)\mathsf{coNTS}[n^c]$. Note that $c^2(x_k)^2 \geq \sum_{i=1}^{k} x_i$ implies $x_{k+1} \geq 1$. By induction we have that

$$\begin{aligned}
\mathsf{NTS}[n^{\sum_{i=1}^{k+1} x_i}] &\subseteq (\exists\ n^{x_{k+1}})(\forall\ \log n)^1\ \mathsf{NTS}[n^{\sum_{i=1}^{k} x_i}] \quad \text{(Speedup)}\\
&\subseteq (\exists\ n^{x_{k+1}})(\forall\ \log n)^1(\exists\ n^{x_k})(\forall\ n^{x_k})\mathsf{coNTS}[n^{x_k}] \quad \text{(by Induction Hypothesis)}\\
&\subseteq (\exists\ n^{x_{k+1}})(\forall\ \log n)^1(\exists\ n^{x_k})\mathsf{NTS}[n^{cx_k}] \quad \text{(Slowdown)}\\
&\subseteq (\exists\ n^{x_{k+1}})(\forall\ \log n)^1\mathsf{coNTS}[n^{c^2x_k}] \quad \text{(Slowdown)}\\
&\subseteq (\exists\ n^{x_{k+1}})(\forall\ \log n)^1(\forall\ n^{c^2(x_k)^2/(\sum_i x_i)})(\exists\ n^{c^2(x_k)^2/(\sum_i x_i)})\mathsf{NTS}[n^{c^2(x_k)^2/(\sum_i x_i)}]\\
&\qquad\qquad \text{(Induction Hypothesis, and Assumption)}\\
&\subseteq (\exists\ n^{x_{k+1}})(\forall\ \log n)^1(\forall\ n^{c^2(x_k)^2/(\sum_i x_i)})\mathsf{coNTS}[n^{c^3(x_k)^2/(\sum_i x_i)}] \quad \text{(Slowdown)}\\
&\subseteq (\exists\ n^{x_{k+1}})(\forall\ n^{x_{k+1}})\mathsf{coNTS}[n^{x_{k+1}}].
\end{aligned}$$

□

Note that the lemma indeed applies its induction hypothesis twice. The proof of the lower bound on tautologies can now be derived.

Proof of Theorem 4.2. Assume $\mathsf{coNTIME}[n] \subseteq \mathsf{NTS}[n^c]$. Suppose ℓ is the smallest integer satisfying $c^2(x_\ell)^2 < \sum_{i=1}^{\ell} x_i$. Note that $c^2(x_2)^2 = c^4 \geq c + 1 = x_1 + x_2$ for $c > 1.2207$, which we know holds due to Fortnow and Van Melkebeek. Therefore $\ell \geq 3$. By Lemma 4.5 and the Slowdown Lemma, for every $k < \ell$ we have

$$\mathsf{NTS}[n^{\sum_{i=1}^{k} x_i}] \subseteq (\exists\ n^{x_k})(\forall\ n^{x_k})\mathsf{coNTS}[n^{x_k}] \subseteq \mathsf{coNTS}[n^{c^2x_k}]. \tag{6}$$

Define the sequence $s_k := (\sum_{i=1}^{k} x_i)/x_k = 1 + \sum_{i=1}^{k-1} x_i/x_k$. By induction it is easily shown that $s_k = 1 + (s_{k-1})^2/c^3$, and that this sequence is increasing. The inclusion (6) says that we can obtain a contradiction with Lemma 4.3 when $c^2 \leq s_k$. Hence if $c^2 \leq s_{\ell-1}$, we have a contradiction with Lemma 4.3 (the NTS versus coNTS hierarchy). However, we know that $c^2x_\ell < \sum_{i=1}^{\ell} x_i/x_\ell$ and $c^2x_{\ell-1} \geq \sum_{i=1}^{\ell-1} x_i/x_{\ell-1}$, by our choice of ℓ. Taken together, the two inequalities are equivalent to the condition $x_{\ell+1} < c \leq x_\ell$. By simple algebra and the fact that $c^3 > c + 1$ (which holds for $c > 1.33$), one can show that this condition implies $c^2 \leq s_{\ell-1}$. Hence no such ℓ exists.

So suppose instead that $c^2(x_k)^2 \geq \sum_{i=1}^{k} x_i$, for all k. Then inclusion (6) holds for all k. For which c can we obtain $c^2 \leq s_k$, for sufficiently large k? Either the sequence $\{s_k\}$ is unbounded (in which case we're done, as the inequality holds for all c) or it has a limit point. In the latter case, we have $s_\infty = 1 + s_\infty^2/c^3$. The polynomial $p(x) = 1 + x^2/c^3 - x$ has roots $x = c \cdot (c^2/2 \pm \sqrt{(c^4 - 4c)/2})$. When $c = 4^{1/3}$, this root is imaginary, therefore s_∞ would be imaginary, a contradiction. It follows that $c < 4^{1/3}$. □

It can be verified that the above theorem's proof annotations are precisely A_1, A_2, A_3, *etc.*; in fact, the sequence of exponents x_i are precisely those chosen by the computer program for the annotations A_1, A_2, A_3, *etc.* Therefore the above formal proof completely corresponds to our best experimental results. For further generalizations of this new lower bound, the reader is invited to peruse our joint tech report with Diehl and Van Melkebeek [DvMW07]. Given that our experimental results indicated that the sequence A_1, A_2, A_3, *etc.* is among the best one can do, we posit that our above lower bound is optimal.

Conjecture 4.1 *There is no alternation trading proof that* $\mathsf{coNTIME}[n] \not\subseteq \mathsf{NTS}[n^c]$*, for any* $c > \sqrt[3]{4}$.

5 Lower Bounds for Hybrid One-Tape Turing Machines

Finally, we consider lower bounds on a "hybrid" Turing machine model, which subsumes both the small-space random access machine and off-line one-tape Turing machine models. As in the previous section, the automated approach yields new lower bounds here too, and we are able to extrapolate an inductive lower bound proof from the experiments. Recall that the hybrid one-tape TM 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.

The hybrid one-tape model holds importance in this area, since it is the most powerful deterministic model we know for which one can still prove an $n^{1+\Omega(1)}$ lower bound for solving satisfiability.

5.1 The Framework

We define $\mathsf{DTIME}_h[t(n)]$ to be the class of languages recognized by hybrid Turing machines running in $O(t(n))$ time. The lower bound proofs for hybrids have a somewhat different structure from the earlier ones. These proofs exploit the fact that a hybrid one-tape Turing machine can be weakly simulated by a nondeterministic random access machine having a modest space bound. More precisely, let Q represent any string of quantifier blocks of the form $Q = (Q_1\ n^{a_1})^{b_2} \cdots (Q_{k-1}\ n^{a_{k-1}})$.

Lemma 5.1 (DTIME_h to DTISP) *Let* $Q_{k+1} \in \{\exists, \forall\}$*. Then for all* $0 < s \leq a$ *and* $b \geq 1$, $Q^b\mathsf{DTIME}_h[n^a] \subseteq Q^b(Q_{k+1}n^{a-s})^{\max\{a-s,b\}}\mathsf{DTISP}[n^a, n^s]$.

The proof of Lemma 5.1 is a standard crossing sequence argument. We include a sketch of it for completeness.

Proof. (Sketch) Let M be a hybrid one-tape machine and let x be an input. Define $CS(x)$ to be the set of crossing sequences for $M(x)$ on its sequential tape. We associate each sequence with a cell of the tape. Notice that each element of a crossing sequence for a particular cell is of $n^{o(1)}$ size.

For every $i = 1, \ldots, n^s$, define $CS(x, i)$ to be the subset of sequences from $CS(x)$ ranging over the cells numbered $i + k \cdot n^s$, for $k = 0, 1, \ldots, n^{a-s} - 1$. Observe that the union of all $CS(x, i)$ is exactly $CS(x)$, and that $CS(x, i) \cap CS(x, j) = \varnothing$ for $i \neq j$. It follows that since the total sum of the lengths of all crossing sequences for $M(x)$ is at most n^a, there exists a j so that the total length of all crossing sequences in $CS(x, j)$ is at most n^{a-s}. If we guess $CS(x, j)$ upfront, then our computation of $M(x)$ reduces to a $n^{a+o(1)}$ time and $n^{s+o(1)}$ space computation, by checking that the computation between the crossing sequences of $CS(x, j)$ is a valid and accepting computation. Note that the guess can be existential or universal: if it is universal then we work with crossing sequences of the complement machine $\overline{M}$, and verify that $\overline{M}(x)$ does not accept, no matter which offset and n^{a-s} subset of crossing sequences that we try. $\square$

Using the DTIME_h to DTISP Lemma, one can incorporate the Speedup and Slowdown Lemmas for space-bounded machines (Lemmas 2.1 and 2.2) to infer complexity class inclusions. Another Slowdown Lemma is also required, but its proof is straightforward and follows the lines of earlier results. Again, let Q be a string of quantifier blocks.

Lemma 5.2 (Slowdown for DTIME_h) *Suppose* $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_h[n^c]$. *Then for all appropriate* $a_i, b_i \geq 0$,

$$Q^{b_k}(Q_{k+1}\ n^{a_k})^{b_{k+1}}\mathsf{DTIME}_h[n^{a_{k+1}}] \subseteq Q^{b_k}\mathsf{DTIME}_h[n^{c\cdot\max\{b_k,b_{k+1},a_k,a_{k+1}\}}]$$

and for all $e \leq a_{k+1}$,

$$Q^{b_k}(Q_{k+1}\ n^{a_k})^{b_{k+1}}\mathsf{DTISP}[n^{a_{k+1}}, n^e] \subseteq Q^{b_k}\mathsf{DTIME}_h[n^{c\cdot\max\{b_k,b_{k+1},a_k,a_{k+1}\}}].$$

Finally, we use a simple time hierarchy for hybrid machines.

Lemma 5.3 *If* $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_h[n^c]$ *for some* $c \geq 1$, *then* $\mathsf{DTIME}_h[n^a] \subseteq \mathsf{DTIME}_h[n^{a'}]$ *for* $a > a'$.

Proof. Analogous to Lemma 3.3, which showed that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $\mathsf{DTS}[n^a] \not\subseteq \mathsf{DTS}[n^{a'}]$. $\square$

Example. In 1983, Kannan proved that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTIME}_h[n^{\sqrt[4]{3/2}}]$. His argument used a weaker version of Lemma 5.1. By reproducing the steps of his argument, we obtain:

$$\begin{aligned}
\mathsf{DTIME}_h[n^{3/2}] &\subseteq (\exists\ n)\mathsf{DTISP}[n^{3/2}, n^{1/2}] \quad \text{by } \mathsf{DTIME}_h \text{ to } \mathsf{DTISP} \text{ (Lemma 5.1)}\\
&\subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{1/2}] \quad \text{by } \mathsf{DTISP} \text{ Speedup (Lemma 2.1)}\\
&\subseteq (\exists\ n)\mathsf{DTIME}_h[n^c] \quad \text{by } \mathsf{DTISP} \text{ Slowdown (Lemma 2.1)}\\
&\subseteq \mathsf{DTIME}_h[n^{c^2}] \quad \text{by Slowdown for } \mathsf{DTIME}_h
\end{aligned}$$

A contradiction is reached for $c < \sqrt{3/2}$. But SAT $\in \mathsf{DTIME}_h[n^c]$ implies $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_h[n^{c+o(1)}]$ (Theorem 2.1), so SAT does not have an $n^{\sqrt{3/2}-\varepsilon}$ algorithm on a hybrid TM. This is the lower bound attained by Van Melkebeek-Raz [vMR05] and Maass-Schorr [MS87] who used Lemma 5.1.

5.2 Translation Into Linear Programming

As before, using the above tools as a direct guide, we can define a corresponding notion of alternation-trading proofs and proof annotations. Note that the alternating classes under consideration can now have a $\mathsf{DTISP}[t, s]$ or $\mathsf{DTIME}_h[t]$ class in the deterministic phase, *i.e.* classes of the form $(Q_1\ n^{a_1})^{b_2} \cdots (Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTIME}_1[n^{a_{k+1}}]$ or $(Q_1\ n^{a_1})^{b_2} \cdots (Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTISP}[n^{a_{k+1}}, n^{s_{k+1}}]$ are both possible. Also notice that at every step there are two possible speedup lemmas for a class with a DTISP phase: one that introduces only a single quantifier, and one that introduces two quantifiers. However, just as in the deterministic space-bounded case, we can show (along the lines of Lemma 3.2) that the second speedup lemma is unnecessary when the space bound is $O(n)$. Due to this, we can define proof annotations for this setting just as before, with bits for the two rules we can apply at each step: if the deterministic class is a DTIME_h class, a "speedup" means that we apply the DTIME_h to DTISP Lemma; if the deterministic class is DTISP, a "speedup" means that we apply the Speedup Lemma. Thus, our short example above will have the annotation $[1, 1, 0, 0]$. The procedure for determining if an annotation is valid is a little more complex, since speedups have different effects depending on whether the deterministic class is DTISP or DTIME_h.

Similar to the previous cases, we can also define a version of normal form proof, which begins with a $\mathsf{DTIME}_h[n^a]$ class and ends with a $\mathsf{DTIME}_h[n^{a'}]$ where $a' \leq a$ and the proof has at least two lines. The fact that normal form proofs imply lower bounds can be obtained from Lemma 5.3. One can prove that for every alternation-trading proof of a lower bound there is a corresponding normal form proof by using an argument exactly along the lines of Theorem 4.1.

The translation of a proof annotation to a linear program is analogous to the previous cases, except that when translating the DTIME_h to DTISP simulation (Lemma 5.1), we must introduce a new variable s_i for the space exponent, and such a variable must be present in each line i that contains a DTISP class.

5.3 Experimental Results

A summary of lower bounds found by the LP-based theorem prover is given in the below table. Unlike the previous two cases, the optimal bounds attained by optimal proofs has non-monotonic behavior (with respect to length) at first.

#Lines	Best Proof Annotation(s)	L.B.
5	[1, 1, 0, 0]	1.224
6	[1, 1, 0, 1, 0]	1.224
7	[1, 1, 1, 0, 0, 0]	1.201
8,9	[1, 1, 0, 1, 1, 0, 0], [1, 1, 0, 1, 1, 0, 1, 0]	1.262
10	[1, 1, 1, 0, 0, 1, 1, 0, 0]	1.261
11, 12	[1, 1, 0, 1, 1, 0, 1, 1, 0, 0], [1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0]	1.274
13	[1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0]	1.277
14, 15	[1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0],[1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0]	1.278
16, 17	[1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0], [1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0]	1.287
19	[1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0]	1.292
25	[1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0]	1.297
28	[1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0]	1.298

After 15 lines, the optimal annotations look like

$$A = [1\ 1\ 1\ (0\ 1\ 1)^k\ 0\ (0\ 1\ 1)^\ell\ 0\ 0],$$

for integers k, ℓ. In other words, the string '00' occurs exactly twice. Might it be that for larger proof lengths we begin to see optimal annotations with three occurrences of '0 0'? Apparently this is not the case. As with the other two problems, we implemented a heuristic search routine that takes a queue of annotations as input. The search repeatedly takes the annotation at the head of the queue and tries all possible ways to insert up to four bits in the annotation (*i.e.* for all $3^4 = 81$ ways to choose a subset of four Boolean variables and set them, the search tries all possible ways to insert those bits into the current annotation). If an annotation is obtained that is at least as good as the best one found of that length so far, it is added to the end of the queue. Even when we chose the initial queue to consist of $[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]$ (an annotation that is far in Hamming distance from the allegedly optimal annotations) this search uncovered more interesting annotations, but all of the best found had the form of A above. For example, the best 24 line proof was:

```
0, DTIMEh[n^1.751958454]
1, (E n^.9999975822)DTISP[n^1.751958454,n^.7519608720]
2, (E n^1.040108911)(A n^1.)DTISP[n^1.463810415,n^.7519608720]
3, (E n^1.040108911)(A n^1.)(E n^1.)DTISP[n^1.215771287,n^.7519608720]
4, (E n^1.040108911)(A n^1.)DTIMEh[n^1.577881470]
5, (E n^1.040108911)(A n^1.)DTISP[n^1.577881470,n^.5778814720]
6, (E n^1.040108911)(A n^1.)(E n^1.)DTISP[n^1.155762944,n^.5778814720]
7, (E n^1.040108911)(A n^1.)DTIMEh[n^1.5]
8, (E n^1.040108911)(A n^1.)DTISP[n^1.5,n^.5]
9, (E n^1.040108911)(A n^1.)(E n^1.)DTISP[n^1.,n^.5]
10, (E n^1.040108911)(A n^1.)DTIMEh[n^1.297844000]
11, (E n^1.040108911)DTIMEh[n^1.684399048]
12, (E n^1.040108909)DTISP[n^1.684399048,n^.6442901394]
13, (E n^1.040108909)(A n^1.)DTISP[n^1.288580278,n^.6442901394]
14, (E n^1.040108909)DTIMEh[n^1.672376183]
15, (E n^1.040108909)DTISP[n^1.672376183,n^.6322672739]
16, (E n^1.040108909)(A n^1.)DTISP[n^1.264534548,n^.6322672739]
17, (E n^1.040108909)DTIMEh[n^1.641168576]
18, (E n^1.040108909)DTISP[n^1.641168576,n^.6010596669]
19, (E n^1.040108911)(A n^1.)DTISP[n^1.202119332,n^.6010596669]
20, (E n^1.040108911)DTIMEh[n^1.560163362]
21, (E n^1.040108911)DTISP[n^1.560163362,n^.5200544533]
22, (E n^1.040108908)(A n^1.)DTISP[n^1.040108908,n^.5200544533]
23, (E n^1.040108908)DTIMEh[n^1.349899105]
24, DTIMEh[n^1.751958454]
```

The best found by running our heuristic search routine for a couple of days was the 66 line annotation

$$[1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,\\ 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0],$$

which gives a 1.300925 exponent. It seems clear that the best annotations indeed have the form $[1\ 1\ 1\ (0\ 1\ 1)^k\ 0\ (0\ 1\ 1)^\ell\ 0\ 0]$. In the next section we show that the convergence is rather rapid in this setting, as the above 66-line proof agrees with the case where k and ℓ are arbitrarily large in four decimal places.

5.4 New Time Lower Bound for Hybrid Machines

The annotation form $[1\ 1\ 1\ (0\ 1\ 1)^k\ 0\ (0\ 1\ 1)^\ell\ 0\ 0]$ suggests a proof where we establish an inductive lemma capturing the $(0\ 1\ 1)^*$ behavior, and apply this lemma twice to a proof of six more lines. This suggestion leads readily to a new lower bound.

Theorem 5.1 $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTIME}_h[n^c]$, *for all* $c < r$ *where* r *is the unique root in* $(1,2)$ *of the polynomial* $p(x) = 12x^5 - 12x^4 - x^3 - 13x + 4x^2 + 2$.

By numerical calculation we determine that $r < 1.30094$. The following is immediate.

Corollary 5.1 *Satisfiability requires* $n^{1.3009}$ *time to solve on a hybrid Turing machine.*

Before we prove Theorem 5.1, we first give the (promised) inductive lemma, the structure of which resembles a similar result proved for the space-bounded setting in our earlier work [Wil06]. Let $c \geq 1$, and define the sequence $e_1 := 2$, $e_{k+1} := 1 + e_k/(2c)$.

Lemma 5.4 *Suppose* $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_h[n^c]$. *Then for all* $k \geq 1$,

$$\mathsf{DTIME}_h[n^{e_k}] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}].$$

Proof. When $k = 1$,

$$\mathsf{DTIME}_h[n^{e_k}] \subseteq (\exists\ n)\mathsf{DTISP}[n^2, n^{.5}] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}],$$

by Lemma 5.1 (DTIME_h to DTISP) and Lemma 2.1 (DTISP Speedup), respectively. For the inductive step, we have

$$\begin{aligned}
\mathsf{DTIME}_h[n^{1+e/(2c)}] &\subseteq (\exists\ n)\mathsf{DTISP}[n^{1+e/(2c)}, n^{e/(2c)}] \quad (\mathsf{DTIME}_h \text{ to } \mathsf{DTISP}) \\
&\subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n^{e/c}, n^{e/(2c)}] \quad (\text{Speedup}) \\
&\subseteq (\exists\ n)\mathsf{DTIME}_h[n^e] \quad (\mathsf{DTIME}_h \text{ Slowdown}) \\
&\subseteq (\exists\ n)(\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}] = (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}],
\end{aligned}$$

where the last inclusion holds by the induction hypothesis. □

It is easy to see that any proof annotation for $\mathsf{DTIME}_h[n^{e_k}] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}]$ has the form $(1\ 1\ 0)^{k-1}\ 1\ 1$, by following the steps of the above proof.

Corollary 5.2 *For all* $\varepsilon > 0$ *and* $c \geq 1$, *if* $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_h[n^c]$ *then* $\mathsf{DTIME}_h[n^{1+1/(2c-1)-\varepsilon}] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}]$.

Proof. For all $e < 1 + 1/(2c-1)$, we have that $e < 1 + e/(2c-1)$. In fact, the sequence $s_k = 1 + s_{k-1}/(2c-1)$ converges to $1 + 1/(2c-1)$ for all $c \geq 1$. (Note $e = 1 + e/(2c)$ implies $e = 1/(1-1/(2c)) = 2c/(2c-1) = 1+1/(2c-1)$.) Therefore, for any $e^* < 1+1/(2c-1)$, by setting

$e = 1.5$ and observing that $\mathsf{DTIME}_h[n^{1.5}] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}]$, we can apply Lemma 5.4 a constant number of times to get that the same containment holds for $\mathsf{DTIME}_h[n^{e^*}]$. $\square$

Intuitively, the corollary tells us that if we make stronger and stronger assumptions about how quickly SAT can be solved on a hybrid TM, then we can place more and more of $\mathsf{DTIME}_h[n^{O(1)}]$ in the class $(\exists\ n)(\forall\ \log n)\mathsf{DTISP}[n, n^{.5}]$, when $c < 1.5$. We are now ready to prove the new time lower bound.

Proof of Theorem 5.1. Let $a \geq 1$ be a parameter to set later. Following the optimal normal form annotations:

$$\begin{aligned}
\mathsf{DTIME}_1[n^a] &\subseteq (\exists\ n)\mathsf{DTISP}[n^a, n^{a-1}] \quad (\mathsf{DTIME}_h \text{ to } \mathsf{DTISP}) \\
&\subseteq (\exists\ n^{x+(a-1)})^{x+(a-1)}(\forall\ n)^1\mathsf{DTISP}[n^{a-x}, n^{a-1}] \quad \text{(Speedup)} \\
&= (\exists\ n^{x+a-1})^{x+(a-1)}(\forall\ n)^1\mathsf{DTISP}[n^{a-x}, n^{a-1}],
\end{aligned}$$

where x is a parameter satisfying $x + a \geq 2$. Now by applying Speedup again, the above is in

$$\begin{aligned}
&(\exists\ n^{x+a-1})^{x+(a-1)}(\forall\ n^{a-1+(2-a)})^1(\exists\ n)^1\mathsf{DTISP}[n^{a-x-(2-a)}, n^{a-1}] \\
\subseteq\ &(\exists\ n^{x+a-1})^{x+(a-1)}(\forall\ n)^1\mathsf{DTIME}_1[n^{c(a-x-(2-a))}] \quad \text{(Slowdown)}.
\end{aligned}$$

Suppose that a and x satisfy $c(2a - 2 - x) = 1 + 1/(2c - 1) - \varepsilon$, for some $\varepsilon > 0$. Then we can apply Corollary 5.2, and the above is contained in

$$\begin{aligned}
(\exists\ n^{x+a-1})(\forall\ n)(\exists\ n)\mathsf{DTISP}[n^1, n^{.5}] &\subseteq (\exists\ n^{x+a-1})(\forall\ n)\mathsf{DTISP}[n^c, n^{1/2}] \quad \text{(Slowdown)} \\
&\subseteq (\exists\ n^{x+a-1})\mathsf{DTIME}_h[n^{c^2}]. \quad \text{(Slowdown)}
\end{aligned}$$

Note the input to the DTIME_h class is now n^{x+a-1}. Suppose further that a and c satisfy the equation $c^2/(x + a - 1) = 1 + 1/(2c - 1) - \varepsilon$. Then we can apply Corollary 5.2 again, getting

$$\begin{aligned}
(\exists\ n^{x+a-1})(\forall\ n)\mathsf{DTISP}[n^{(x+a-1)}, n^{(x+a-1)/2}] &\subseteq (\exists\ n^{x+a-1})\mathsf{DTIME}_h[n^{c(x+a-1)}] \quad \text{(Slowdown)} \\
&\subseteq \mathsf{DTIME}_h[n^{c^2(x+a-1)}]. \quad \text{(Slowdown)}.
\end{aligned}$$

We set $a = c^2(x+a-1)$, so that the first DTIME_h class equals the last one, and we get a contradiction with Lemma 5.3.

Observe that a proof annotation for the above has the form $[1\ 1\ 1\ 0\ (1\ 1\ 0)^*\ 1\ 1\ 0\ 0\ (1\ 1\ 0)^*\ 1\ 1\ 0\ 0]$, or $[1\ 1\ 1\ (0\ 1\ 1)^k\ 0\ (0\ 1\ 1)^\ell\ 0\ 0]$, as desired. The analysis introduced three parameters (c, a, x) along with three equations to satisfy. Numerically solving the system of equations

$$\begin{aligned}
a &= r^2(x + a - 1) \\
r(2a - 2 - x) &= 1 + 1/(2r - 1) \\
r/(x + a - 1) &= 1 + 1/(2r - 1)
\end{aligned}$$

under the constraint that $r \geq 1$, we obtain the unique solution

$$\{r = 1.300942884, x = .2784778050, a = 1.763503141\}.$$

(Observe that this solution also satisfies the additional constraint $x + a \geq 2$ that arose in the analysis.) One can deduce from the equations above that r is a root of the quintic $p(x) = 12x^5 - 12x^4 - x^3 - 13x + 4x^2 + 2$.

Finally, for any $c < r$, we can find a, x, and $\varepsilon > 0$ that satisfy $c(2a-2-x) = 1+1/(2c-1)-\varepsilon$, $a = c^2(x+a-1)$, and $c^2/(x+a-1) = 1+1/(2c-1)-\varepsilon$. This completes the proof. $\square$

Given the experimental results, we feel safe in conjecturing that our above theorem is optimal for alternation-trading proofs.

6 Conclusion

We have introduced a general methodology for automatically proving time lower bounds for many different scenarios in the alternation-trading framework.[9] The key contributions of this work are an improved formalization of the tools used in the framework, and the result that if we fix a parameter c and sequence of proof rules, the task of finding an alternation-trading proof that uses the rule sequence to obtain an $\Omega(n^c)$ lower bound can often be posed as a linear programming instance. This makes it feasible to search deeply through the space of possible alternation-trading proofs, and attain a much greater understanding of the limits and potential of current techniques. Implementing a small-scale theorem prover, we discovered

1. overwhelming empirical evidence that the $\Omega(n^{2\cos(\pi/7)})$ lower bound for satisfiability on $n^{o(1)}$ space algorithms is actually *optimal* for the current framework,
2. an $\Omega(n^{4^{1/3}-o(1)})$ lower bound for solving tautologies with nondeterministic algorithms, and
3. an $\Omega(n^{1.3009})$ lower bound for solving NP-complete problems with a hybrid Turing machine model, which has random access to its input, random access to $n^{o(1)}$ storage, and sequential access to a worktape.

Our approach applies to a wider variety of problems than those discussed here, in particular problems higher up in the polynomial hierarchy such as k-QBF (the problem of solving Quantified Boolean Formulas with at most k quantifier blocks).

Previously, the problem of finding the right setting of parameters to get a good lower bound was often an excruciating technical exercise. We believe that our work should reduce the load on future researchers in this area: once a new speedup or slowdown theorem for a class is found, one only needs to find the relevant linear programming formulation in order to discover how the new theorem can contribute. An automated proof system drastically reduces the time spent on tweaking parameters and allows us increase our focus on finding new tools.

Let us end with a philosophical remark. When we talk about our area's relation with other branches of computer science, we typically speak of "applying theory to practice." This work reminds us that the relationship need not be asymmetric, and we should also look out for ways to *apply practice to theory.*

[9] In fact, the only alternation-trading proof we know of that does not fit in our scheme is the Paul-Pippenger-Szemeredi-Trotter theorem that $\mathsf{NTIME}[n] \neq \mathsf{DTIME}[n]$. This is because the version of the speedup theorem given in that work does not contain a free parameter, and it is unclear how to prove a parameterized variant. Note there *is* a parameterized variant of Hopcroft-Paul-Valiant, namely $\mathsf{DTIME}[t] \subseteq \Sigma_s\mathsf{TIME}[t/\log s]$ ([DT83],p.340).

Acknowledgements

Numerous discussions were helpful, and I owe thanks for every one of them. I am especially grateful to Dieter van Melkebeek, Ryan O'Donnell, Manuel Blum, and Steven Rudich for their invaluable feedback on my PhD thesis which included some of the results in this work.

References

[ADH97] L. Adleman, J. DeMarrais, and M. Huang. Quantum computability. *SIAM Journal on Computing* 26:1524-1540, 1997.

[AKRRV01] E. Allender, M. Koucky, D. Ronneburger, S. Roy, and V. Vinay. Time-Space Tradeoffs in the Counting Hierarchy. In *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 295–302, 2001.

[AS80] D. B. Arnold and M. R. Sleep. Uniform random generation of balanced parenthesis strings. *ACM Transactions on Programming Languages and Systems* 2(1):122–128, 1980.

[BBS86] L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing* 15:364–383, 1986.

[BM78] A. Bruss and A. R. Meyer. On time-space classes and their relation to the theory of real addition. *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 233–239, 1978.

[CKS81] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM* 28(1):114–133, 1981.

[CS76] A. K. Chandra and L. J. Stockmeyer. Alternation. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)* 98–108, 1976.

[Coo88] S. A. Cook. Short Propositional Formulas Represent Nondeterministic Computations. *Information Processing Letters* 26(5): 269-270, 1988.

[DvM06] S. Diehl and D. van Melkebeek. Time-Space Lower Bounds for the Polynomial-Time Hierarchy on Randomized Machines. *SIAM Journal on Computing* 36: 563-594, 2006.

[DvMW07] S. Diehl, D. van Melkebeek, and R. Williams. A New Time-Space Lower Bound for Nondeterministic Algorithms Solving Tautologies. University of Wisconsin-Madison, Department of Computer Sciences, Technical Report 1601, 2007.

[DT83] P. W. Dymond and M. Tompa. Speedups of Deterministic Machines by Synchronous Parallel Machines. *Proceedings of ACM Symposium on Theory of Computing (STOC)* 336–343, 1983.

[For97] L. Fortnow. Nondeterministic Polynomial Time Versus Nondeterministic Logarithmic Space: Time-Space Tradeoffs for Satisfiability. In *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 52–60, 1997.

[FvM00] L. Fortnow and D. van Melkebeek. Time-Space Tradeoffs for Nondeterministic Computation. In *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 2–13, 2000.

[FLvMV05] 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.

[HPV77] J. Hopcroft, W. Paul, and L. Valiant. On time versus space. *Journal of the ACM (JACM)* 24(2):332–337, 1977.

[Kan83] R. Kannan. Alternation and the power of nondeterminism. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 344–346, 1983.

[Kan84] R. Kannan. Towards Separating Nondeterminism from Determinism. *Mathematical Systems Theory* 17(1):29–45, 1984.

[Kar84] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica* 4:373–395, 1984.

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademia Nauk SSSR*, 1093–1096, 1979.

[LV99] R. J. Lipton and A. Viglas. On the Complexity of SAT. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 459–464, 1999.

[MS87] W. Maass and A. Schorr. Speed-Up of Turing Machines with One Work Tape and a Two-Way Input Tape. *SIAM Journal on Computing* 16(1):195–202, 1987.

[vM04] 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.

[vM07] D. van Melkebeek. A Survey of Lower Bounds for Satisfiability and Related Problems. To appear in *Foundations and Trends in Theoretical Computer Science*, 2007.

[vMR05] D. van Melkebeek and R. Raz. A Time Lower Bound for Satisfiability. *Theoretical Computer Science* 348(2-3):311–320, 2005.

[vMW07] D. van Melkebeek and T. Watson. A quantum time-space lower bound for the counting hierarchy. Technical Report 1600, Department of Computer Sciences, University of Wisconsin-Madison, 2007.

[Nep70] V. Nepomnjascii. Rudimentary predicates and Turing calculations. *Soviet Math. Doklady* 11:1462–1465, 1970.

[PPST83] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On determinism versus nondeterminism and related problems. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 429–438, 1983.

[Pap94] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Sch78] C. Schnorr. Satisfiability is quasilinear complete in NQL. *Journal of the ACM* 25(1):136–145, 1978.

[Tou01] I. Tourlakis. Time-Space Tradeoffs for SAT on Nonuniform Machines. *Journal of Computer and System Sciences* 63(2):268–287, 2001.

[Vio07] E. Viola. On approximate majority and probabilistic time. In *Proceedings of the 22th IEEE Conference on Computational Complexity (CCC)*, 2007.

[Wil06] R. Williams. Inductive Time-Space Lower Bounds for SAT and Related Problems. *Computational Complexity* 15:433–470, 2006.

[Wil07a] R. Williams. Time-Space Tradeoffs for Counting NP Solutions Modulo Integers. In *Proceedings of the IEEE Conference on Computational Complexity (CCC)*, 70–82, 2007.

[Wil07b] R. Williams. Algorithms and Resource Requirements for Fundamental Problems. Ph.D. Thesis, Carnegie Mellon University, CMU-CS-07-147, August 2007.