Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 669-680
www.stacs-conf.org

ALTERNATION-TRADING PROOFS, LINEAR PROGRAMMING, AND LOWER BOUNDS (EXTENDED ABSTRACT)

RYAN WILLIAMS [1]

[1] IBM Almaden Research Center
650 Harry Road, San Jose, CA, USA 95120
E-mail address: ryanwill@us.ibm.com
URL: http://www.cs.cmu.edu/ ryanw/

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. The proofs of these lower bounds follow a certain proof-by-contradiction strategy that we call *alternation-trading*. An important open problem is to determine how powerful such proofs can possibly be.

We propose a methodology for studying these proofs that makes them amenable to both formal analysis and automated theorem proving. 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. Implementing a small-scale theorem prover based on this result, we extract new human-readable time lower bounds for several problems. This framework can also be used to prove concrete limitations on the current techniques.

1. Introduction

Many known lower bounds for natural problems follow a type of algorithmic argument that we call a *resource-trading proof*. Such a proof assumes that a hard problem *can* be solved by a "good" algorithm, and tries to derive a contradiction by combining two essential components. One is a *speedup lemma*, which simulates all good algorithms super-efficiently on some "interesting" computational model, trading time for some resource. The second component is a *slowdown lemma*, which uses the assumed good algorithm for the hard problem to simulate computations from the "interesting" model by good algorithms, thereby trading the "interesting" resource for more time. Clever combinations of speedup and slowdown lemmas are used to contradict a known result, in particular some complexity

1998 ACM Subject Classification: F.2.3, I.2.3.

Key words and phrases: time-space tradeoffs, lower bounds, alternation, linear programming.

This material is based on work supported in part by NSF grant CCR-0122581 while the author was a student at Carnegie Mellon University, and NSF grant CCF-0832797 while the author was a member of the Institute for Advanced Study.

27th Symposium on Theoretical Aspects of Computer Science, Nancy, 2010
Editors: Jean-Yves Marion, Thomas Schwentick
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.STACS.2010.2494

hierarchy theorem. That is, by assuming a "good" algorithm for a hard problem, we derive something like $\mathsf{TIME}[n^2] \subseteq \mathsf{TIME}[n]$, a contradiction.

As an example, one can prove a time-space tradeoff for satisfiability (SAT) as follows. Assume SAT has an algorithm running in n^c time and $\text{poly}(\log n)$ space, for some $c > 1$. One speedup lemma is that computations running in n^a time and $\text{poly}(\log n)$ space can be simulated by an *alternating machine* that switches from co-nondeterministic mode to nondeterministic mode once (i.e., a Π_2 machine), and runs in $n^{a/2+o(1)}$ time. This speedup lemma *trades time for alternations*. The relevant slowdown lemma is: if SAT has an n^c time, $\text{poly}(\log n)$ space algorithm, then (by a strengthening of the Cook-Levin theorem) every language in $\mathsf{NTIME}[t]$ has $t^{c+o(1)}$ time, $\text{poly}(\log t)$ space algorithms. Consequently, an alternating machine running in t time and making $k-1$ alternations has $t^{c^k+o(1)}$ time, $\text{poly}(\log t)$ space algorithms. Combining these speedup and slowdown lemmas, we derive

$$\Sigma_2\mathsf{TIME}[t] \subseteq \mathsf{DTISP}[t^{c^2+o(1)}, \text{poly}(\log t)] \subseteq \Pi_2\mathsf{TIME}[t^{c^2/2}],$$

where the first inclusion holds by slowdown and the second holds by speedup. Now observe that the alternating time hierarchy is contradicted when $c^2 < 2$. This proof is the $n^{\sqrt{2}-\varepsilon}$ time lower bound of Lipton and Viglas [LV99].

Some of the best known separations in complexity theory use resource-trading proofs. Hopcroft, Paul, and Valiant [HPV77] showed that $\mathsf{SPACE}[n] \not\subseteq \mathsf{DTIME}[o(n \log n)]$ for multitape Turing machines, by proving the "speedup lemma" that $\mathsf{DTIME}[t] \subseteq \mathsf{SPACE}[t/\log t]$ and invoking diagonalization. Their result was later extended to general models [PR81, HLMW86]. Paul, Pippenger, Szemeredi, and Trotter [PPST83] proved that $\mathsf{NTIME}[n] \neq \mathsf{DTIME}[n]$ for multitape Turing machines. The key component in the proof is the "speedup lemma" $\mathsf{DTIME}[t] \subseteq \Sigma_4\mathsf{TIME}[t/\log^* t]$ for multitape TMs. Despite their age, the above separations still constitute the best known progress on P vs PSPACE and P vs NP, respectively.

In more recent years, resource-trading proofs have established time-space lower bounds for NP-complete problems and problems higher in the polynomial hierarchy [Kan84, For97, LV99, FvM00, FLvMV05, Wil06, Wil08]. For instance, the best known time lower bound for solving SAT with $n^{o(1)}$-space algorithms is $n^{2\cos(\pi/7)-o(1)} \geq n^{1.801}$, obtained with a resource-trading proof [Wil08]. (Note if one could improve the 1.801 exponent to arbitrary constants, one would separate $\mathsf{LOGSPACE}$ from NP.) For nondeterministic algorithms using $n^{o(1)}$ space, the best known time lower bound for solving the coNP-complete TAUTOLOGY problem was $n^{\sqrt{2}-o(1)}$ for several years [FvM00]. Certain time-space lower bounds for probabilistic and quantum computations also follow the resource-trading paradigm [AKRRV01, DvM06, Vio09, vMW07]. Resource-trading proofs are also abound in the multidimensional "hybrid" Turing machine model, which has read-only random access to its input and an $n^{o(1)}$ read-write store, as well as read-write two-way access to a d-dimensional tape for some $d \geq 1$. This is the most powerful (and physically realistic) model known where we still know nontrivial time lower bounds for problems such as SAT. Multidimensional TMs have a long history; e.g., [Lou80, PR81, Kan83, MS87, vMR05, Wil06] proved lower bounds for them. (For a more complete literature review, please see the full version of the paper.)

1.1. Main Results

We introduce a methodology for reasoning about resource-trading proofs that is also practically implementable for finding short proofs. Informally, the "hard work" in these

proofs can be replaced by solving a series of linear programming problems. This perspective not only aids us practically in the search for new lower bounds, but also allows us to show non-trivial limitations on what can be proved.

This methodology is applied to several lower bound problems. In all cases considered here, the resource being "traded" is alternations, so we call the proofs *alternation-trading*.

Deterministic Time-Space Lower Bounds. Aided by results of a computer program, we show that any SAT algorithm running in $t(n)$ time and $s(n)$ space satisfies $t \cdot s \geq \Omega(n^{2\cos(\pi/7)-o(1)})$. Previously, the best known result was $t \cdot s \geq \Omega(n^{1.573})$ [FLvMV05]. It has been conjectured that the current framework sufficed to prove a $n^{2-o(1)}$ time lower bound for SAT, against algorithms using $n^{o(1)}$ space. We prove that it is not possible to obtain n^2 with the framework, formalizing a conjecture of [FLvMV05].* A computer search over proofs of short length suggests that the best known $n^{2\cos(\pi/7)-o(1)}$ lower bound [Wil08] is already optimal for the framework. We also prove lower bounds on QBF_k (quantified Boolean formulas with at most k quantifier blocks), showing that the problem requires $\Omega(n^{k+1-\delta_k})$ time for $n^{o(1)}$ space algorithms, where $\delta_k < 0.2$ and $\lim_{k\to\infty} \delta_k = 0$.†

Nondeterministic Time-Space Lower Bounds. Adapting our ideas to proving lower bounds for TAUTOLOGIES, a computer program found a very short proof improving upon Fortnow and Van Melkebeek's lower bound. Longer proofs suggested an interesting pattern. Joint work with Diehl and Van Melkebeek on this observation resulted in an $n^{4^{1/3}-o(1)} \geq n^{1.587}$ time lower bound [DvMW09]. Computer search suggests that this lower bound is best possible for the framework. We prove that it is not possible to obtain an n^{ϕ} time lower bound, where $\phi = 1.618\ldots$ is the golden ratio. This is surprising since we have known for some time that an n^{ϕ} lower bound *is* provable for *deterministic* algorithms [FvM00].

Multidimensional Turing Machine Lower Bounds. Here our method uncovers peculiar behavior in the best lower bound proofs, regardless of the dimension. Studying computer search results, we extract an $\Omega(n^{r_d})$ time lower bound for the d-dimensional case, where $r_d \geq 1$ is the root of a particular quintic $p_d(x)$ with coefficients depending on d. For example, $r_1 \approx 1.3009$, $r_2 \approx 1.1887$, and $r_3 \approx 1.1372$. Again, our search suggests this is best possible, and we can prove it is not possible to improve the bound for d-dimensional TMs to $n^{1+1/(d+1)}$ with the current tools.

These limitations also hold for other NP and coNP-hard problems; the only property required is that all languages in $\mathsf{NTIME}[n]$ (respectively, $\mathsf{coNTIME}[n]$) have sufficiently efficient reductions to the problem. Also our linear programming approach is not limited to the above, and can be applied to the league of lower bounds discussed in Van Melkebeek's surveys [vM04, vM07].

1.2. Some Remarks on the Reduction to Linear Programming

The key to our formulation is to separate the *discrete choices* in an alternation-trading proof from the *real-valued choices*. The discrete choices consist of the sequence of lemmas to apply in each step, and what sort of hierarchy theorem to use in the contradiction. We present several simplifications that greatly reduce the number of discrete choices, without loss of generality. The real-valued choices are the running time exponents that arise from

*That is, we formalize the statement: "...some complexity theorists feel that improving the golden ratio exponent beyond 2 would require a breakthrough" in Section 8 of [FLvMV05].

†Note the QBF_k results appeared in the author's PhD thesis in 2007 but have been unpublished to date.

the choices of time bounds and rule applications. We prove that once the discrete choices are made, the remaining real-valued problem can be expressed as an instance of linear programming. This makes it possible to search for new proofs via computer, and it also gives us a formal handle on the limitations of these proofs.

One cannot easily search over all possible proofs, as the number of discrete choices is still about $2^n/n^{3/2}$ for proofs of n lines (proportional to the nth Catalan number). Nevertheless it is still feasible to try all 24+ line proofs. These proof searches reveal patterns, indicating that certain strategies will be most successful in proving lower bounds; in each case we study, the resulting strategies differ. Following the strategies, we establish new lower bound proofs. The patterns also suggest how to show limitations on the proof systems.

Note: *Due to space limitations, we can only describe how our methods apply to SAT time-space lower bounds. Please see the full version of the paper for proofs and more details.*

2. Preliminaries

We assume familiarity with Complexity Theory, especially the notion of alternationWe use big-Ω notation in the infinitely often sense, so statements like "SAT is not in $O(n^c)$ time" are equivalent to "SAT requires $\Omega(n^c)$ time." All functions are assumed constructible within the appropriate bounds. Our default computational model is the random access machine, broadly construed: particular variants do not affect the results. $\mathsf{DTISP}[t(n), s(n)]$ is the class of languages accepted by a RAM running in $t(n)$ time and $s(n)$ space, simultaneously. For convenience, we set $\mathsf{DTS}[t(n)] := \mathsf{DTISP}[t(n)^{1+o(1)}, n^{o(1)}]$ to omit negligible $o(1)$ factors.

In order to properly formalize alternation-trading proofs, we introduce notation for alternating complexity classes that include *input constraints* between alternations. Let us start with an example of the notation, then give a general definition. Define $(\exists\ f(n))^b\mathsf{DTS}[n^a]$ to be the class of languages recognized by a machine which, on an input x of length n, writes a $f(n)^{1+o(1)}$ bit string y nondeterministically, copies at most $n^{b+o(1)}$ bits z from the pair $\langle x, y\rangle$ (in $O(n^{b+o(1)})$ time), then feeds z as input to a machine M running in $n^{a+o(1)}$ time and $n^{o(1)}$ space. Note the runtime of M is measured with respect to the initial input length n, not the latter input length $n^{b+o(1)}$ of z.

We generalize this definition as follows. Let $\mathcal{C}$ be a complexity class. For $i = 1, \ldots, k$, let $Q_i \in \{\exists, \forall\}$ and $a_i, b_i \geq 0$. Define

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathcal{C}$$

to be the class of languages recognized by a machine M that, on input x of length n, has the following general behavior on input x:

Set $z_0 := x$.
For $i = 1, \ldots, k$,
If $Q_i = \exists$, switch to *existential mode.*
If $Q_i = \forall$, switch to *universal mode.*
Guess an $n^{a_i+o(1)}$ bit string y (universally or existentially).
Copy at most $n^{b_{i+1}+o(1)}$ bits z_i from the pair $\langle z_{i-1}, y\rangle$.
End for
Run a machine recognizing a language in class $\mathcal{C}$ on the input z_k.

When an input constraint b_i is unspecified, its default value is $\max\{a_i, 1\}$. We say that the existential and universal modes of an alternating computation are *quantifier blocks*, to reflect the complexity class notation. It is crucial to observe that the time bound in the ith quantifier block is measured with respect to n, the input to the *first quantifier block*.

Notice that by simple properties of nondeterminism and conondeterminism, we can combine adjacent quantifier blocks that are of the same type, e.g., $(\exists n^a)^a(\exists n^b)^b\mathsf{DTS}[n^c] = (\exists n^{\max\{a,b\}})^b\mathsf{DTS}[n^c]$. This useful property is exploited in alternation-trading proofs.

2.1. A Short Introduction to Alternation-Trading Proofs

Here we give a brief overview of the tools used in alternation-trading proofs. In this extended abstract we focus on deterministic time lower bounds for satisfiability for algorithms using $n^{o(1)}$ workspace; the other lower bound problems use similar tools.

It is known that satisfiability of Boolean formulas in conjunctive normal form (SAT) is a complete problem under tight reductions for a small nondeterministic complexity class. The class NQL, called *nondeterministic quasilinear time*, is defined as

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

Theorem 2.1 ([Coo88, Sch78, Tou01, FLvMV05])**.** *SAT is* NQL*-complete under quasilinear time* $O(\log n)$ *space reductions, for both multitape and random access machine models. Moreover, each bit of the reduction can be computed in* $O(poly(\log n))$ *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,
- d-dimensional Turing machine using time t.

Theorem 2.1 implies that if $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[t]$, then SAT $\notin \mathcal{C}[t/\mathrm{poly}(\log t)]$.

Corollary 2.2. *If* $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[t(n)]$*, then SAT* $\notin \mathcal{C}[t(n)/\log^k t(n)]$ *for some* $k > 0$*.*

Hence we wish to prove $\mathsf{NTIME}[n] \not\subseteq \mathcal{C}[n^c]$ for large $c > 1$. To prove time-space lower bounds, we work with $\mathcal{C}[n^c] = \mathsf{DTS}[n^c] = \mathsf{DTISP}[n^c, n^{o(1)}]$. Van Melkebeek and Raz [vMR05] 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. Therefore similar time lower bounds hold for these problems as well.

Speedups, Slowdowns, and Contradictions. Now that our goal is to prove $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$, how can we do this? In an alternation-trading proof, we attempt to establish a contradiction from assuming $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$, by applying two lemmas which complement one another. A *speedup lemma* takes a $\mathsf{DTS}[t]$ class and places it in an alternating class with runtime $o(t)$. A *slowdown lemma* takes an alternating class with runtime t and places it in a class with one less alternation and runtime $O(t^c)$. The Speedup Lemma dates back to Nepomnjascii [Nep70] and Kannan [Kan84].

[‡]In the multitape Turing machine model we assume that the tape heads are already oriented on the appropriate cells, otherwise it may take linear time to find the appropriate cells on a tape.

Lemma 2.3 (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 use n^a time and n^e space. Let y be an input of length n. A complete description (i.e. *configuration*) of $M(y)$ at any step can be described in $O(n^e + \log n)$ space. To simulate M in $(\exists\ n^{x+e})^{\max\{1,x+e\}}(\forall\ \log n)^{\max\{1,e\}}\mathsf{DTISP}[n^{a-x}, n^e]$, the algorithm $N(y)$ existentially guesses a sequence of configurations $C_1, \ldots, C_{n^x}$ of $M(x)$. Then $N(y)$ appends the initial configuration C_0 of $M(y)$ to the beginning of the sequence, and an accepting configuration C_{n^x+1} to the end. $N(y)$ universally guesses a $i \in \{0, \ldots, n^x\}$, erases all configurations except C_i and C_{i+1}, then simulates $M(y)$ starting from C_i, accepting if and only if C_{i+1} is reached within n^{a-x} steps. It is easy to see the simulation is correct. The input constraints on the quantifier blocks are satisfied since after the universal guess, the input is only y, C_i, and C_{i+1}, which is of size $n + 2n^{e+o(1)} \leq n^{\max\{1,e\}+o(1)}$. ∎

Observe 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 subtle property that is exploited heavily in alternation-trading proofs. The Slowdown Lemma is the following simple result:

Lemma 2.4 (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 recognizing L. On an input x of length n, A guesses a string y of length $n^{a'+o(1)}$ and 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 $\mathsf{NTIME}[p(n)] \cup \mathsf{coNTIME}[p(n)] \subseteq \mathsf{DTISP}[p(n)^c, p(n)^e]$ for polynomials $p(n) \geq n$. Therefore A can be simulated with a deterministic algorithm B. Since the 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 2.5 (Alternating Time Hierarchy)**.** *For $k \geq 0$, for all $Q_i \in \{\exists, \forall\}$, $1 \leq a_i' < a_i$, and $1 \leq b_i' \leq b_i$,*

$$(Q_1\ n^{a_1})^{b_2}\ldots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}] \not\subseteq (R_1\ n^{a_1'})^{b_2'}\ldots^{b_k'}(R_k\ n^{a_k'})^{b_{k+1}'}\mathsf{DTS}[n^{a_{k+1}'}],$$

where $R_i \in \{\exists, \forall\}$ and $R_i \neq Q_i$ for all $i = 2, \ldots, k+1$.

Two Examples. Let us give a couple of examples of alternation-trading proofs. To simplify the presentation we do not specify the input constraints to quantifiers in the below.

(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 $\varepsilon > 0$. By Theorem 2.1, if SAT is in $n^{\sqrt{2}-\varepsilon}$ time and $n^{o(1)}$ space then $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ with $c^2 < 2$. We have

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

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 2.5. 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 the ε can be removed from the previous statement and state that SAT cannot be solved in $n^{\sqrt{2}}$ time and $n^{o(1)}$ exactly.[§]

(2) Improving on the previous example, one can show SAT $\notin \mathsf{DTS}[n^{1.6004}]$. If $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and $\sqrt{2} \le c < 2$, then applying the Speedup and Slowdown Lemmas one can derive:

$$\begin{array}{lcll}\mathsf{DTS}[n^{c^2/2+2}] & \subseteq & (\exists\ n^{c^2/2})(\forall\ \log n)\mathsf{DTS}[n^2] & \text{(Speedup)}\\ & \subseteq & (\exists\ n^{c^2/2})(\forall\ \log n)(\forall\ n)(\exists\ \log n)\mathsf{DTS}[n] & \text{(Speedup)}\\ & = & (\exists\ n^{c^2/2})(\forall\ n)(\exists\ \log n)\mathsf{DTS}[n] & \text{(Combining } \forall \text{ Quantifiers)}\\ & \subseteq & (\exists\ n^{c^2/2})(\forall\ n)\mathsf{DTS}[n^c] & \text{(Slowdown)}\\ & \subseteq & (\exists\ n^{c^2/2})\mathsf{DTS}[n^{c^2}] & \text{(Slowdown)}\\ & \subseteq & (\exists\ n^{c^2/2})(\exists\ n^{c^2/2})(\forall\ \log n)\mathsf{DTS}[n^{c^2/2}] & \text{(Speedup)}\\ & = & (\exists\ n^{c^2/2})(\forall\ \log n)\mathsf{DTS}[n^{c^2/2}] & \text{(Combining } \exists \text{ Quantifiers)}\\ & \subseteq & (\exists\ n^{c^2/2})\mathsf{DTS}[n^{c^3/2}] & \text{(Slowdown)}\\ & \subseteq & \mathsf{DTS}[n^{c^4/2}] & \text{(Slowdown)}\end{array}$$

When $c^2/2 + 2 > c^4/2$ (which happens if $c < 1.6004$), we have $\mathsf{DTS}[n^a] \subseteq \mathsf{DTS}[n^{a'}]$ for some $a > a'$. One can 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.

Example (2) was discovered by a computer program. By "discovered", we mean that the program applied speedups and slowdowns in precisely the same way, having only minimum knowledge of the lemmas. Furthermore, the program verified that above is the *best possible* alternation-trading proof that applies the Speedup and Slowdown Lemmas at most 7 times. A more formal definition of "alternation-trading proof" is given in the next section.

3. Formalizing Alternation-Trading Proofs

We formalize alternation-trading proofs of lower bounds on DTS classes as follows:[¶]

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

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}], \tag{3.1}$$

[§]Suppose $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and $\Sigma_2\mathsf{TIME}[n] \subseteq \Pi_2\mathsf{TIME}[n^{1+o(1)}]$. The first assumption, along with the Speedup and Slowdown Lemmas, implies that for every k there's a K satisfying $\Sigma_2\mathsf{TIME}[n^k] \subseteq \mathsf{NTIME}[n^{kc}] \subseteq \Sigma_K\mathsf{TIME}[n]$. But the second assumption implies that $\Sigma_K\mathsf{TIME}[n] = \Sigma_2\mathsf{TIME}[n^{1+o(1)}]$. Hence $\Sigma_2\mathsf{TIME}[n^k] \subseteq \Sigma_2\mathsf{TIME}[n^{1+o(1)}]$, which contradicts the time hierarchy for $\Sigma_2\mathsf{TIME}$.

[¶]This formalization has *implicitly* appeared in several prior works, but not to the degree that we investigate in this paper.

where $k \geq 0$, $Q_i \in \{\exists, \forall\}$, $Q_i \neq Q_{i+1}$, $a_i > 0$, 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 four possible forms:

Speedup Rule 0: For $k = 0$ and any $x \in (0, a_1)$, $(Q_0\ n^x)^{\max\{x,1\}}(Q_1\ n^0)^1\mathsf{DTS}[n^{a_1-x}]$.[‖]

Speedup Rule 1: For $k > 0$ and any $x \in (0, a_{k+1})$,

$$(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x\}})^{\max\{x,b_{k+1}\}}(Q_{k+1}\ n^0)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

Speedup Rule 2: For $k > 0$ and any $x \in (0, a_{k+1})$,

$$(Q_1\ n^{a_1})^{b_2}\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_{k+1}\ n^x)^{\max\{x,b_{k+1}\}}(Q_{k+2}\ n^0)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}-x}].$$

Slowdown Rule: For $k > 0$,

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

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

The above definition comes directly from the Speedup Lemma (Lemma 2.3) and Slowdown Lemma (Lemma 2.4). The rules are easily verified to be syntactic formulations of the corresponding lemmas. For instance, Speedup Rule 1 holds, as

$$\begin{aligned}&(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]\\ \subseteq\ &(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{a_k})^{b_{k+1}}(Q_k\ n^x)^{\max\{b_{k+1},x\}}(Q_{k+1}\ n^0)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}]\\ \subseteq\ &(Q_1\ n^{a_1})^{b_2}(Q_2\ n^{a_2})\cdots^{b_k}(Q_k\ n^{\max\{a_k,x\}})^{\max\{b_{k+1},x\}}(Q_{k+1}\ n^0)^{b_{k+1}}\mathsf{DTS}[n^{a_{k+1}}].\end{aligned}$$

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 Lemma 2.4. It follows that alternation-trading proofs are sound.

Note Speedup Rules 0 and 2 add two quantifier blocks, Speedup Rule 1 adds one quantifier, and all three rules introduce a parameter x. By considering "normal form" proofs (defined in the following paragraphs), we can prove that Rule 2 can always be replaced by applications of Rule 1. (A proof is in the full version of the paper.) For this reason we just refer to *the Speedup Rule*, depending on which of Rule 0 or Rule 1 applies.

Define a class of the form (3.1) to be *simple*. Define classes A_1 and A_2 to be *complementary* if A_1 is the class of complements of languages in A_2. Every known (model-independent) time-space lower bound for SAT shows "$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $A_1 \subseteq A_2$", for some complementary simple classes A_1 and A_2, contradicting a time hierarchy (cf. Theorem 2.5). A similar claim holds for nondeterministic time-space lower bounds against tautologies (which prove "$\mathsf{NTIME}[n] \subseteq \mathsf{coNTS}[n^c]$ implies $A_1 \subseteq A_2$"), for d-dimensional TM lower bounds (which prove "$\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}_d[n^c]$ implies $A_1 \subseteq A_2$"), and other problems.

Normal Form. It will be very convenient to introduce a *normal form* for alternation-trading proofs. We show that any lower bound provable with complementary simple classes can also be established with a normal form proof. This greatly reduces the degrees of freedom in a proof, as we no longer need to worry about *which* time hierarchy to contradict.

[‖]Please note that the $(k+1)$th quantifier is n^0 in order to account for the $O(\log n)$ size of the quantifier.

Definition 3.2. Let $c \geq 1$. An alternation-trading proof for c is in *normal form* if (a) the first and last lines are $\mathsf{DTS}[n^a]$ and $\mathsf{DTS}[n^{a'}]$ respectively, for some $a \geq a'$, and (b) no other lines are DTS classes.

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 $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$.*

Theorem 3.4. *Let A_1 and A_2 be complementary. If there is an alternation-trading proof P for c that shows $(\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow A_1 \subseteq A_2)$, then there is a normal form proof for c, of length at most that of P.*

Proofs of Lemma 3.3 and Theorem 3.4 are in the full version. The upshot of these results is that we may focus our proof search on normal form proofs. For the remainder of this section, we assume all alternation-trading proofs are in normal form.

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 $(\ell-1)$-bit vector A where for all $i = 1, \ldots, \ell - 1$, $A[i] = 1$ (respectively, $A[i] = 0$) if the ith line applies a Speedup Rule (respectively, a Slowdown Rule).

An $(\ell - 1)$-bit proof annotation corresponds to a "strategy" for an ℓ-line proof. For a normal form proof of ℓ lines, it is not hard to show that its annotation A must have $A[1] = 1$, $A[\ell - 2] = 0$, and $A[\ell - 1] = 0$.

Note that an annotation *does not* determine a proof entirely, as other parameters need optimizing. (The problem of optimizing them is tackled in the next section.) To illustrate the annotation concept, we give four examples.

- The $n^{\sqrt{2}}$ lower bound of Lipton and Viglas has the annotation $[1, 0, 0]$.
- The $n^{1.6004}$ bound from Section 2.1 corresponds to $[1, 1, 0, 0, 1, 0, 0]$.
- The n^{ϕ} bound of Fortnow and Van Melkebeek [FvM00] is an inductive proof, corresponding to an infinite sequence of annotations. In normal form, the sequence is $[1, 0, 0], [1, 1, 0, 0, 0], [1, 1, 1, 0, 0, 0, 0], \ldots$
- The $n^{2\cos(\pi/7)}$ bound [Wil08] has two inductive stages. Let $A = 1, 0, 1, 0, \ldots, 1, 0, 0$, where the '…' contain any number of repetitions. The sequence is
$$[A], [1, A, A], [1, 1, A, A, A], [1, 1, 1, A, A, A, A], \ldots$$
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.

3.1. Translation To Linear Programming

Given a (normal form) proof annotation, how can we determine the best proof possible with it? We need to optimally set the runtimes of the first and last DTS classes in the proof, as well as the x_i parameters that arise from each application of a Speedup Rule. It turns out that an annotation A and $c > 1$ can be reduced to a polynomial size linear program that is feasible if and only if there is an alternation-trading proof of $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$

with annotation A. More precisely, the problem of optimizing parameters can be viewed as an arithmetic circuit evaluation, where the circuit has max gates, addition gates, and input gates that may multiply their input by c. Such circuits can be evaluated using a linear program that minimizes the sum of the gate values (cf. [Der72]).

Let A be an annotation of $\ell - 1$ bits, and let m be the maximum number of quantifier blocks in any line of A (note m is easily computed in linear time). The target LP has variables $a_{i,j}$, $b_{i,j}$, and x_i, for all $i = 0, \ldots, \ell - 1$ and $j = 1, \ldots, m$. The variables $a_{i,j}$ represent the runtime exponent of the jth quantifier block in the class on the ith line, $b_{i,j}$ is the input exponent 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 Speedup Rule. For example:

- If the kth line of a proof is $\mathsf{DTS}[n^a]$, the corresponding constraints are
$$a_{k,1} = a,\ b_{k,1} = 1,\ (\forall k > 0)\ a_{k,i} = b_{k,i} = 0.$$
- If the kth line of a proof is $(\exists\ n^{a'})^b\mathsf{DTS}[n^a]$, then the constraints are
$$a_{k,0} = a,\ b_{k,1} = b,\ a_{k,1} = a',\ b_{k,1} = 1,\ (\forall k > 1)\ a_{k,i} = b_{k,i} = 0.$$

The objective is to minimize $\sum_{i,j}(a_{i,j} + b_{i,j}) + \sum_i x_i$. The LP constraints depend on the lines of the annotation, as follows.

Initial Constraints. For the 0th and $(\ell - 1)$th lines we have $a_{0,1} \geq a_{\ell-1,1}$, and

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

representing $\mathsf{DTS}[n^{a_{0,1}}]$ and $\mathsf{DTS}[n^{a_{\ell-1,0}}]$, respectively. The 1st line of a proof always applies Speedup Rule 1, having the form $(Q_1 n^x)^{\max\{x,1\}}(Q_2\ n^0)^1\mathsf{DTS}[n^{a-x}]$. So the constraints for the 1st line are:

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

The below constraint sets simulate the Speedup and Slowdown Rules:

Speedup Rule Constraints. For the ith line where $i > 1$ and $A[i] = 1$, we have

$$a_{i,1} \geq 1,\ a_{i,1} \geq a_{i-1,1} - x_i,\ b_{i,1} = b_{i-1,1},\ a_{i,2} = 0,\ b_{i,2} \geq x_i,\ b_{i,2} \geq b_{i-1,1},\ a_{i,3} \geq a_{i-1,2},$$
$$a_{i,3} \geq x_i,\ b_{i,3} \geq b_{i-1,2},\ (\forall\ k :\ 4 \leq k \leq m)\ a_{i,k} = a_{i-1,k-1}, b_{i,k} = b_{i-1,k-1}.$$

The constraints express that $\cdots\ ^{b_2}(Q_2\ n^{a_2})^{b_1}\mathsf{DTS}[n^{a_1}]$ in the $(i-1)$th line is replaced by

$$\cdots\ ^{b_2}(Q_2\ n^{\max\{a_2,x\}})^{\max\{x,b_1\}}(Q_1\ n^0)^{b_1}\mathsf{DTS}[n^{\max\{a_1-x,1\}}]$$

in the ith line, where Q_1 is opposite to Q_2.

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

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

These express the replacement of $\cdots\ ^{b_2}(Q_1 n^{a_2})^{b_1}\mathsf{DTS}[n^{a_1}]$ in the $(i-1)$th line with

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

in the ith line.

This concludes the description of the linear program. To find the largest c that still yields a feasible LP, we can simply binary search for it. The following summarizes this section.

Theorem 3.6. *Given an annotation of n lines, the best possible alternation-trading proof following the annotation can be determined up to n digits of precision, in* $\mathrm{poly}(n)$ *time.*

3.2. Results

Following the above formulation, we wrote proof search routines in Maple. Many millions of proof annotations were tried, including all those corresponding to prior work, with no success beyond the $2\cos(\pi/7)$ exponent. The best lower bounds followed a highly regular pattern; see the full version for more on this. We are led to:

Conjecture 3.7. *There is no alternation-trading proof that* $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$ *for all* $c > 2\cos(\pi/7)$.

Proving the conjecture seems currently out of reach. However, we can show:

Theorem 3.8. *There is no alternation-trading proof that* $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^2]$.

A proof is in the full version. At a high level, the proof argues that any minimum length proof of a quadratic lower bound could be shortened, giving a contradiction.

Despite this bad news, the theorem prover did provide enough insight to aid in a new lower bound of $n^{2\cos(\pi/7)-o(1)}$ on the time-space product of any SAT algorithm.

Theorem 3.9. *Let* $t(n)$ *and* $s(n)$ *be bounded above by polynomials. Any algorithm solving SAT in time* t *and space* s *requires* $t \cdot s = \Omega(n^{2\cos(\pi/7)-\varepsilon})$ *for all* $\varepsilon > 0$.

These lower bounds have also been generalized to the QBF problem:

Theorem 3.10. *For all* $k \geq 1$, QBF_k *requires* $\Omega(n^c)$ *time on* $n^{o(1)}$ *space RAMs, where* $c^3/k - c^2 - 2c + k < 0$.

4. Discussion

We introduced a methodology for reasoning about alternation-trading proofs of lower bounds. It provides a generic means for computers to help us attack lower bound problems, and lets us establish limitations on known techniques. We now have a better understanding of what these techniques can and cannot do, and a tool for addressing future problems. Previously, the problem of setting parameters to achieve a good lower bound was a highly technical exercise. Our work should facilitate further research: once a new speedup or slowdown lemma is found, one only needs to find the relevant linear programming formulation to begin understanding its power. We conclude with two open-ended problems.

(1) *Establish tight limitations for alternation-trading proofs.* That is, show that the best possible alternation-trading proofs match those we have provided. Our computer search results have been met with healthy skepticism. It is critical to verify these perceived limitations with formal proof. We have managed to prove non-trivial limitations; it is possible that the ideas in those can be extended.

(2) *Discover new ingredients to add to the framework.* One possibility is to find new separation results that lead to new contradictions. Another is to find improved Speedup and/or Slowdown Lemmas. The Slowdown Lemmas are the "blandest" of the ingredients, in that they are the most elementary (and they relativize).

Acknowledgements. I am grateful to my thesis committee for their invaluable feedback on my PhD thesis, which included preliminary results on this work. Thanks to Scott Aaronson for useful discussions about irrelativization, and thanks to the STACS referees for very thoughtful comments.

References

[AKRRV01] E. Allender, M. Koucky, D. Ronneburger, S. Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proc. IEEE Conference on Computational Complexity (CCC)*, 295–302, 2001.

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

[Coo88] S. A. Cook. Short propositional formulas represent nondeterministic computations. *IPL* 26(5): 269-270, 1988.

[Der72] C. Derman. *Finite state Markov decision processes.* Academic Press, 1972.

[DvM06] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM J. Computing* 36: 563-594, 2006.

[DvMW09] S. Diehl, D. van Melkebeek, and R. Williams. An improved time-space lower bound for tautologies. In *Proc. of Computing and Combinatorics (COCOON)*, Springer LNCS 5609, 429–438, 2009.

[For97] L. Fortnow. Nondeterministic polynomial time versus nondeterministic logarithmic space. In *Proc. IEEE Conference on Computational Complexity (CCC)*, 52–60, 1997.

[FvM00] L. Fortnow and D. van Melkebeek. Time-Space Tradeoffs for Nondeterministic Computation. In *Proc. 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. *JACM* 52(6):835–865, 2005.

[HLMW86] J. Y. Halpern, M. C. Loui, A. R. Meyer, and D. Weise. On Time versus Space III. *Mathematical Systems Theory* 19(1):13–28, 1986.

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

[Kan83] R. Kannan. Alternation and the power of nondeterminism. In *Proc. ACM STOC*, 344–346, 1983.

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

[LV99] R. J. Lipton and A. Viglas. On the complexity of SAT. In *Proc. IEEE FOCS*, 459–464, 1999.

[Lou80] M. C. Loui. Simulations among multidimensional Turing machines. Ph.D. Thesis, Massachusetts Institute of Technology TR-242, 1980.

[MS87] W. Maass and A. Schorr. Speed-up of Turing machines with one work tape and a two-way input tape. *SIAM J. Computing* 16(1):195–202, 1987.

[vM04] D. van Melkebeek. Time-space lower bounds for NP-complete problems. In *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. *Foundations and Trends in Theoretical Computer Science* 2(3):197–303, 2007.

[vMR05] D. van Melkebeek and R. Raz. A time lower bound for satisfiability. *TCS* 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.

[PR81] W. Paul and R. Reischuk. On time versus space II. *JCSS* 22:312–327, 1981.

[PPST83] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On determinism versus nondeterminism and related problems. In *Proc. IEEE FOCS*, 429–438, 1983.

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

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

[Vio09] E. Viola. On approximate majority and probabilistic time. *Computational Complexity* 18(3):337–375, 2009.

[Wil06] R. Williams. Inductive time-space lower bounds for SAT and related problems. *Computational Complexity* 15:433–470, 2006.

[Wil07] R. Williams. Algorithms and resource requirements for fundamental problems. Ph.D. Thesis, Carnegie Mellon University, CMU-CS-07-147, August 2007.

[Wil08] R. Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Computational Complexity* 17(2):179–219, 2008.