Time-Space Lower Bounds for the Polynomial-Time Hierarchy on Randomized Machines*

Scott Diehl† Dieter van Melkebeek‡
University of Wisconsin, Madison, WI 53706, USA

May 17, 2006

Abstract

We establish the first polynomial-strength time-space lower bounds for problems in the linear-time hierarchy on randomized machines with two-sided error. We show that for any integer $\ell > 1$ and constant $c < \ell$, there exists a positive constant d such that QSAT_ℓ cannot be computed by such machines in time n^c and space n^d, where QSAT_ℓ denotes the problem of deciding the validity of a quantified Boolean formula with at most $\ell - 1$ quantifier alternations. Moreover, d approaches $1/2$ from below as c approaches 1 from above for $\ell = 2$, and d approaches 1 from below as c approaches 1 from above for $\ell \geq 3$. In fact, we establish the stronger result that for any constants $a \leq 1$ and $c < 1+(\ell-1)a$, there exists a positive constant d such that linear-time alternating machines using space n^a and $\ell - 1$ alternations cannot be simulated by randomized machines with two-sided error running in time n^c and space n^d, where d approaches $a/2$ from below as c approaches 1 from above for $\ell = 2$ and d approaches a from below as c approaches 1 from above for $\ell \geq 3$.

Corresponding to $\ell = 1$, we prove that there exists a positive constant d such that the set of Boolean tautologies cannot be decided by a randomized machine with one-sided error in time $n^{1.759}$ and space n^d. As a corollary, this gives the same lower bound for satisfiability on deterministic machines, improving on the previously best known such result.

1 Introduction

Satisfiability, the problem of deciding if a propositional formula has at least one satisfying assignment, is among the most fundamental NP-complete problems. Proving lower bounds for satisfiability remains an open problem of paramount importance to the field of computational complexity. Although it is widely conjectured that any deterministic algorithm requires exponential time to solve satisfiability, a proof of this belief seems far out of reach. The trivial linear-time lower bound, which follows from the observation that the machine must look at its entire input formula in the worst case, is the state-of-the-art bound for random-access machines. Despite several decades of effort, there has been no success in proving super-linear time lower bounds for satisfiability.

*A preliminary version of this work appeared as an extended abstract in the *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming* [7].

†sfdiehl@cs.wisc.edu, supported by NSF Career award CCR-0133693.
‡dieter@cs.wisc.edu, partially supported by NSF Career award CCR-0133693.

A few years ago, Fortnow [8] established nontrivial time lower bounds for satisfiability on machines which are restricted to use a small amount of workspace. Fortnow's technique has its roots in earlier work by Kannan [14] and has been further developed in recent years [17, 19, 24]. For example, for machines using a subpolynomial amount of space, Fortnow and Van Melkebeek [9] derived a time lower bound of $n^{\phi-o(1)}$, where $\phi \approx 1.618$ denotes the golden ratio. Recently, Williams [24] improved this lower bound to $n^{1.732}$, and we can further boost it to $n^{1.759}$ as a corollary to one of our results.

However, the main focus of this paper is not on lower bounds for deterministic machines, but rather on lower bounds for *randomized machines with two-sided error* (bounded away from 1/2). While it is conjectured that satisfiability requires exponential time even on such machines, proving lower bounds in the presence of randomness becomes a more difficult task than in the deterministic setting. No nontrivial lower bounds for satisfiability have been established on randomized random-access machines with two-sided error, even when the workspace of such machines is restricted to be logarithmic. In fact, previous to this work, no nontrivial time-space lower bounds have been shown for any complete problems in the polynomial-time hierarchy on randomized random-access machines with two-sided error.

1.1 Results

In this paper, we establish such time-space lower bounds. We consider the problem QSAT_ℓ of deciding the validity of a given quantified Boolean formula with at most $\ell-1$ quantifier alternations (beginning with existential). For any integer $\ell \geq 1$, QSAT_ℓ is complete for the ℓ^{th} level of the polynomial-time hierarchy. These problems are generalizations of satisfiability — for $\ell = 1$, QSAT_ℓ is precisely the satisfiability problem. Time-space lower bounds for QSAT_ℓ have been previously considered for deterministic machines. For example, Fortnow and Van Melkebeek show an $n^{\ell-o(1)}$ time lower bound for deterministic machines solving QSAT_ℓ in subpolynomial space, for $\ell \geq 2$. We match these bounds for randomized machines running in subpolynomial space, and a more careful analysis yields lower bounds for small polynomial space bounds:

Theorem 1 (Main Theorem). *For any integer $\ell \geq 2$ and constant $c < \ell$, there exists a positive constant d such that QSAT_ℓ cannot be solved by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above for $\ell = 2$, and d approaches 1 from below as c approaches 1 from above for $\ell \geq 3$.*

The randomized machines in Theorem 1 and in the rest of this paper refer to the natural coin flip model, in which the machine has one-way read-only access to a tape with random bits. Viola [23] recently extended Theorem 1 to the model in which the randomized machines have two-way access to the random bit tape, although his approach yields weaker lower bounds and only works for $\ell \geq 3$.

We note that Theorem 1 establishes the first polynomial-strength time-space lower bounds for problems in the polynomial-time hierarchy on two-sided error randomized machines. By time-space lower bounds of "polynomial strength" we mean time lower bounds of the form $\Omega(n^c)$ for some constant $c > 1$ under nontrivial space upper bounds. Previous works establish randomized time-space lower bounds but they either consider problems believed not to be in the polynomial-time hierarchy, or the time lower bounds involved are only slightly super-linear. Allender et al.'s [1] time-space lower bounds for problems in the counting hierarchy on probabilistic machines with unbounded error fall within the first category. On the other hand, Beame et al.'s [3] nonuniform time-space lower bound for a binary quadratic form problem in P falls within the second category.

Because of the tight connection between QSAT_ℓ and the ℓ^{th} level of the linear-time hierarchy, Theorem 1 can be stated equivalently as a time-space lower bound for simulations of the ℓ^{th} level of the linear-time hierarchy on randomized machines with two-sided error. In fact, we can strengthen Theorem 1 and establish time-space lower bounds for simulations of linear-time alternating machines which only use space n^a for constant $a \leq 1$.

Theorem 2. *For any integer $\ell \geq 2$ and any constants $a \leq 1$ and $c < 1 + (\ell - 1)a$, there exists a positive constant d such that linear-time alternating machines using space n^a and $\ell - 1$ alternations cannot be simulated by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $a/2$ from below as c approaches 1 from above for $\ell = 2$, and d approaches a from below as c approaches 1 from above for $\ell \geq 3$.*

Note that when $a = 1$, the space restriction on the alternating machines disappears and Theorem 2 becomes Theorem 1.

The $\ell \geq 2$ restriction in Theorem 1 implies that it does not give any bounds for the first level of the polynomial-time hierarchy, i.e., for satisfiability or its complement, tautology, the problem of deciding if a propositional formula is true under all assignments. However, we are able to strengthen the known lower bounds for tautology on randomized machines with *one-sided error*. Previously, Fortnow and Van Melkebeek showed a lower bound of $n^{\sqrt{2}-o(1)}$ for the tautology problem on nondeterministic machines using subpolynomial space. Since randomized machines with one-sided error (on the "yes" side) are special cases of nondeterministic machines, these bounds also apply to this setting. Using ideas from the proof of our main result, we manage to take advantage of the extra structure provided by the randomized machine and improve the known lower bounds for tautology on randomized machines with one-sided error.

Theorem 3. *There exists a positive constant d such that tautology cannot be solved by randomized random-access machines with one-sided error running in time $n^{1.759}$ and space n^d.*

Notice that the lower bound of Theorem 3 applies to deterministic machines as a special case. Therefore, by the closure of deterministic classes under complement, Theorem 3 implies the improved lower bound for satisfiability on deterministic machines mentioned earlier.

Corollary 4. *There exists a positive constant d such that satisfiability cannot be solved by deterministic random-access machines running in time $n^{1.759}$ and space n^d.*

1.2 Techniques

Our proofs follow the paradigm of indirect diagonalization. This technique establishes a desired separation by contradiction – assuming the separation does not hold, we derive a sequence of progressively unlikely inclusions of complexity classes until we reach one that contradicts a known diagonalization result. Kannan [14] used the paradigm avant la lettre to investigate the relationship between deterministic linear time and nondeterministic linear time. All of the recent work on time-space lower bounds for satisfiability and problems higher up in the polynomial-time hierarchy [8, 17, 9, 19, 24] follow it as well. Allender et al. [1] employed the technique to establish time-space lower bounds for problems in the counting hierarchy.

At first glance, it might seem that current techniques from space-bounded derandomization let us derive time-space lower bounds on randomized machines as immediate corollaries to time-space lower bounds on deterministic machines. In particular, assuming that we can solve satisfiability on a randomized machine in logarithmic space and time n^c, Nisan's deterministic simulation [21] yields a deterministic algorithm for satisfiability that runs in polylogarithmic space and polynomial time. However, even for $c = 1$, the degree of the latter polynomial is far too large for this simulation to yield a contradiction with known time-space lower bounds for deterministic machines. Thus, we need a more delicate approach for the randomized setting.

The critical ingredient in this approach is a time- and space-efficient simulation of randomized computations in the second level of the polynomial-time hierarchy with very few guess bits. The latter follows from a careful combination of Nisan's partial space-bounded derandomization [20], deterministic amplification by a random walk on an expander [5, 13], and a version of Lautemann's proof that randomized machines with two-sided error can be simulated by an alternating machine at a polynomial time-overhead [15]. It gives us (i) an unconditional way to speed up small-space randomized computations with two-sided error in higher levels of the polynomial-time hierarchy, and (ii) a conditional efficient complementation of computations within the ℓ^{th} level of the polynomial-time hierarchy for $\ell \geq 2$. The condition on the latter is the hypothesis of the indirect diagonalization argument that the ℓ^{th} level of the linear-time hierarchy can be simulated by randomized machines with two-sided error that run in time n^c and small space. Combining that hypothesis with (i) and (ii), we conclude that computations in the ℓ^{th} level of the polynomial-time hierarchy that run in time T (where T is some sufficiently large polynomial), can be complemented in time $g(T)$, where g is some function depending on c. For sufficiently small values of c, $g(T)$ becomes $o(T)$, which yields a contradiction with a known diagonalization result. For somewhat larger values of c, we do not obtain a contradiction right away but we obtain a more efficient complementation within the ℓ^{th} level of the polynomial-time hierarchy for larger polynomials T. We then run the argument again using the new efficient complementation in step (ii), yielding an even more efficient complementation. This allows us to rule out larger values of c, and further improve the efficiency of complementation. Bootstrapping this way leads to Theorem 1.

A careful analysis shows that we can handle space bounds of the form n^d, where d is a positive constant depending on c. For $\ell = 2$, the above argument yields a constant d approaching 1/2 from below when c approaches 1 from above. For $\ell \geq 3$, we achieve a better value of d for such small values of c by deriving a more efficient simulation of randomized computations in the *third* level of the polynomial-time hierarchy.[1] This follows by exploiting the structure of the second-level simulation described above and adding an alternation to reduce the time overhead. The savings in time are more substantial for large values of d. When c approaches 1 from above, the modified argument can handle values of d approaching 1 from below. For larger values of c, the cost of the additional alternation obviates the savings in running time and makes the earlier argument the better one. By paying close attention to the space used by the simulations involved, we obtain the strengthening given in Theorem 2.

For our tautology lower bounds, we extend Williams' recent lower bounds for deterministic machines [24] to nondeterministic machines with few guess bits, and add a new component to the argument. The proof in [24] contains two ingredients. The first one is a bootstrapping argument similar to the one we just described. Instead of yielding more and more efficient complementations within the ℓ^{th} level of the polynomial-time hierarchy for some fixed ℓ at larger and larger poly-

[1] Viola [23] independently obtained the same simulation using a somewhat more complicated argument.

nomial time bounds T, it gives more and more efficient complementations of linear time within higher and higher levels of the polynomial-time hierarchy. The second ingredient is an improved starting point for the bootstrapping argument. This involves using the hypothesis to obtain a — conditional — better speedup of small-space deterministic computations in the second level of the polynomial-time hierarchy. To obtain the quantitative improvement stated in Corollary 4, we take the idea of exploiting the hypothesis of the indirect diagonalization argument further and show how to improve the conditional speedup of small-space deterministic computations in higher levels of the polynomial-time hierarchy. In order to establish the lower bounds on randomized machines with one-sided error (as in Theorem 3), we show that the above argument extends from deterministic machines to nondeterministic machines that use few guess bits. We exploit Nisan's partial derandomization again, this time to transform one-sided error randomized machines into equivalent nondeterministic machines with few guess bits at a marginal cost in time and space.

1.3 Organization

Section 2 introduces the notation and machine models we use for this paper. Additionally, we state some useful complexity results which are fundamental to our techniques.

In Section 3, we describe the general framework of our proofs. This includes a tight connection between QSAT_ℓ and linear time on an alternating machine with $\ell - 1$ alternations, which allows us to focus on proving lower bounds for the latter from there on. We also describe the paradigm of indirect diagonalization which our lower bound proofs follow, and give a concrete example.

Section 4 shows how we can leverage Nisan's space-bounded derandomization, deterministic amplification, and Lautemann's proof that randomized machines with two-sided error can be simulated by alternating machines with one alternation at a polynomial time-overhead. Specifically, we obtain simulations of space-bounded randomized machines by alternating machines which use few guess bits and only marginally more time and space than the randomized machine. We exploit these simulations in Section 5 to establish the lower bounds given by Theorem 1.

Section 6 contains our other results. Specifically, we show the more general time-space lower bounds for space-bounded linear-time alternating machines given by Theorem 2, as well as the time-space lower bounds for tautology on randomized machines with one-sided error given by Theorem 3.

Finally, we conclude in Section 7 by discussing some open problems that remain directions for further research. We also include an appendix proving some results regarding the running time of Nisan's generator and of deterministic amplification based on a random walk on the Gabber-Galil expander. We could not find these results in the literature and they may be of independent interest.

2 Preliminaries

While much of the notation we use is standard [2, 22], we introduce some conventions and additional notation in this section. We also state a few results which we use throughout the rest of the paper.

2.1 Machine Model

Our lower bounds are robust with respect to the choice of machine model. In particular, they hold for random-access machines. We refer to [19] for the details of the specific model we use for our derivations.

We convene that the time and space bounds of these machines are constructible functions from natural numbers to natural numbers which are at least logarithmic, and refer to them as time and space functions. We often discuss "subpolynomial" space functions, which refer to space functions in $n^{o(1)}$. Our results ultimately apply to computations with polynomial time and space bounds which certainly meet the required constructibility conditions. Note that machines running in sublinear time or sublogarithmic space trivially cannot solve problems like satisfiability where the answer can depend on the entire input.

2.2 Notation

We introduce some additional terminology to reason about randomized computation. In particular, we use the notation BPTISP$[T, S]$ to refer to the class of languages recognized by randomized machines using time T and space S with error bounded by $\frac{1}{3}$ on both sides. Similarly, RTISP$[T, S]$ refers to randomized machines with one-sided error on the membership side bounded by $\frac{1}{2}$.

As is standard, we assume that the random bits are presented to such machines on a one-way read-only worktape. If the machine wishes to re-read random bits, it must copy them down on a worktape at the cost of space, as opposed to the more powerful model which has two-way access to the random tape. Except where stated otherwise, our results about randomized machines hold only for the former machine model.

Our arguments involve alternating computations in which the numbers of bits guessed at each stage are bounded by explicitly given small functions. To this end, we use the following notation to describe such computations:

Definition. *Given a complexity class $\mathcal{C}$ and a function f, we define the class $\exists^f \mathcal{C}$ to be the set of languages that can be described as*

$$\{x | \exists y \in \{0,1\}^{O(f(|x|))} P(x,y)\},$$

where P is a predicate accepting a language in the class $\mathcal{C}$ when its complexity is measured in terms of $|x|$ (not $|x| + |y|$). We analogously define $\forall^f \mathcal{C}$.

For example, $\exists^f$DTIME$[n]$ and $\forall^f$DTIME$[n]$ are subsets of NP and coNP for $f(n) = n^{O(1)}$. The requirement that the complexity of P be measured in terms of $|x|$ allows us to express the running times in terms of the original input length, which is a more natural convention for our arguments.

A subtlety arises when we consider space-bounded classes $\mathcal{C}$. Computations corresponding to $\exists^f \mathcal{C}$ and $\forall^f \mathcal{C}$ explicitly write down their guess bits y and then run a space-bounded machine on the combined input consisting of the original input x and the guess bits y. Thus, the space-bounded machine effectively has two-way access to the guess bits y. For example, although machines corresponding to $\exists^n$DTISP$[n, n^{o(1)}]$ and to NTISP$[n, n^{o(1)}]$ both use only a subpolynomial amount of space to verify their guesses, they do not necessarily have the same computational power. This is because the former machines have two-way access to the guess bits, which are written down on a separate tape that does not count towards its space bound, whereas the latter machines only have one-way access to these bits and do not have enough space to write them down on their work tape.

2.3 Speedup of Space-Bounded Computations

We also make use of the standard divide-and-conquer approach for speeding up space-bounded computations by introducing alternations. This technique is described in detail in [19]. By splitting up the computation tableau of a DTISP$[T, S]$ computation into $B \geq 1$ equal-sized blocks, we obtain

$$\mathrm{DTISP}[T, S] \subseteq \exists^{BS} \forall^{\log B} \mathrm{DTISP}[T/B, S] \subseteq \Sigma_2\mathrm{TIME}[BS + T/B]. \tag{1}$$

By the closure of DTISP under complement, this inclusion can also be stated for the Π-side of the polynomial time hierarchy, which will be more convenient to use in some of our arguments. If we choose B to optimize the running time of the resulting Π_2-computation, the result is a square-root speedup for small space bounds S:

$$\mathrm{DTISP}[T, S] \subseteq \forall^{\sqrt{TS}} \exists^{\log T} \mathrm{DTISP}[\sqrt{TS}, S] \subseteq \Pi_2\mathrm{TIME}[\sqrt{TS}]. \tag{2}$$

Recursively applying (1) while exploiting DTISP's closure under complementation to conserve alternations yields

$$\mathrm{DTISP}[T, S] \subseteq \underbrace{\forall^{BS} \exists^{BS} \ldots \overline{Q}^{BS}}_{k-1} \; Q^{\log B} \mathrm{DTISP}[T/B^{k-1} + BS, S] \subseteq \Pi_k\mathrm{TIME}[T/B^{k-1} + BS] \tag{3}$$

for any integer $k \geq 2$, where $Q = \exists$ if k is even and $Q = \forall$ otherwise, and $\overline{Q}$ denotes the quantifier complementary to Q. Choosing B to optimize the resulting running time, (3) achieves a k^{th}-root speedup for small space bounds S.

$$\begin{aligned} \mathrm{DTISP}[T, S] &\subseteq \underbrace{\forall^{(TS^{k-1})^{1/k}} \exists^{(TS^{k-1})^{1/k}} \ldots \overline{Q}^{(TS^{k-1})^{1/k}}}_{k-1} \; Q^{\log(T/S)} \mathrm{DTISP}[(TS^{k-1})^{1/k}, S] \\ &\subseteq \Pi_k\mathrm{TIME}[(TS^{k-1})^{1/k}]. \end{aligned} \tag{4}$$

2.4 Diagonalization Results

Finally, we need a standard diagonalization result from which we can derive contradictions. The following lemma states that we cannot speed up the computation of every language in $\Sigma_\ell\mathrm{TIME}[T]$ by switching to Π_ℓ.

Lemma 5 (Folklore). *Let ℓ be a positive integer and T a time function. Then*

$$\Sigma_\ell\mathrm{TIME}[T] \not\subseteq \Pi_\ell\mathrm{TIME}[o(T)].$$

Our lower bounds for space-bounded alternating linear time require a stronger diagonalization result which is both time- and space-sensitive.

Lemma 6 ([9]). *Let T be a time function and S a space function. Then for any integer $\ell > 0$,*

$$\Sigma_\ell\mathrm{TISP}[T, S] \not\subseteq \Pi_\ell\mathrm{TISP}[o(T), o(S)].$$

3 Earlier Techniques

We now outline some of the techniques common to many time-space lower bound arguments. We also use them for our results.

3.1 Alternating Linear Time Versus QSAT_ℓ

The first such result involves a tight connection between QSAT_ℓ and linear time on an alternating Turing machine with ℓ alternating stages, $\Sigma_\ell\mathrm{TIME}[n]$. At the first level, we know that satisfiability can be solved in nondeterministic quasi-linear time, so that time-space lower bounds for satisfiability imply the same lower bounds for $\mathrm{NTIME}[n]$ up to polylogarithmic factors. Conversely, Fortnow and Van Melkebeek [9] show that a sufficient strengthening of the Cook-Levin Theorem gives a reduction from $\mathrm{NTIME}[n]$ to satisfiability which is efficient in both time *and* space, showing that if satisfiability can be solved in time n^c and space n^d, then $\mathrm{NTIME}[n]$ can be solved in time $n^c \operatorname{polylog}(n)$ and space $n^d \operatorname{polylog}(n)$. Thus, time-space lower bounds for $\mathrm{NTIME}[n]$ and for satisfiability are equivalent up to polylogarithmic factors.

At higher levels of the polynomial-time hierarchy, we know that QSAT_ℓ can be solved in quasi-linear time on a machine which makes $\ell-1$ alternations, so that time-space lower bounds for QSAT_ℓ imply the same lower bounds for $\Sigma_\ell\mathrm{TIME}[n]$ up to polylogarithmic factors. Conversely, just as the Cook-Levin theorem generalizes from NP to higher levels of the polynomial-time hierarchy and shows that QSAT_ℓ is complete for Σ_ℓ^p, the reductions given by Fortnow and Van Melkebeek [9] generalize to give time- and space-efficient reductions from $\Sigma_\ell\mathrm{TIME}[n]$ to QSAT_ℓ:

Theorem 7. *For any integer $\ell \geq 1$ and constants $c, d > 0$, if*

$$\mathrm{QSAT}_\ell \in \mathrm{DTISP}[n^c, n^d],$$

then

$$\Sigma_\ell\mathrm{TIME}[n] \subseteq \mathrm{DTISP}[n^c \operatorname{polylog}(n), n^d \operatorname{polylog}(n)].$$

This also holds if we replace DTISP *by* BPTISP *or* RTISP*.*

This establishes the equivalence of time-space lower bounds for $\Sigma_\ell\mathrm{TIME}[n]$ and QSAT_ℓ up to polylogarithmic factors. In particular, polynomial-strength time-space lower bounds for $\Sigma_\ell\mathrm{TIME}[n]$ on randomized machines yield essentially the same lower bounds for QSAT_ℓ. With this in mind, we focus on proving lower bounds for $\Sigma_\ell\mathrm{TIME}[n]$ for the rest of the paper. We include a proof of Theorem 7 here for completeness.

Proof. Let L be a language decided by a random-access linear-time alternating Turing machine which makes $\ell - 1$ alternations, beginning in an existential stage. For such an L, let P be the predicate recognized by a random-access linear-time nondeterministic machine so that the condition $x \in L$ can be written as

$$(\exists y_1 \in \{0,1\}^{rn})(\forall y_2 \in \{0,1\}^{rn}) \dots (Q y_{\ell-1} \in \{0,1\}^{rn}) R(x, y_1, y_2, \dots, y_{\ell-1}),$$

where r is some constant, and $Q = \forall$, $R = P$ if ℓ is odd, and $Q = \exists$, $R = \neg P$ otherwise. All that remains to represent the acceptance condition of L as a quantified Boolean formula is to reduce P to satisfiability. The original Cook-Levin reduction produces a formula of quadratic size, which is too large for our purposes. However, Cook [6] shows how to leverage the oblivious simulations of Hennie and Stearns [12] to obtain a formula of quasilinear size. More precisely, we can construct a formula φ depending only on P and n such that φ:

- has size $O(n \operatorname{polylog}(n))$ where each bit can be constructed in time $O(\operatorname{polylog}(n))$ and space $O(\log(n))$,

- involves the bits of $x, y_1, \ldots, y_{\ell-1}$ input to P as well as $O(n \log n)$ additional Boolean variables z, and
- is satisfiable in z on input $x, y_1, \ldots, y_{\ell-1}$ if and only if P accepts $x, y_1, \ldots, y_{\ell-1}$.

Defining $\varphi' \doteq \varphi$ if ℓ is odd and $\varphi' \doteq \neg\varphi$ otherwise, this shows that the Σ_ℓ-formula

$$\psi \doteq \exists y_1 \forall y_2 \ldots Q y_{\ell-1} \overline{Q} z \, \varphi',$$

is in QSAT_ℓ if and only if $x \in L$. The size of ψ is only $O(n \log n)$ more than φ — the log factor is required to write down the bit indices of the quantified variables. The easy nature of these extensions to φ endows ψ with the same constructibility properties as φ.

In the case that $\mathrm{QSAT}_\ell \in \mathrm{DTISP}[n^c, n^d]$, let M be a deterministic machine deciding QSAT_ℓ in time $O(n^c)$ and space $O(n^d)$. We simulate M on input ψ to decide L. However, computing ψ and writing down the result on a worktape requires too much space. Instead, when M needs a bit of ψ, the simulation computes this bit from scratch. As φ is of size $O(n \,\mathrm{polylog}(n))$, M runs for time $O(n^c \,\mathrm{polylog}(n))$ and space $O(n^d \,\mathrm{polylog}(n))$ on input ψ. Computing the bits of ψ on the fly adds a multiplicative overhead of $O(\mathrm{polylog}(n))$ to the time and an additive overhead of $O(\log n)$ to the space. Thus, this deterministic simulation decides L in the desired time and space bounds.

The cases for BPTISP and RTISP follow from the same argument. $\square$

3.2 Indirect Diagonalization

We set out to prove Theorem 1, i.e., that $\Sigma_\ell\mathrm{TIME}[n] \not\subseteq \mathrm{BPTISP}[t, s]$ for certain interesting values of t and s. To accomplish this, we follow the same general technique that is used to prove all previous time-space lower bounds for nondeterministic linear time, namely indirect diagonalization. This paradigm can be described as following three basic steps:

1. Assume the inclusion that we wish to show does not hold. In our case, assume that $\Sigma_\ell\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[t, s]$.
2. Using the hypothesis, derive inclusions of complexity classes which are increasingly unlikely.
3. Eventually, one of these inclusions contradicts a known diagonalization result, proving the desired result.

There is a myriad of ways to derive new inclusions from the hypothesis in step 2, with different approaches yielding different results. Often, the inclusions derived in step 2 are obtained by a combination of two opposing processes. These can be loosely thought of as deriving a speedup at the cost of adding alternations, and removing alternations at the cost of a small slowdown. The main intuition that guides how we apply these processes is that we wish the speedup gained by introducing alternations to outweigh the cost of eliminating them, so that overall we obtain a contradiction to Lemma 5 (or Lemma 6).

The former process, deriving a speedup, involves simulating a space-bounded machine of the type for which we are attempting to derive a lower bound (i.e., deterministic or randomized) in significantly less time. We must pay a price to achieve this task, and we choose to pay in the currency of alternations. Specifically, we introduce a small number of alternations to the computation and carry out a fast simulation on an alternating machine. In the deterministic setting, such a simulation yields an inclusion resembling

$$\mathrm{DTISP}[T,S] \subseteq \Pi_\ell\mathrm{TIME}[f(T,S)],$$

for $\ell \geq 1$ and $f(T,S) \ll T$ when $S \ll T$. Note that the Π_2-simulation of a DTISP-computation given by (2) is an example of this process.

In the other direction, the process of removing alternations involves the task of simulating an alternating machine by another alternating machine which makes fewer alternations and runs for only slightly more time. In many cases, this is accomplished by deriving a statement such as

$$\Sigma_\ell\mathrm{TIME}[T] \subseteq \Pi_\ell\mathrm{TIME}[g(T)],$$

for a small function g. When g is polynomial, such an inclusion results in a collapse of the polynomial-time hierarchy to the ℓ^{th} level, and we refer to it as an *efficient complementation*. Note that a complementation can be derived unconditionally by using exhaustive search in lieu of an alternating step, but in this case g will be exponential in T. In our arguments, we can derive an efficient complementation which is conditional on the hypothesis of step 1, from which more efficient complementations are derived in an inductive fashion in step 2.

Notice that by Lemma 5, a complementation with $g(T) = o(T)$ is not just unlikely but impossible. Deriving such an impossibly efficient complementation provides the desired inclusion to arrive at a contradiction in step 3. In the following section, we give an example of how to accomplish this end via an appropriate combination of a speedup and an efficient complementation.

3.3 A Concrete Example

We step through an instantiation of the indirect diagonalization paradigm and prove the result of [17] that satisfiability cannot be solved by deterministic random-access machines running in time n^c and space $n^{o(1)}$ for constants $c < \sqrt{2}$. The first step is to assume that

$$\mathrm{NTIME}[n] \subseteq \mathrm{DTISP}[n^c, n^{o(1)}]. \tag{5}$$

This allows a simulation of NTIME$[T]$ by DTISP$[T^c, T^{o(1)}]$ for some polynomial T. The speedup given by the inclusion (2) then yields a square-root speedup at the cost of two alternations. The net result is

$$\mathrm{NTIME}[T] \subseteq \mathrm{DTISP}[T^c, T^{o(1)}] \subseteq \Pi_2\mathrm{TIME}[T^{c/2+o(1)}], \tag{6}$$

which represents a speedup of NTIME$[T]$ for $c < 2$ by using one more alternating stage. To contradict Lemma 5, we need to arrive at such a speedup which uses just a universal stage. Therefore, we use an efficient complementation to remove the inner existential stage from the Π_2-speedup represented by (6). Note that the hypothesis (5) gives a simulation of NTIME$[n]$ by a deterministic machine, and since any deterministic machine is trivially a conondeterministic machine, we have

$$\mathrm{NTIME}[n] \subseteq \mathrm{coNTIME}[n^c]. \tag{7}$$

In order to use this efficient complementation to remove an alternation, we write

$$\Pi_2\mathrm{TIME}[T^{c/2+o(1)}] = \forall^{T^{c/2+o(1)}} \underbrace{\mathrm{NTIME}[T^{c/2+o(1)}]}_{(\alpha)}.$$

Note that (α) represents a nondeterministic computation which takes an input of size $n + T^{c/2+o(1)}$ and runs in time $T^{c/2+o(1)}$. For $T(n) \geq n^{2/c}$, the running time is at least linear in the input size, so that (7) gives a conondeterministic simulation of (α) running in time $T^{c^2/2+o(1)}$. By merging the resulting adjacent universal stages, we obtain

$$\mathrm{NTIME}[T] \subseteq \forall^{T^{c/2+o(1)}}\mathrm{coNTIME}[T^{c^2/2+o(1)}] = \mathrm{coNTIME}[T^{c^2/2+o(1)}].$$

When $c^2/2 < 1$, we can see this process has delivered a net speedup of nondeterministic time T on conondeterministic machines, which is a contradiction to Lemma 5. This proves the desired result.

4 Lautemann's Proof and Derandomization

We now adopt the techniques of the previous section to prove Theorem 1, beginning with the case $\ell = 2$. By way of Theorem 7 and the indirect diagonalization paradigm, we seek to derive a contradiction from the assumption

$$\Sigma_2\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[t, s] \tag{8}$$

for some interesting functions t and s. The known proofs that BPP lies in the second level of the polynomial-time hierarchy provide a simulation of randomized machines with two-sided error by Π_2-machines with a polynomial overhead in time. Combined with hypothesis (8), this immediately gives us one of the ingredients we need to carry through the program from the previous section, namely the efficient complementation of $\Sigma_2\mathrm{TIME}[n]$.

Assuming the Π_2-simulation is sufficiently time- and space-efficient, we can also use it for the other ingredient we need, namely the speedup. This is because the divide-and-conquer strategy for DTISP-computations from Section 2.3 applies to Σ_kTISP-computations as well. However, it turns out that the known Π_2-simulations of randomized two-sided error machines are not time- and space-efficient enough to obtain any lower bounds this way. Moreover, in order to achieve the quantitative strength of our lower bounds, we need to save alternations by applying the speedup as in (1) to the final deterministic phase of the simulation as opposed to the Π_2-simulations as a whole. For that approach to yield an overall speedup, we need the number of guess bits in the alternating phases of the simulation to be small — otherwise, the time needed for the guesses would obviate the speedup obtained in the final deterministic phase. The known simulations use too many guess bits from that perspective. Therefore, in this section, we develop a new Π_2-simulation of randomized machines with two-sided error that meets all the above efficiency requirements.

We start by analyzing Lautemann's proof that any language L in BPP is also in $\Sigma_2^p \cap \Pi_2^p$. The proof assumes a randomized algorithm using R random bits to decide L with error ϵ. When ϵ is small enough in comparison to R, there is a $v \geq 1$ so that membership of x in L can be characterized by the existence of v shifts of the set of random strings accepting x which together cover the universe of all random strings. If $x \in L$, the set of random strings accepting x is large enough to guarantee that such shifts exist as long as $\epsilon^v < 2^{-R}$. On the other hand, if $x \notin L$, the set of accepting random strings is small enough so that v shifts cannot cover the universe of random strings as long as $\epsilon < \frac{1}{v}$. For such ϵ and v, these complementary conditions are expressed by a Σ_2^p predicate. Since BPP is closed under complement, this shows that BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$. Specifically, we are interested in the Π_2^p-side of the inclusion.

Theorem 8 (Lautemann [15]). *Let L be a language recognized by a randomized machine M that runs in time T, space S, and uses R random bits with error bounded on both sides by ϵ. Then for any $v \geq 1$ such that $\epsilon < \min(2^{-R/v}, \frac{1}{v})$, we have that*

$$L \in \forall^{vR}\exists^{R}\mathrm{DTISP}[vT, S + \log v]. \tag{9}$$

In Lautemann's proof that $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$, one starts from an algorithm deciding $L \in \mathrm{BPP}$ that has error less than the reciprocal of the number R' of random bits it uses. Such an error probability can be achieved from a standard BPP algorithm deciding L that uses R random bits by taking the majority vote of $O(\log R)$ independent trials, which results in $R' = O(R \log R)$. For $\epsilon < \frac{1}{R'}$, choosing $v = R'$ satisfies the conditions of Lemma 8. This shows that if L can be decided by a $\mathrm{BPTISP}[T, S]$ machine using R random bits, then

$$L \in \forall^{(R\log R)^2}\exists^{R\log R}\mathrm{DTISP}[RT\log R, S]. \tag{10}$$

We can further reduce the number of guess bits in the alternating stages of the simulation by using more efficient methods of amplification. With such methods, the error can be made as small as 2^{-R} while using only $O(R)$ random bits. Specifically, the algorithm runs $O(R)$ trials which are obtained from the labels of vertices on a random walk of length $O(R)$ in an easily constructible expander graph, and accepts if a majority of these trials accept. For our purposes, we choose the Gabber-Galil family of expanders [10], a construction based on the Margulis family [18] where the vertices are connected via simple affine transformations on the labels. The easy form of the edge relations ensures that the walk is efficiently computable in time $O(R^2)$ and space $O(R)$.

Theorem 9 ([5, 13]). *Let M be a randomized machine with constant error bounded away from $\frac{1}{2}$ that runs in time T, space S, and uses R random bits. Then M can be simulated by another randomized machine M' that runs in time $O(RT)$ and space $O(R + S)$, while using only $O(R)$ random bits to achieve error 2^{-R}.*

When an algorithm has been amplified as in Theorem 9, $v = O(1)$ shifts suffice for Theorem 8. This shows that if L can be decided by a $\mathrm{BPTISP}[T, S]$ machine using R random bits, then

$$L \in \forall^{R}\exists^{R}\mathrm{DTISP}[RT, R + S]. \tag{11}$$

The efficiency of this simulation depends on the number of random bits R for all the criteria we mentioned: the number of bits guessed in the alternating stages, the multiplicative time overhead, and the additive space overhead for the final deterministic stage are all $O(R)$. Since R can be as large as T, we need an additional ingredient to do better. That ingredient exploits the fact that we are dealing with *space-bounded* BPP-computations. In that setting, we know of techniques to reduce the number of random bits without increasing the time or space by much, which in turn increases the efficiency of the Π_2-simulation in (11). The means by which we achieve the needed reduction in randomness is the space-bounded derandomization of Nisan [20]. We state a version here, and leave the proof to the Appendix (where we also present a brief overview of Theorem 9).

Theorem 10. *Any randomized machine M running in time T and space S with error ϵ can be simulated by another randomized machine running in time $O(T\,\mathrm{polylog}(T))$, space $O(S\log T)$, and using only $O(S\log T)$ random bits. The error of the simulation is $\epsilon + 2^{-S}$, and is one-sided if M has one-sided error.*

Note that we do not apply Theorem 10 to deterministically simulate the randomized machine. Instead, we use it to reduce the randomness required by a BPTISP$[T, S]$ machine to $O(S \log T)$. If we subsequently efficiently amplify using Theorem 9, then the overhead of the alternating simulation given by Theorem 8 becomes acceptable for polynomial T and small S. More precisely, we have:

Theorem 11.

$$\mathrm{BPTISP}[T, S] \subseteq \forall^{S \log T} \exists^{S \log T} \mathrm{DTISP}[TS \operatorname{polylog}(T), S \log T]. \tag{12}$$

Proof. Let M be the randomized time T, space S machine for recognizing $L \in \mathrm{BPTISP}[T, S]$. By the derandomization of Theorem 10, we obtain a simulation using $R \doteq O(S \log T)$ random bits, time $O(T \operatorname{polylog}(T))$, and space $O(S \log T)$, with error $\frac{1}{3} + 2^{-S}$. Theorem 9 gives a machine deciding L with error 2^{-R} while using $O(R)$ random bits. The time increases to $O(TS \operatorname{polylog}(T))$, while the space is still $O(S \log T)$. Applying Theorem 8 for $v = O(1)$ yields the desired alternating simulation of M. $\square$

We point out that using the instantiation of Theorem 8 given by (10) instead of (11) in the proof of Theorem 11 yields a simulation similar to (12). The main difference is that the initial universal phase takes time $O((S \log T)^2)$ rather than $O(S \log T)$. This version is still efficient enough to yield time-space lower bounds as in Theorem 1, but the dependence of the space parameter d on c becomes worse.

4.1 Speedup

Theorem 11 has the desired nice properties that allow us to derive a speedup for BPTISP. The simulation spends only time $O(S \log T)$ in its alternating phases, which is small when S is small. In this case, the running time is dominated by the final deterministic computation, so that a speedup of this final stage results in a speedup of the computation as a whole. Since the final deterministic computation of the simulation given by (12) is space-bounded, we can achieve this by applying the speedup of (1) or (3). For example, applying (1) adds two alternations, and by merging adjacent existential stages we obtain a simulation given by:

$$\begin{aligned} \mathrm{BPTISP}[T, S] &\subseteq \forall^{S \log T} \exists^{S \log T} \exists^{BS \log T} \forall^{\log B} \mathrm{DTISP}[TS \operatorname{polylog}(T)/B, S \log T] \\ &= \forall^{S \log T} \exists^{BS \log T} \forall^{\log B} \mathrm{DTISP}[TS \operatorname{polylog}(T)/B, S \log T]. \end{aligned}$$

Choosing B to optimize the running time of this simulation up to a polylog(T) factor, we get

$$\mathrm{BPTISP}[T, S] \subseteq \forall^{S \log T} \exists^{\sqrt{T} S} \forall^{\log T} \mathrm{DTISP}[\sqrt{T} S \operatorname{polylog}(T), S \log T]. \tag{13}$$

For small S, we obtain a speedup for space-bounded randomized machines similar to that which (2) gives for deterministic machines. However, the simulation uses three alternations rather than two to realize the same speedup by a square root.

4.2 Complementation

We also use Theorem 11 to derive an efficient complementation under the hypothesis of the indirect diagonalization argument. Note that the hypothesis (8) gives a simulation of Σ_2 by BPTISP, which can be simulated in turn by the Π_2-machine given by Theorem 11. Thus, we derive

$$\Sigma_2\mathrm{TIME}[n] \subseteq \forall^{s\log t}\exists^{s\log t}\mathrm{DTISP}[ts\,\mathrm{polylog}(t), s\log t], \tag{14}$$

which gives the desired complementation for small enough t and s. For example, when t is polynomial, say $t = n^c$ and s is subpolynomial, (14) becomes

$$\Sigma_2\mathrm{TIME}[n] \subseteq \forall^{n^{o(1)}}\exists^{n^{o(1)}}\mathrm{DTISP}[n^{c+o(1)}, n^{o(1)}] \subseteq \Pi_2\mathrm{TIME}[n^{c+o(1)}]. \tag{15}$$

Thus, we can see that when s is small, this complementation allows us to eliminate alternations at a cost little more than raising the running time to the power of c. In this manner, (15) can be used analogously to the complementation (7) in the deterministic case of Section 3.3.

We note that the bottleneck causing our lower bounds for QSAT_ℓ to hold only for $\ell \geq 2$ arises right here. In the case $\ell = 1$, the hypothesis becomes $\mathrm{NTIME}[n] \subseteq \mathrm{BPTISP}[t, s]$; combining with Theorem 11 as above, we obtain $\mathrm{NTIME}[n] \subseteq \Pi_2\mathrm{TIME}[ts\,\mathrm{polylog}(t)]$, which is trivial and does not represent an efficient complementation

4.3 Higher Levels of the Hierarchy

The previous discussion in this section developed techniques towards proving lower bounds for $\Sigma_2\mathrm{TIME}[n]$. These readily generalize to higher levels, where the hypothesis (8) becomes

$$\Sigma_\ell\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[t, s] \tag{16}$$

for $\ell \geq 3$. Theorem 11 then allows for an efficient simulation of $\Sigma_\ell\mathrm{TIME}[n]$ by a Π_2-machine, which can be used to eliminate alternations. This allows us to establish Theorem 1 for values of $c < \ell$, where d approaches some small value depending on ℓ from below as c approaches 1 from above.

For $\ell \geq 3$, we can get a better dependence of d on c when c approaches 1. In this setting, a Π_3-simulation of BPTISP suffices to achieve an efficient complementation. The ability to use an additional alternation allows us to achieve a more time-efficient simulation than the one given by Theorem 11. Specifically, we add an alternation to the latter Π_2-simulation and eliminate the time blowup incurred by running the $O(S\log T)$ trials required by the amplification of Theorem 9. Rather than deterministically simulate all of these trials, we use the power of alternation to efficiently verify that a majority of these trials accept.

Lemma 12. *Let M be a randomized machine with constant error bounded away from $\frac{1}{2}$ that runs in time T, space S, and uses R random bits. Then M can be simulated by another randomized machine M' that uses $O(R)$ random bits to achieve error 2^{-R}. Furthermore, the acceptance of M' on input x and random string r can be decided in*

$$\exists^R\forall^{\log R}\mathrm{DTISP}[T + R\,\mathrm{polylog}(R), R + S].$$

We defer the proof to the Appendix. Notice that the final deterministic stage of the simulation represented by (12) can be replaced by the Σ_2-verification given by Lemma 12. Merging the resulting adjacent existential phases results in a simulation using one more alternation but running in less time.

Theorem 13.

$$\mathrm{BPTISP}[T,S] \subseteq \forall^{S\log T}\exists^{S\log T}\forall^{\log S}\mathrm{DTISP}[T\,\mathrm{polylog}(T), S\log T]. \tag{17}$$

As in Sections 4.1 and 4.2, Theorem 13 admits a speedup of BPTISP to Π_4 as well as an efficient complementation of Σ_3. When $t = n^c$ and $s = n^d$ in (16), the latter eliminates alternations essentially at the cost of raising the running time to the power of c. For values of c close to 1, this cost is small enough to alleviate the effects of the extra alternation in (17). In this case, the better dependence of the running time of the simulation on the space parameter allows us to derive contradictions for larger values of d than we can by using Theorem 11. On the other hand, for larger values of c, the extra alternation in (17) has a greater impact, and eventually prevents us from reaching a contradiction. In this case, switching to the more alternation-efficient simulation given by Theorem 11 allows us to derive a contradiction for such larger values of c. However, we must restrict d to smaller values in order to counteract the worse dependence of (12) on the space bound. Therefore, to derive Theorem 1, we focus on using Theorem 11 to obtain the bounds for large values of c first, and then show how Theorem 13 yields the larger values of d when c is small.

5 Main Result

We now use the techniques discussed in the previous sections to formulate an indirect diagonalization argument for the case $\ell = 2$ of Theorem 1. For clarity, we present the following exposition in terms of subpolynomial space bounds, and generalize these techniques to polynomial space bounds in the subsequent formal proof. Thus, to obtain a lower bound for $\Sigma_2\mathrm{TIME}[n]$, we start with the assumption (8) for $t = n^c$ and $s = n^{o(1)}$, i.e.,

$$\Sigma_2\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[n^c, n^{o(1)}].$$

Consider a $\Sigma_2\mathrm{TIME}[T]$ computation for some time function $T(n) = n^{O(1)}$. We adopt the approach outlined in Section 3.3, namely speeding up $\Sigma_2\mathrm{TIME}[T]$ at the cost of adding alternations and then removing these alternations via an efficient complementation to arrive at a $\Pi_2\mathrm{TIME}[o(T)]$ computation. The hypothesis gives a simulation of $\Sigma_2\mathrm{TIME}[T]$ in $\mathrm{BPTISP}[T^c, T^{o(1)}]$. We then apply the square-root speedup of (13) to obtain a simulation in Π_3.

$$\begin{aligned}
\Sigma_2\mathrm{TIME}[T] &\subseteq \mathrm{BPTISP}[T^c, T^{o(1)}] \\
&\subseteq \forall^{T^{o(1)}}\underbrace{\exists^{T^{\frac{c}{2}+o(1)}}\forall^{\log T}\mathrm{DTISP}[T^{c/2+o(1)}, T^{o(1)}]}_{(\alpha)}.
\end{aligned}$$

We have arrived at a simulation which makes one more alternation than we started with. To balance the number of alternations, we eliminate one alternation. Notice that the stages of the computation indicated by (α) can be seen as a computation in Σ_2 taking input of size $n + T^{o(1)}$ and running in time $T^{\frac{c}{2}+o(1)}$. When T is large enough, this running time is at least linear in the input size, and we can pad (15) to allow us to switch (α) to Π_2. Merging the resulting adjacent universal stages yields the desired Π_2-simulation:

$$\Sigma_2\mathrm{TIME}[T] \subseteq \Pi_2\mathrm{TIME}[T^{\frac{c^2}{2}+o(1)}]. \tag{18}$$

For $c < \sqrt{2}$, this results in a net speedup which is a contradiction to Lemma 5. Thus we have derived a lower bound of $n^{\sqrt{2}-o(1)}$ for QSAT_2 on subpolynomial-space randomized machines. However, we can do better by observing what (18) represents for values of c that do not immediately contradict Lemma 5. Specifically, (18) gives a complementation of Σ_2 of the same form as (15), but has exponent cost of $c^2/2$ rather than c. Thus, we have derived a *more efficient* complementation for sufficiently large polynomial time bounds T when $c < 2$.

We now reiterate the above argument, except we use (18) to eliminate alternations more efficiently than we did with (15). This yields an even more efficient complementation for sufficiently large polynomials T:

$$\Sigma_2\mathrm{TIME}[T] \subseteq \Pi_2\mathrm{TIME}[T^{\frac{c^3}{4}+o(1)}].$$

This can in turn be used to derive another more efficient complementation, and so on. In this manner, we can derive a series of complementations, each one more efficient than the previous one. Specifically, each iteration multiplies the exponent cost of the complementation by $\frac{c}{2}$, so after k iterations we obtain

$$\Sigma_2\mathrm{TIME}[T] \subseteq \Pi_2\mathrm{TIME}[T^{c\cdot e_k+o(1)}],$$

where $e_k = (\frac{c}{2})^k$. Note that for $c < 2$, $e_k \to 0$ as $k \to \infty$. Thus, by choosing k large enough so that $c \cdot e_k < 1$, we arrive at a contradiction to Lemma 5, which proves the desired lower bound of $n^{2-o(1)}$.

The following lemma precisely derives the series of complementations of Σ_2 for larger space bounds than $n^{o(1)}$. Specifically, we consider the hypothesis (8) for polynomial t and s, namely $t = n^c$ and $s = n^d$ for some constants $c \geq 1$ and $d > 0$, and derive the running time of the resulting Π_2-simulation in terms c, d, and k, the number of times the argument is recursively applied.

Lemma 14. *Suppose that*

$$\Sigma_2\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[n^c, n^d] \tag{19}$$

for some constants $c \geq 1$ and $d > 0$ where $c + 2d \leq 2$. Then for any time function T and integer $k \geq 0$ such that $d \leq f_k$,

$$\Sigma_2\mathrm{TIME}[T] \subseteq \Pi_2\mathrm{TIME}\left[\left((T^{f_k} + n)^{c+d}\right) \mathrm{polylog}(T+n)\right],$$

where

$$f_k \;=\; (\tfrac{c+2d}{2})^k. \tag{20}$$

Proof. We give a proof by induction on k. For $k = 0$, a padded version of the initial complementation given by (14) offers a Π_2-simulation of $\Sigma_2\mathrm{TIME}[T]$ running in the desired time.

We now show the inductive step by using the k^{th} complementation to derive the $(k+1)^{\mathrm{st}}$. Padding the hypothesis (19) gives a simulation of $\Sigma_2\mathrm{TIME}[T]$ in $\mathrm{BPTISP}[(T+n)^c, (T+n)^d]$. Note that the addition of the term n to T ensures the validity of this step for arbitrary T, in particular for sublinear T. Applying (13) gives a speedup of the BPTISP simulation at the cost of three alternations, yielding

$$\Sigma_2\mathrm{TIME}[T] \subseteq \forall^{(T+n)^d \log(T+n)} \underbrace{\Sigma_2\mathrm{TIME}\left[(T+n)^{\frac{c+2d}{2}} \mathrm{polylog}(T+n)\right]}_{(\alpha)}. \tag{21}$$

The complementation given by the inductive hypothesis switches (α) to a Π_2-computation, eliminating one alternation. Specifically, (α) represents a Σ_2-machine running in time

$$\tilde{T} \doteq O\left((T+n)^{\frac{c+2d}{2}} \operatorname{polylog}(T+n)\right)$$

on inputs comprised of the original n-bit input in addition to the $(T+n)^d \log(T+n)$ bits guessed in the preceeding universal stage, for a total input size of

$$\tilde{n} \doteq O\left(n + (T+n)^d \log(T+n)\right).$$

The inductive hypothesis allows us to simulate (α) by a Π_2-machine running in time $O\left((\tilde{T}^{f_k} + \tilde{n})^{c+d} \operatorname{polylog}(\tilde{T} + \tilde{n})\right)$. Using this Π_2-machine for (α) and accounting for the time spent in the initial universal stage of the simulation given by (21) results in a Π_2 simulation of $\Sigma_2\mathrm{TIME}[T]$ running in time big-O of

$$(T+n)^d \log(T+n) + \left((\tilde{T}^{f_k} + \tilde{n})^{c+d}\right) \operatorname{polylog}(\tilde{T} + \tilde{n}).$$

All that remains is to show that the above running time is of the desired form under the conditions on c and d. To simplify this expression, note that $\operatorname{polylog}(\tilde{T} + \tilde{n}) = \operatorname{polylog}(T+n)$. Collecting all of the other polylog terms, the running time can be written as big-O of

$$\left[(T+n)^d + \left(\left((T+n)^{\frac{c+2d}{2}}\right)^{f_k} + n + (T+n)^d\right)^{c+d}\right] \operatorname{polylog}(T+n).$$

The terms depending on T in the big-O expression have exponents d, $\frac{c+2d}{2} f_k(c+d)$, and $d(c+d)$ (putting aside the $\operatorname{polylog}(T+n)$ factor for a moment). Thus, when $d \leq \frac{c+2d}{2} f_k = f_{k+1}$ and $c + d \geq 1$, the dominating term is $T^{\frac{c+2d}{2} f_k(c+d)} = T^{f_{k+1}(c+d)}$.

Similarly, the terms depending on n have exponents d, $\frac{c+2d}{2} f_k(c+d)$, $c+d$, and $d(c+d)$. Under the same conditions on c and d, the first and last terms are subsumed by the second one, which can be rewritten as $f_{k+1}(c+d)$. Additionally, when $c + 2d \leq 2$, $f_k \leq 1$ for all $k \geq 0$, so that the n^{c+d} term dominates. Thus, we simplify the running time to big-O of

$$\left((T^{f_{k+1}} + n)^{c+d}\right) \operatorname{polylog}(T+n),$$

which is of the desired form. □

The series of complementations given by Lemma 14 lead to a contradiction with Lemma 5 for certain values of c and d, which proves the desired lower bound.

Theorem 15. *For any constant $c < 2$, there exists a positive constant d such that $\Sigma_2\mathrm{TIME}[n]$ cannot be simulated by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above.*

Proof. For $c < 1$, the Theorem holds for any d by standard techniques. Namely, take the parity function, which is surely in $\Sigma_2\mathrm{TIME}[n]$ and consider any $\mathrm{BPTIME}[n^c]$ machine M which purportedly computes it. On input 0^n, we must have that

$$\sum_{i=1}^{n} \Pr[M \text{ looks at } i^{th} \text{ bit on input } 0^n] \leq n^c.$$

For $c < 1$, this implies that there is a bit position that M looks at very rarely. Namely, there exists an i such that

$$\Pr[M \text{ looks at } i^{th} \text{ bit on input } 0^n] = o(1).$$

From this, we can deduce that M has approximately the same probability of accepting 0^n as it does 0^n with the i^{th} bit flipped. Therefore, M cannot compute parity.

We prove the case for $c \geq 1$ via indirect diagonalization. Suppose, by way of contradiction, that

$$\Sigma_2\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[n^c, n^d], \tag{22}$$

for some constant $d > 0$ to be determined later. Then for any time function $\tau(n)$, Lemma 14 gives us the complementations

$$\Sigma_2\mathrm{TIME}[\tau] \subseteq \Pi_2\mathrm{TIME}\left[\left((\tau^{f_k} + n)^{c+d}\right) \mathrm{polylog}(\tau + n)\right],$$

when $c + 2d \leq 2$ and $d \leq f_k$. Choosing τ so that $\tau(n)^{f_k} \geq n$ allows us to simplify this to

$$\Sigma_2\mathrm{TIME}[\tau] \subseteq \Pi_2\mathrm{TIME}[\tau^{(c+d)f_k} \mathrm{polylog}(\tau)]. \tag{23}$$

The inclusion (23) gives a contradiction with Lemma 5 for any k with $f_k < \frac{1}{c+d}$. Note that $f_k \to 0$ as $k \to \infty$ if $c + 2d < 2$. Therefore, all that remains is to show that the latter condition is compatible with the other ones, i.e., that we can pick a constant $d > 0$ and an integer $k > 0$ such that

$$\begin{aligned} c + 2d &< 2, & (24)\\ d &\leq f_k, \text{ and} & (25)\\ f_k &< \frac{1}{c+d}. & (26) \end{aligned}$$

For any c and d satisfying (24), consider choosing $k \geq 1$ to be the smallest integer such that (26) is satisfied. Observe that $f_k \geq \frac{c+d}{2} f_{k-1}$, and by how we chose k, $f_{k-1} \geq \frac{1}{c+d}$. This shows that $f_k \geq 1/2$, so (25) is satisfied when $d \leq 1/2$. From (24), we have $d < \frac{2-c}{2}$, which is at most $1/2$ when $c \geq 1$. Therefore, choosing d such that $d < \frac{2-c}{2}$ and then calculating k as described above yields a d and k satisfying all of the constraints, leading to the desired contradiction. As c approaches 1 from above, $\frac{2-c}{2}$ approaches $1/2$ from below, so the largest value of d that yields a contradiction approaches $1/2$ as well. This proves that $\Sigma_2\mathrm{TIME}[n] \not\subseteq \mathrm{BPTISP}[n^c, n^d]$ for such c and d. $\square$

We point out that, although we can handle the same values of c as in the deterministic setting, the dependence of d on c in Theorem 15 is worse. In particular, the proofs of the time-space lower bounds for deterministic machines show that d approaches 1 from below as c approaches 1 from above [9], while in our result d approaches $1/2$ from below as c approaches 1 from above.

The proof of Theorem 15 generalizes to $\Sigma_\ell\mathrm{TIME}[n]$ for any $\ell \geq 3$. In this setting, the hypothesis becomes

$$\Sigma_\ell\mathrm{TIME}[n] \subseteq \mathrm{BPTISP}[n^c, n^d]. \tag{27}$$

Along with the Π_2-simulation of $\mathrm{BPTISP}[T, S]$ of Theorem 11, this yields a collapse of the form $\Sigma_\ell \subseteq \Pi_2$, which allows us to eliminate more than one alternation at the same cost of removing one alternation in the setting of Theorem 15 where $\ell = 2$. Therefore, we can afford to use more alternations using (4) and achieve a greater speedup. In a manner analogous to the proof of Theorem 15, we derive a series of increasingly efficient complementations of Σ_ℓ to Π_ℓ for $c < \ell$, eventually reaching a contradiction as long as $d \leq \frac{1}{\sqrt{\ell}}$.

Alternatively, we can use extra alternations to achieve a dependence of d on c where d approaches 1 from below as c approaches 1 from above, as in the deterministic case. For example, when $\ell = 3$, this follows by deriving a complementation of Σ_3 following the same technique that derives the complementation of Σ_2 given by (18) when $\ell = 2$, modulo the replacement of the Π_2-simulation given by Theorem 11 with the Π_3-simulation given by Theorem 13. Specifically, hypothesis (27) and Theorem 13 give a complementation of Σ_3:

$$\Sigma_3\mathrm{TIME}[n] \subseteq \forall^{n^d \log n}\exists^{n^d \log n}\forall^{\log n}\mathrm{DTISP}[n^c \operatorname{polylog}(n), n^d \log n]. \tag{28}$$

Consider a $\Sigma_3\mathrm{TIME}[\tau]$ computation for some time function τ to be determined. Applying the speedup of (2) to the final deterministic stage of the simulation given by (28) yields

$$\Sigma_3\mathrm{TIME}[\tau] \subseteq \forall^{\tau^d \log \tau} \underbrace{\exists^{\tau^d \log \tau}\Pi_2\mathrm{TIME}[\tau^{\frac{c+d}{2}} \operatorname{polylog}(\tau)]}_{(\alpha)}.$$

We can now use (28) to simulate (α) by a Π_3-machine, yielding a more efficient complementation than (28) for certain values of c and d:

$$\Sigma_3\mathrm{TIME}[\tau] \subseteq \forall^{\tau^d \log \tau}\Pi_3\mathrm{TIME}[(\tau^{\frac{c+d}{2}} + n + \tau^d)^c \operatorname{polylog}(\tau)].$$

When $\tau(n) \geq n^{\frac{2}{c+d}}$ (and $d \leq c$), this simplifies to

$$\Sigma_3\mathrm{TIME}[\tau] \subseteq \Pi_3\mathrm{TIME}[\tau^{c\frac{c+d}{2}} \operatorname{polylog}(\tau)], \tag{29}$$

yielding a contradiction to Lemma 5 when $c < \sqrt{2}$ and $d < \frac{2-c^2}{c}$. Therefore, as c approaches 1 from above, the upper bound on d approaches 1 from below when $\ell = 3$. Indeed, this analysis establishes the desired behavior of d as c approaches 1 for any level $\ell \geq 3$, since if (27) holds for $\ell > 3$, it must also hold for $\ell = 3$.

We point out that we can augment the strategy leading to (28) with a bootstrapping argument similar to the one in the proof of Lemma 14 and obtain increasingly efficient complementations of Σ_3. This approach results in a contradiction for $c < 2$, whereas the analogous approach using Theorem 11 leads to a contradiction for $c < 3$. More generally, at level $\ell \geq 3$, we can modify the above approach to take full advantage of the stronger hypothesis and arrive at complementations of Σ_ℓ. However, we only reach a contradiction for $c < \ell - 1$ whereas the strategy based on Theorem 11 results in a contradiction for $c < \ell$. Therefore, we need *both* approaches to prove Theorem 1 – the latter achieves the lower bound for large values of c, while the former establishes the dependence of d on c for small values of c.

Since much of the proof of Theorem 1 closely follows the outline of Theorem 15, we give only a brief sketch of it here. In fact, Theorem 1 also follows from the more general Theorem 2, which gives time-space lower bounds for simulations of space-bounded linear-time alternating machines. A complete proof of Theorem 2 appears in the next section.

Proof of Theorem 1. The case for $c < 1$ follows by standard techniques, as in the proof of Theorem 15.

For the case $c \geq 1$, assume that $\Sigma_\ell\text{TIME}[n] \subseteq \text{BPTISP}[n^c, n^d]$ for some constant $d > 0$ to be determined later. Using the speedup given by (4) rather than (2), we can step through an argument similar to that of Lemma 14 to show

$$\Sigma_\ell\text{TIME}[T] \subseteq \Pi_\ell\text{TIME}\left[\left((T^{g_k} + n)^{c+d}\right)\text{polylog}(T+n)\right], \tag{30}$$

as long as $c + \ell d \leq \ell$ and $d \leq g_k$, where

$$g_k = (\frac{c + \ell d}{\ell})^k. \tag{31}$$

We can now use (30) as we used Lemma 14 in the proof of Theorem 15. For a time function τ such that $\tau(n)^{g_k} \geq n$, (30) gives that

$$\Sigma_\ell\text{TIME}[\tau] \subseteq \Pi_\ell\text{TIME}\left[\tau^{(c+d)g_k}\right],$$

which is a contradiction with Lemma 5 when $g_k < \frac{1}{c+d}$. Therefore, it remains to show that it is possible to choose a positive d and integer k satisfying the latter condition as well as those on (30). More specifically, for $c < \ell$ we can choose $d < \frac{\ell - c}{\ell}$ so that there exists a smallest positive integer k such that $g_k < \frac{1}{c+d}$. Since $g_k = \frac{c+\ell d}{\ell} \cdot g_{k-1} \geq \frac{c+\ell d}{\ell} \cdot \frac{1}{c+d}$, we can guarantee that the only constraint possibly left unsatisfied, namely $d \leq g_k$, is met by restricting our choice of d to $d \leq \frac{c+\ell d}{\ell} \cdot \frac{1}{c+d}$.

When $\ell = 2$, the upper bound on d approaches $\frac{1}{2}$ as c approaches 1. For $\ell \geq 3$, the upper bound on d approaches $\frac{1}{\sqrt{\ell}}$ as c approaches 1 but we can improve it using the combination of the hypothesis and Theorem 13 that leads to the complementation of Σ_3 represented by (29). For large enough τ, this gives a contradiction for values of d approaching 1 from below as c approaches 1 from above when $\ell \geq 3$.

Theorem 7 transfers the lower bounds to QSAT_ℓ. □

6 Other Results

In this section, we strengthen Theorem 1 to establish time-space lower bounds for problems decidable by alternating machines that run in linear time and only use a small amount of space. We also prove Theorem 3 regarding time-space lower bounds for tautology on randomized machines with *one-sided* error.

6.1 Sublinear Space on Linear-Time Alternating Machines

By paying close attention to the space used by the simulations in the proof of Theorem 1, we actually obtain time-space lower bounds for randomized simulations of linear-time alternating machines using space n^a for $a < 1$, given by Theorem 2. The main task is to use the weaker assumption that $\Sigma_\ell\text{TISP}[n, n^a] \subseteq \text{BPTISP}[n^c, n^d]$ to eliminate the alternations introduced by the speedup of (3). This requires that the Σ_k-simulation after the speedup uses an amount of space which is at most the a^{th} power of its running time. Since the simulation guesses (and stores) $O(BS)$ bits in each alternating stage, this restricts us to choose a small value of B, which in turn grants a smaller speedup. Therefore, our bounds become weaker as a becomes smaller.

To prove Theorem 2, we first prove an analog of Lemma 14 which gives a series of complementations of $\Sigma_\ell\text{TISP}$.

Lemma 16. *Suppose that*

$$\Sigma_\ell\mathrm{TISP}[n, n^a] \subseteq \mathrm{BPTISP}[n^c, n^d] \tag{32}$$

for some integer $\ell \geq 2$ and constants $0 < a \leq 1$, $c \geq 1$, and $d > 0$ with $c + \ell d \leq 1 + (\ell - 1)a$ and $(1-a)d \leq ac$. Then for any time function T and integer $k \geq 0$ such that $d \leq h_k$,

$$\Sigma_\ell\mathrm{TISP}[T, T^a] \subseteq \Pi_\ell\mathrm{TISP}\left[\left((T^{h_k} + n)^{c+d}\right)\mathrm{polylog}(T+n), (T+n)^d\,\mathrm{polylog}(T+n)\right], \tag{33}$$

where

$$h_k \;=\; \left(\tfrac{c+\ell d}{1+(\ell-1)a}\right)^k. \tag{34}$$

Proof. The base case $k = 0$ is given by combining the hypothesis (32) and the efficient Π_2-simulation of BPTISP given by Theorem 11. We now prove the inductive step $k \to k+1$. Consider a $\Sigma_\ell\mathrm{TISP}[T, T^a]$ computation. From the hypothesis (32), Theorem 11, and the speedup of (3) (to Σ_ℓ rather than Π_ℓ), we obtain a simulation which (neglecting the polylog$(T+n)$ factors) is in

$$\forall^{(T+n)^d}\underbrace{\Sigma_\ell\mathrm{TISP}\left[B(T+n)^d + \frac{(T+n)^{c+d}}{B^{\ell-1}}, B(T+n)^d\right]}_{(\alpha)}.$$

In order to apply the inductive hypothesis to complete the complementation to Π_ℓTISP, the space bound of (α) must be at most the a^{th} power of its running time. This is the case when B satisfies

$$B(T+n)^d = \left(\frac{(T+n)^{c+d}}{B^{\ell-1}}\right)^a,$$

which offers the choice

$$B = (T+n)^{\frac{ac+(a-1)d}{1+(\ell-1)a}}$$

as long as $ac + (a-1)d \geq 0$. For such a choice, (α) is a computation in

$$\Sigma_\ell\mathrm{TISP}\left[(T+n)^{\frac{c+\ell d}{1+(\ell-1)a}}, (T+n)^{a\frac{c+\ell d}{1+(\ell-1)a}}\right],$$

which takes an input of size $O(n + (T+n)^d)$, which is in $O(n + T^d)$ when $d \leq 1$. Thus, instead of achieving an ℓ^{th} root speedup as we do when we are not concerned about the space used by the simulation of (3), we instead achieve a $(1 + (\ell-1)a)^{\text{th}}$ root speedup.

The inductive hypothesis gives a Π_ℓTISP-simulation of (α), and hence a Π_ℓTISP-simulation of $\Sigma_\ell\mathrm{TISP}[T, T^a]$ running in time big-O of (neglecting the polylog$(T+n)$ terms)

$$T^* \doteq (T+n)^d + \left((T+n)^{h_k\frac{c+\ell d}{1+(\ell-1)a}} + n + T^d\right)^{c+d}$$

and using space big-O of

$$S^* \doteq (T+n)^d + \left((T+n)^{\frac{c+\ell d}{1+(\ell-1)a}} + n + T^d\right)^d.$$

When $c + \ell d \leq 1 + (\ell-1)a$ and $d \leq 1$, S^* simplifies to $(T+n)^d$. When we have the further constraint that $d \leq h_{k+1}$, T^* has the appropriate leading terms and simplifies to $(T^{h_{k+1}} + n)^{c+d}$. Accounting for the polylog factors, we have shown that the simulation is of the desired form. ◻

We note that the proof of Lemma 16 also yields a version in which the Π_ℓ on the right-hand side of (33) is replaced by Π_2. However, we do not see a way to exploit that fact to strengthen our final result. For the sake of clarity and consistency, we present Lemma 16 as stated.

When $c + \ell d < 1 + (\ell - 1)a$, $h_k \to 0$ as $k \to \infty$. In such a case, we can use Lemma 16 to derive a contradiction to Lemma 6 in an analogous fashion to how we used Lemma 14 to prove Theorem 15 for values of c as large as possible. For $\ell = 2$, this gives a value of d approaching $\frac{a}{2}$ from below as c approaches 1 from above. We can do better when $\ell \geq 3$ by using Theorem 13 to derive a space-bounded complementation analogous to that given by (29) in the unrestricted case.

Lemma 17. *Suppose that*

$$\Sigma_3\mathrm{TISP}[n, n^a] \subseteq \mathrm{BPTISP}[n^c, n^d] \tag{35}$$

for constants $0 < a \leq 1$, $c \geq 1$ and $0 < d \leq ac$ with $c + d \leq 1 + a$. Then for any time function T,

$$\Sigma_3\mathrm{TISP}[T, T^a] \subseteq \Pi_3\mathrm{TISP}[(T^{\frac{c+d}{1+a}} + n)^c \operatorname{polylog}(T+n), (T+n)^d \operatorname{polylog}(T+n)].$$

Proof. The hypothesis (35) and Theorem 13 yield the complementation

$$\Sigma_3\mathrm{TISP}[n, n^a] \subseteq \forall^{n^d \log n}\exists^{n^d \log^2 n}\forall^{\log n}\mathrm{DTISP}[n^c \operatorname{polylog}(n), n^d \log n]. \tag{36}$$

Consider a $\Sigma_3\mathrm{TISP}[T, T^a]$ computation. Padding (36) and applying the speedup of (1) (on the Π_2-side) to the final deterministic stage gives a simulation of $\Sigma_3\mathrm{TISP}[T, T^a]$ which (neglecting the polylog terms) is in

$$\forall^{(T+n)^d}\exists^{(T+n)^d}\underbrace{\Pi_2\mathrm{TISP}[B(T+n)^d + (T+n)^c/B, B(T+n)^d]}_{(\beta)}.$$

Choosing $B = (T+n)^{\frac{ac-d}{1+a}}$ so that the in the space bound of (β) is the a^{th} power of its running time places the simulation in

$$\forall^{(T+n)^d}\underbrace{\exists^{(T+n)^d}\Pi_2\mathrm{TISP}[(T+n)^{\frac{c+d}{1+a}}, (T+n)^{a\frac{c+d}{1+a}}]}_{(\gamma)},$$

when $d \leq ac$. Under the same condition, (γ) is a computation in $\Sigma_3\mathrm{TISP}[(T+n)^{\frac{c+d}{1+a}}, (T+n)^{a\frac{c+d}{1+a}}]$ taking an input of size $n + (T+n)^d$, so that (36) gives a simulation of (γ) in $\Pi_3\mathrm{TISP}$. Overall we have derived

$$\begin{aligned}\Sigma_3\mathrm{TISP}[T, T^a] &\subseteq \Pi_3\mathrm{TISP}[\left((T+n)^{\frac{c+d}{1+a}} + n + (T+n)^d\right)^c, (T+n)^d + \left((T+n)^{\frac{c+d}{1+a}} + n + (T+n)^d\right)^d] \\ &\subseteq \Pi_3\mathrm{TISP}[(T^{\frac{c+d}{1+a}} + n)^c, (T+n)^d],\end{aligned}$$

where the last inclusion follows as long as $d \leq ac$ and $\frac{c+d}{1+a} \leq 1$. Accounting for the $\operatorname{polylog}(T+n)$ terms finishes the proof. $\square$

We are now ready to prove Theorem 2.

Proof of Theorem 2. The proof for $c < 1$ follows from standard techniques, as in the proof of Theorem 15.

For the case $c \geq 1$, assume by way of contradiction that $\Sigma_\ell\mathrm{TISP}[n, n^a] \subseteq \mathrm{BPTISP}[n^c, n^d]$ for some constant $d > 0$ to be determined later. Then for a time function τ where $\tau(n)^{h_k} \geq n$, Lemma 16 gives the complementation

$$\Sigma_\ell\mathrm{TISP}[\tau, \tau^a] \subseteq \Pi_\ell\mathrm{TISP}\left[\tau^{(c+d)h_k}\,\mathrm{polylog}(\tau), \tau^d\,\mathrm{polylog}(\tau)\right] \tag{37}$$

when $c + \ell d \leq 1 + (\ell-1)a$, $(1-a)d \leq ac$, and $d \leq h_k$. When $h_k < \frac{1}{c+d}$, the time bound of the right-hand side of (37) is $o(\tau)$. Provided that $c < 1 + (\ell-1)a$, we can choose a positive d and an integer k such that all the above conditions are met. More specifically, for any value of $d < \frac{1+(\ell-1)a-c}{\ell}$, there exists a smallest positive integer k such that $h_k < \frac{1}{c+d}$. Since $h_k = \frac{c+\ell d}{1+(\ell-1)a} \cdot h_{k-1} \geq \frac{c+\ell d}{1+(\ell-1)a} \cdot \frac{1}{c+d}$, we can guarantee that $d \leq h_k$ by imposing the condition

$$d \leq \frac{c + \ell d}{1 + (\ell-1)a} \cdot \frac{1}{c+d}. \tag{38}$$

It follows that we can meet all constraints mentioned so far by choosing d below some positive threshold. All that remains to reach a contradiction with Lemma 6 is to ensure that the space bound of the right-hand side of (37) is $o(\tau^a)$, which can be done with the additional constraint on d that $d < a$.

For $\ell = 2$ the upper bound on d approaches $\frac{a}{2}$ from below as c approaches 1 from above. In the case $\ell \geq 3$ the upper bound on d imposed by the conditions other than (38) approaches $\frac{\ell-1}{\ell}a$ as c approaches 1; condition (38) implies an upper bound of $1/\sqrt{\ell}$ for $a = 1$ and a somewhat weaker bound for smaller values of a. However, we can achieve a contradiction for larger d as c approaches 1 and $\ell \geq 3$ using the following argument. For $\tau(n) \geq n^{\frac{1+a}{c+d}}$, Lemma 17 gives

$$\Sigma_3\mathrm{TISP}[\tau, \tau^a] \subseteq \Pi_3\mathrm{TISP}[\tau^{c\frac{c+d}{1+a}}\,\mathrm{polylog}(\tau), \tau^d\,\mathrm{polylog}(\tau)]$$

when $c + d \leq 1 + a$ and $d \leq ac$. When $c < \sqrt{1+a}$, we can choose $d < \min(\frac{(1+a)-c^2}{c}, a)$ and arrive at a contradiction with Lemma 6. As c approaches 1 from above, the upper bound on d approaches a from below, which gives the desired dependence. $\square$

6.2 Tautology on Randomized Machines with One-Sided Error

In this section, we consider problems in the first level of the polynomial-time hierarchy and establish the time-space lower bounds of Theorem 3 for tautology on randomized machines with one-sided error. To do so, we actually prove time-space lower bounds for a more powerful class of machines, namely nondeterministic machines which are restricted to guess few bits.

Theorem 18. *There exists positive constants b and d such that tautology cannot be solved by nondeterministic random-access machines which run in time $n^{1.759}\,\mathrm{polylog}(n)$, space n^d, and nondeterministically guess only n^b bits.*

By way of the space-bounded derandomization of Theorem 10, a space-bounded randomized machine with one-sided error can be made to use very few random bits. Since a randomized machine with one-sided error is also a special type of nondeterministic machine, we can view the one-sided error machines obtained by Theorem 10 as nondeterministic machines which guess very few random bits. Thus, the time-space lower bounds of Theorem 3 follow as a corollary to Theorem 18.

Proof of Theorem 3. Let b^* the value of b given by Theorem 18, and d^* the value of d. By the derandomization of Theorem 10, if tautology can be solved in $\mathrm{RTISP}[n^c, n^d]$, then it can also be solved in $\exists^{n^d \log n}\mathrm{DTISP}[n^c \operatorname{polylog}(n), n^d]$. Thus by Theorem 18, tautology is not in $\mathrm{RTISP}[n^{1.759}, n^d]$ for any $d < \min(b^*, d^*)$. $\square$

We now focus on proving Theorem 18 using the ideas outlined in Section 1.2. In this setting, the hypothesis of the indirect diagonalization argument becomes $\mathrm{NTIME}[n] \subseteq \forall^{n^b}\mathrm{DTISP}[n^c, n^d]$. This unlikely scenario certainly yields an efficient complementation, and it is actually strong enough to allow for something more: Under this hypothesis, we can improve the space-bounded speedups of (2) and (4) for certain values of b, c and d.

Lemma 19. *Suppose that*

$$\mathrm{NTIME}[n] \subseteq \forall^{n^b}\mathrm{DTISP}[n^c, n^d] \tag{39}$$

for some constants b, $c \geq 1$, and d such that $c + d \leq 2$ and $b \leq c + d - 1$. Then for any time function T, space function S and integers $i \geq 0$ and $k \geq 2$,

$$\mathrm{DTISP}[T, S] \subseteq \Pi_k\mathrm{TIME}[(TS^{k-1})^{\frac{\gamma_i}{(k-2)\gamma_i+1}} + n + S], \tag{40}$$

where $\gamma_0 = \frac{1}{2}$ and $\gamma_{i+1} = \frac{(c+d)\gamma_i}{(c+d)\gamma_i+1}$.

We point out some facts about the sequence $(\gamma_i)_i$ in order to clearly assess the speedup represented by Lemma 19. From the definition, we can see that $\gamma_{i+1} \leq \gamma_i$ if and only if $\gamma_i \leq \gamma_{i-1}$. It follows that the sequence $(\gamma_i)_i$ is monotonic. Since the transformation $x \mapsto \frac{(c+d)x}{(c+d)x+1}$ has a unique attractive fixed point at $1 - \frac{1}{c+d}$, $(\gamma_i)_i$ converges to this value. Specifically, when $c + d < 2$, this fixed point is less than $\frac{1}{2} = \gamma_0$, so in this case $(\gamma_i)_i$ decreases monotonically to $1 - \frac{1}{c+d}$.

When S is small, Lemma 19 essentially offers a $((k-2) + \frac{1}{\gamma_i})^{\text{th}}$-root speedup of a $\mathrm{DTISP}[T, S]$ machine on a Π_k-machine, provided this running time remains at least linear. From the convergence properties of $(\gamma_i)_i$, we can see that this speedup approaches the $(k-1+\frac{1}{c+d-1})^{\text{th}}$-root as i increases. Recall that the unconditional speedup offered by (4) gives a similar k^{th}-root speedup. Thus, when the hypothesis (39) holds for $c + d < 2$, Lemma 19 offers a greater speedup than we had unconditionally.

To prove Lemma 19, we start with the case $k = 2$ by inductively deriving better and better speedups of $\mathrm{DTISP}[T, S]$ into Π_2. For the case $k > 2$, we use $k - 2$ alternations for a speedup as in (3) and then one more alternation to speed up the final deterministic phase in (3) by applying (40) for $k = 2$. An optimal choice of the number of blocks B yields the result.

Proof of Lemma 19. We prove the case $k = 2$ by induction on i. In particular, we need to prove

$$\mathrm{DTISP}[T, S] \subseteq \Pi_2\mathrm{TIME}[(TS)^{\gamma_i} + n + S]. \tag{41}$$

For the base case, (41) holds unconditionally for $i = 0$ by the standard square root speedup of (2). For the inductive step $i \to i + 1$, consider a $\mathrm{DTISP}[T, S]$ computation. Using the Π_2-version of the inclusion (1), we speed up this computation in Π_2:

$$\mathrm{DTISP}[T, S] \subseteq \forall^{BS} \underbrace{\exists^{\log B}\mathrm{DTISP}[T/B, S]}_{(i)}.$$

We can see that (i) represents a computation in nondeterministic time $O(T/B)$ taking an input of length $O(n+BS)$. Thus, the hypothesis (39) gives a simulation of (i) which yields

$$\mathrm{DTISP}[T,S] \subseteq \underbrace{\forall^{BS}\forall^{(T/B+n+BS)^b}}_{(ii)}\underbrace{\mathrm{DTISP}[(T/B+n+BS)^c,(T/B+n+BS)^d]}_{(iii)}.$$

Notice that the final space-bounded deterministic stage (iii) takes an input of size $O(n+BS+(T/B)^b)$ provided $b \leq 1$, so that the inductive hypothesis yields a simulation of this stage in

$$\Pi_2\mathrm{TIME}[((T/B+n+BS)^{c+d})^{\gamma_i} + n + BS + (T/B)^b + (T/B+n+BS)^d].$$

Merging the initial universal phase of this simulation with that represented by (ii) and noting that $B \geq 1$, we see that we have arrived at a simulation of $\mathrm{DTISP}[T,S]$ on a Π_2-machine running in time big-O of

$$BS + (T/B+n+BS)^{\max(b,d)} + ((T/B+n+BS)^{c+d})^{\gamma_i} + n.$$

To simplify this, notice that when $c+d \leq 2$, we have that $(c+d)\gamma_i \leq 1$ (since $\gamma_i \leq \frac{1}{2}$). If $c \geq 1$, we have $d \leq (c+d)\gamma_i$ (since $\gamma_i \geq 1 - \frac{1}{c+d}$). Furthermore, if $b \leq c+d-1$, we have $b \leq (c+d)\gamma_i$ (since $\gamma_i \geq 1 - \frac{1}{c+d}$). Under these conditions, the above running time simplifies to big-O of

$$BS + (T/B)^{(c+d)\gamma_i} + n.$$

To minimize this running time up to a constant factor, we choose a value for B such that $BS = (T/B)^{(c+d)\gamma_i}$, namely $B^* \doteq (\frac{T^{(c+d)\gamma_i}}{S})^{1/((c+d)\gamma_i+1)}$. If $B^* \geq 1$, this choice results in a running time in big-O of

$$(TS)^{\frac{(c+d)\gamma_i}{(c+d)\gamma_i+1}} + n = (TS)^{\gamma_{i+1}} + n.$$

On the other hand, if $B^* < 1$, then $B = 1$ is the best we can do. This yields a running time of $O(S+n)$. In either case, $O((TS)^{\gamma_{i+1}} + n + S)$ is an upper bound on the running time. By induction, (41) holds for all $i \geq 0$ and b, c, and d as above.

Now that we have established (41), we use it to establish (40) for $k > 2$. The first step is to use (3) to speed up a $\mathrm{DTISP}[T,S]$-machine on a Π_{k-1}-machine. This yields

$$\mathrm{DTISP}[T,S] \subseteq \underbrace{\forall^{BS}\exists^{BS}\ldots Q^{\log B}}_{k-1}\underbrace{\mathrm{DTISP}[T/B^{k-2},S]}_{(\alpha)}, \tag{42}$$

where Q is $\forall$ if k is even, and $\exists$ if k is odd.

We can see that (α) represents a computation taking an input of size $O(n+BS)$ and running in time T/B^{k-2} and space S. Provided b, c, and d satisfy the constraints of the lemma, (41) gives simulations of (α) on Π_2-machines running in time

$$O\left((TS/B^{k-2})^{\gamma_i} + n + BS\right),$$

for $B \geq 1$. Since deterministic classes are closed under complement, we also get simulations on Σ_2-machines running in the same amount of time. Choosing the former if $Q = \forall$ in (42), and the latter otherwise, the alternating stages align properly so that replacing (α) in this manner adds only one alternation. Overall, we arrive at a simulation of $\mathrm{DTISP}[T,S]$ by a Π_k-machine running in the above time bound. To minimize this running time up to a constant factor, we choose B so that the two terms depending on B are equal. This occurs at the value

$$B^\dagger \doteq (T^{\gamma_i} S^{\gamma_i - 1})^{\frac{1}{(k-2)\gamma_i+1}}.$$

When $B^\dagger \geq 1$, such a choice yields the running time $O\left((TS^{k-1})^{\frac{\gamma_i}{(k-2)\gamma_i+1}} + n\right)$. If $B^\dagger < 1$, then $B = 1$ is the best we can do and the running time becomes $O(S + n)$. In both cases, we obtain an upper bound of

$$O\left((TS^{k-1})^{\frac{\gamma_i}{(k-2)\gamma_i+1}} + n + S\right)$$

on the running time, which proves the claim. □

We now use a bootstrapping argument to derive a series of increasingly efficient complementations of the linear-time hierarchy at higher and higher levels. At each level, the improved speedup granted by Lemma 19 for sufficiently small b and d allows a more efficient complementation than the unconditional speedup of (4), which is key to obtaining quantitatively stronger lower bounds.

Lemma 20. *For any constants $1 \leq c < 2$, $\epsilon > 0$, and integer $\ell \geq 2$, there exist positive constants b and d such that if*

$$\mathrm{NTIME}[n] \subseteq \forall^{n^b} \mathrm{DTISP}[n^c, n^d], \tag{43}$$

then

$$\Sigma_\ell \mathrm{TIME}[n] \subseteq \Pi_\ell \mathrm{TIME}[n^{c_\ell + \epsilon}], \tag{44}$$

where

$$c_\ell = \begin{cases} c(c-1) & \textit{for } \ell = 2 \\ \frac{c^2(c-1)\prod_{j=2}^{\ell-1} c_j}{(\ell-1)(c-1)+1} & \textit{otherwise.} \end{cases} \tag{45}$$

Closed forms for the exponent c_ℓ defined by (45) become rather complex. One can show by induction that

$$c_\ell = \frac{c^{3 \cdot 2^{\ell-3}} (c-1)^{2^{\ell-2}}}{((\ell-1)(c-1)+1) \cdot \prod_{k=3}^{\ell-1}((k-1)(c-1)+1)^{2^{\ell-k-1}}}$$

for $\ell > 2$.

Proof of Lemma 20. Let c and ϵ be given. We argue by induction that we can choose b and d appropriately so that (43) yields the desired inclusion. For $\ell = 2$, we use the hypothesis to obtain a DTISP simulation of $\Sigma_2\mathrm{TIME}[n]$ when $b \leq c$:

$$\begin{aligned} \Sigma_2\mathrm{TIME}[n] = \exists^n \forall^n \mathrm{DTIME}[n] &\subseteq \exists^n \exists^{n^b} \mathrm{DTISP}[n^c, n^d] \subseteq \mathrm{NTIME}[n^c] \\ &\subseteq \forall^{n^{bc}} \mathrm{DTISP}[n^{c^2}, n^{cd}]. \end{aligned}$$

The input to the final deterministic stage is of size $n + n^{bc}$, so applying Lemma 19 to simulate this stage yields

$$\Sigma_2\mathrm{TIME}[n] \subseteq \forall^{n^{bc}} \mathrm{DTISP}[n^{c^2}, n^{cd}] \subseteq \Pi_2\mathrm{TIME}[(n^{c(c+d)})^{\gamma_i} + n + n^{bc} + n^{cd}].$$

As $i \to \infty$, the exponent $c(c+d)\gamma_i \to c(c+d-1)$. For d such that this limit is at most $c(c-1) + \frac{\epsilon}{2}$, we can choose i large enough so that the exponent is at most $c(c-1) + \epsilon$. Under the additional constraints $b, d \leq c-1$, the running time becomes $O(n^{c(c-1)+\epsilon}+n)$. Therefore, when $c(c-1)+\epsilon \geq 1$, we have the desired inclusion. In fact, this is the only case we need to consider, for if $c(c-1)+\epsilon < 1$, we can find $a > 1$ such that $ac(c-1)+\epsilon = 1$ and apply the above argument to $\Sigma_2\text{TIME}[n^a]$. This yields the inclusion $\Sigma_2\text{TIME}[n^a] \subseteq \Pi_2\text{TIME}[n]$, which contradicts Lemma 5. Therefore, the claim holds for $\ell = 2$.

Now suppose that for $2 \leq k < \ell$, $\Sigma_k\text{TIME}[n] \subseteq \Pi_k\text{TIME}[n^{c_k+\epsilon'}]$ for an ϵ' to be determined later. Let $b_{k,\epsilon'}$ and $d_{k,\epsilon'}$ denote the appropriate values of b and d given by the inductive hypothesis for the complementation at Σ_k to hold. To show that the desired complementation holds for Σ_ℓ, we first derive a simulation of such a computation in $\forall^n\text{DTISP}[n^{O(1)}, n^d]$ and then use Lemma 19 to achieve a faster simulation in Π_ℓ.

We accomplish the former by iteratively deriving simulations one level lower in the polynomial-time hierarchy. At step $j = 0, \ldots, \ell-3$, we start with a $\Sigma_{\ell-j}$-machine and use a complementation given by the inductive hypothesis to obtain a simulation by a $\Sigma_{\ell-j-1}$-machine. This follows by complementing the computation following the initial existential stage of the $\Sigma_{\ell-j}$-machine, which is a $\Pi_{\ell-j-1}$-computation. Thus, when the hypothesis holds for $b \leq b_{\ell-j-1,\epsilon'}$ and $d \leq d_{\ell-j-1,\epsilon'}$, we derive a simulation with one fewer alternation while raising the running time to the power of $c_{\ell-j-1}+\epsilon'$. This lets us write:

$$\begin{aligned} \Sigma_\ell\text{TIME}[n] &\subseteq \Sigma_{\ell-1}\text{TIME}[n^{c_{\ell-1}+\epsilon'}] \\ &\subseteq \Sigma_{\ell-2}\text{TIME}[n^{(c_{\ell-1}+\epsilon')(c_{\ell-2}+\epsilon')}] \\ &\cdots \\ &\subseteq \Sigma_2\text{TIME}[n^{(c_{\ell-1}+\epsilon')(c_{\ell-2}+\epsilon')\cdots(c_2+\epsilon')}]. \end{aligned}$$

Defining $C_{\ell,\epsilon'} \doteq (c_{\ell-1}+\epsilon')(c_{\ell-2}+\epsilon')\cdots(c_2+\epsilon')$ and applying the hypothesis (43) twice to the latter simulation (as in the base case), we obtain the desired simulation:

$$\Sigma_\ell\text{TIME}[n] \subseteq \underbrace{\forall^{n^{bcC_{\ell,\epsilon'}}}}_{(\alpha)} \underbrace{\text{DTISP}[n^{c^2C_{\ell,\epsilon'}}, n^{cdC_{\ell,\epsilon'}}]}_{(\beta)}.$$

The input to (β) is of size $O(n + n^{bcC_{\ell,\epsilon'}})$, so applying Lemma 19 to this stage and absorbing the universal stage (α) gives

$$\Sigma_\ell\text{TIME}[n] \subseteq \Pi_\ell\text{TIME}[\underbrace{(n^{c(c+(\ell-1)d)C_{\ell,\epsilon'}})^{\frac{\gamma_i}{(\ell-2)\gamma_i+1}}}_{(*)} + n + \underbrace{n^{bcC_{\ell,\epsilon'}} + n^{cdC_{\ell,\epsilon'}}}_{(**)}], \tag{46}$$

for small enough b and d and any integer $i \geq 0$. When d is yet further restricted, the exponent of the term $(*)$ approaches $\frac{c^2(c-1)C_{\ell,\epsilon'}}{(\ell-1)(c-1)+1} + \frac{\epsilon}{4}$ as i grows. This allows the choice of i large enough so this exponent is at most $\frac{\epsilon}{4}$ away from its limit point, namely at most $\frac{c^2(c-1)C_{\ell,\epsilon'}}{(\ell-1)(c-1)+1} + \frac{\epsilon}{2}$. We next choose ϵ' small enough so that all of the terms in the exponent of $(*)$ involving ϵ' sum to at most $\frac{\epsilon}{2}$. Under these circumstances, an upper bound for the exponent of $(*)$ is

$$\frac{c^2(c-1)C_{\ell,\epsilon'}}{(\ell-1)(c-1)+1} + \frac{\epsilon}{2} \leq \frac{c^2(c-1)\prod_{i=2}^{\ell-1} c_i}{(\ell-1)(c-1)+1} + \epsilon = c_\ell + \epsilon.$$

When we also have $b, d \leq \frac{c(c-1)}{(\ell-1)(c-1)+1}$, the term $(*)$ dominates $(**)$, so the Π_ℓ-simulation represented by (46) runs in time $O(n^{c_\ell+\epsilon}+n)$. When $c_\ell+\epsilon \geq 1$, this shows that (44) holds for b and d small enough to meet all of the above constraints. This is the only case we need to consider, since $c_\ell + \epsilon < 1$ results in a contradiction to Lemma 5 by applying the above argument to $\Sigma_\ell\text{TIME}[n^a]$ for an appropriate $a > 1$ (as in the step for $\ell = 2$). $\square$

Lemma 20 gives a series of complementations which are increasingly unlikely and eventually contradict Lemma 5 for certain values of c. We obtain Theorem 18 by analyzing the behavior of the sequence of exponents c_ℓ defined by (45).

Proof of Theorem 18 (See page 23 for the statement). For any integer $\ell \geq 2$, consider c_ℓ defined by (45) as a function of c. One can show by induction on ℓ that c_ℓ monotonically grows from $c_\ell = 0$ at $c = 1$ to infinity. By continuity, there exists a unique value c_ℓ^* at which c_ℓ equals 1. For values $c < c_\ell^*$, Lemma 20 gives a contradiction to Lemma 5 for a choice of ϵ small enough such that $c_\ell + \epsilon < 1$. Thus, we can rule out simulations of $\text{NTIME}[n]$ in $\forall^{n^b}\text{DTISP}[n^c, n^d]$ for all $c < c_\ell^*$ and b, d given by Lemma 20.

By (45), we have that at $c = c_\ell^*$

$$c_{\ell+1} = \frac{c_{\ell+1}}{c_\ell} = \left(1 - \frac{1}{\ell + \frac{1}{c-1}}\right) \cdot c_\ell < 1.$$

The monotonicity of $c_{\ell+1}$ then implies that $c_{\ell+1}^* > c_\ell^*$, i.e., the sequence $(c_\ell^*)_\ell$ increases. Numerical calculations show that $c_{14}^* \approx 1.759708$ and $c_{15}^* \approx 1.759719$, which is enough to prove the claimed lower bound of $n^{1.759}\,\text{polylog}(n)$. $\square$

7 Further Research

The techniques discussed in this work allow us to establish time-space lower bounds for QSAT_ℓ on two-sided error randomized machines for $\ell \geq 2$. They do not seem to extend to the first-level problems of satisfiability or tautology in a straightforward way. The only reason is our inability to exploit the assumption $\text{NTIME}[n] \subseteq \text{BPTISP}[t, s]$ to obtain an efficient complementation at some level of the polynomial-time hierarchy. Thus, establishing time-space lower bounds for satisfiability on randomized machines with two-sided error remains open.

We employed and further developed a technique from [24] to improve the known lower bounds for satisfiability on deterministic machines. The original technique also leads to improved lower bounds for QSAT_ℓ with $\ell \geq 2$ on deterministic machines. For example, it allows the boosting of the time lower bound for QSAT_2 on deterministic subpolynomial-space machines from n^2 [9] to $n^{2.761}$ [24]. Although we were able to find purchase in adopting these techniques to establish better lower bounds for randomized machines with one-sided error, we have been unable to adopt them to improve our results for machines with two-sided error. As a next step, we suggest to find a way to extend the improved lower bounds for QSAT_ℓ with $\ell \geq 2$ on deterministic machines to randomized machines with two-sided error.

Acknowledgements

We would like to thank Bess Berg as well as the anonymous referees of ICALP 2005 and of the journal submission for their helpful comments. We are grateful to Emanuele Viola for pushing us to optimize the space parameter in Theorems 1 and 2.

References

[1] E. Allender, M. Koucky, D. Ronneburger, S. Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, pages 295–302. IEEE, 2001.

[2] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[3] P. Beame, M. Saks, X. Sun, and E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *Journal of the ACM*, 50(2):154–195, 2003.

[4] L. Carter and M. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[5] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 14–19. IEEE, 1989.

[6] S. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26:269–270, 1988.

[7] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. In *Proceedings of the 32nd International Colloquium On Automata, Languages and Programming*, pages 982–993. Springer-Verlag, 2005.

[8] L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60:337–353, 2000.

[9] L. Fortnow and D. van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 2–13. IEEE, 2000.

[10] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22:407–420, 1981.

[11] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.

[12] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13:533–546, 1966.

[13] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 248–253. IEEE, 1989.

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

[15] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17:215–217, 1983.

[16] J. H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1991.

[17] R. Lipton and A. Viglas. On the complexity of SAT. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 459–464. IEEE, 1999.

[18] G. Margulis. Explicit construction of concentrators. *Problems of Information Transmission*, 9:325–332, 1973.

[19] D. van Melkebeek. Time-space lower bounds for NP-complete problems. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, pages 265–291. World Scientific, 2004.

[20] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107:135–144, 1993.

[21] N. Nisan. RL $\subseteq$ SC. *Computational Complexity*, 4:1–11, 1994.

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

[23] E. Viola. On probabilistic time versus alternating time. Technical Report TR-05-137, Electronic Colloquium on Computational Complexity, 2005.

[24] R. Williams. Better time-space lower bounds for sat and related problems. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, pages 40–49. IEEE, 2005.

Appendix

We now prove some results on the complexity of Nisan's generator (Theorem 10) and on deterministic amplification by random walks on a Gabber-Galil expander graph (Lemma 12).

A Nisan's Generator

Theorem 10 follows from an analysis of a time- and space-efficient implementation of Nisan's pseudorandom generator using fast Fourier transform multiplication methods to quickly evaluate and invert linear hash functions. In fact, we prove a somewhat stronger version of Theorem 10.

Theorem 21. *Any randomized machine M running in time T and space S with error ϵ can be simulated by another randomized machine running in time $O(T \log^2 S \log\log S)$, space $O(S \log T)$, and using only $O(S \log T)$ random bits. If two-way access to the random bits is allowed, the space requirement is reduced to $O(S)$. The error of the simulation is $\epsilon + 2^{-S}$, and is one-sided if M has one-sided error.*

Theorem 21 gives a tighter time bound than the bound of $O(T \operatorname{polylog}(T))$ stated in Theorem 10. Our arguments in Sections 5 and 6 are not noticeably improved by using the tighter bound stated in Theorem 21, so we use the simpler bounds of Theorem 10 there for clarity. We state the tighter bound here because it may be of independent interest.

Before proving Theorem 21, we introduce Nisan's pseudorandom generator [20] and discuss some of its properties. Define

$$G_{m,k} : \{0,1\}^m \times H_m^k \to (\{0,1\}^m)^{2^k},$$

where H_m is a family of two-universal hash functions $h : \{0,1\}^m \to \{0,1\}^m$ [4]. The evaluation of $G_{m,k}$ is defined recursively as

$$G_{m,k}(y, h_1, \ldots, h_k) = \begin{cases} y & \text{if } k = 0 \\ G_{m,k-1}(y, h_1, \ldots, h_{k-1}) \circ G_{m,k-1}(h_k(y), h_1, \ldots, h_{k-1}) & \text{otherwise,} \end{cases}$$

where "$\circ$" denotes concatenation. Given a randomized machine M running in time T and space S, we define $G \doteq G_{m,k}$ for $k = \log \frac{T}{m}$ where $m = \Theta(S)$ will be determined later. The simulation of M proceeds with the output of G as the random string, one block of length m at a time. Nisan proves that G fools M in the following sense:

Theorem 22 (Nisan [20]). *There exists a constant ν such that if M is a randomized machine running in time T and space S on input x, then for $m \geq \nu \cdot S$ and $k = \log \frac{T}{m}$,*

$$\left| \Pr_r[M(x,r) \text{ accepts}] - \Pr_{y,h_1,\ldots,h_k}[M(x, G_{m,k}(y, h_1, \ldots, h_k)) \text{ accepts}] \right| \leq 2^{-S},$$

where $M(x,r)$ denotes the outcome of running M on input x and random string r.

This satisfies the error requirements of Theorem 21, so all that remains to prove Theorem 21 is to show how to simulate M on the random string $G(y, h_1, \ldots, h_k)$ within the correct bounds on the time, space, and randomness.

We start with the standard simulation that computes G in a block-wise fashion, where each subsequent m-bit block is computed after m simulation steps of M using the current block. One way to do this is to compute each block from scratch, namely, apply the appropriate sequence of at most k hash functions to the m-bit seed y. Although this technique is good enough to derive our main results, we can do slightly better, namely by a factor of $O(\frac{\log S}{\log(T/S)})$. This improvement follows by computing each block in a recursive manner, which avoids the calculations that the "from scratch" method does over and over again. We now work out the details of our simulation to complete the proof of Theorem 21.

Proof of Theorem 21. To accomodate the approach described above, we choose $m = \Theta(S)$ such that $m \geq \nu \cdot S$ and is of the form $2 \cdot 3^q$ for some integer $q \geq 0$. The latter guarantees a simple explicit formula for an irreducible polynomial of degree m over GF(2), namely $z^{2 \cdot 3^q} + z^{3^q} + 1$ [16, Theorem 1.1.28 on page 13]. We choose H_m to be the set of all invertible linear mappings from $\mathrm{GF}(2^m)$ to $\mathrm{GF}(2^m)$, i.e., all functions of the form $x \mapsto ax + b$ where $a, b \in \mathrm{GF}(2^m)$ and $a \neq 0$ [4].[2] Such functions can be described by $2m$ bits, so that the input to G can be described by $(2k+1)m = O(S \log T)$ bits. This meets the requirements on the randomness.

To reach the desired time and space bounds, we must be able to evaluate the functions in H_m much faster than the naïve bound of $O(m^2)$. Using fast Fourier transform multiplication techniques based on those of Schönhage and Strassen and exploiting the sparseness of the above irreducible polynomial, we can evaluate $h \in H_m$ in time $O(m \log m \log \log m)$ and space $O(m)$. These fast multiplication techniques can be combined with the extended Euclidean algorithm to invert $h \in H_m$ in time $O(m \log^2 m \log \log m)$ and space $O(m)$. See [11, Corollary 11.8 on page 319] for more details. We use these algorithms to output the blocks of $G_{m,k}(y, h_1, \ldots, h_k)$ recursively with small space overhead. Specifically, we define the procedure P_m which uses global registers containing $k \geq 0$, $y \in \{0,1\}^m$, and $h_1, h_2, \ldots, h_k \in H_m$ to perform the following steps:

[2]Excluding the non-invertible functions ($a = 0$) introduces a small bias. Although our family H_m is not perfectly two-universal, it is close enough for our purposes.

1. **If** $k > 0$:
2. $\quad k \leftarrow k - 1$;
3. $\quad$ Recursively **call** P_m;
4. $\quad y \leftarrow h_{k+1}(y)$;
5. $\quad$ Recursively **call** P_m;
6. $\quad y \leftarrow h_{k+1}^{-1}(y)$; $k \leftarrow k + 1$ and **return**.
7. **Else output** y and **return**.

The output of $P_m(k, y, h_1, \ldots, h_k)$ is exactly $G_{m,k}(y, h_1, \ldots, h_k)$. Evaluating P_m takes 2^k recursive calls, each accompanied by an evaluation of h_i or h_i^{-1}. Thus, the overall time complexity to output $G_{m,k}$ is $O(2^k \cdot m \log^2 m \log\log m)$. By replacing y by $h_k(y)$ instead of writing down $h_k(y)$ separately on the work tape, only a constant amount of space overhead is required for each level. Thus, the space requirement is $O(m + k)$. For the settings of G, $m = O(S)$ and $k = \log \frac{T}{S} + O(1)$, the time becomes $O(T \log^2 S \log\log S)$, while the space is $O(S + \log \frac{T}{S})$, which is $O(S)$ since $T \leq 2^S$ without loss of generality. This assumes that the hash functions can be accessed repeatedly, so there is an additional space cost of $O(S \log T)$ to copy the hash functions from the random tape to the work tape, bringing the space bound to $O(S \log T + S) = O(S \log T)$. However, if the simulation has two-way access to the random tape, this cost is avoided.

Our simulation runs M for $O(S)$ steps every time P_m outputs a block of G, for a total of T steps while using space S. Thus, the above time and space bounds for computing G hold for the simulation as a whole. $\square$

B Deterministic Amplification

We now prove Lemma 12, beginning with a brief discussion of some properties of the amplification given by Theorem 9. Let M be a randomized machine that runs in time T, space S, and uses R random bits. We may assume that M has error at most some small constant δ to be determined later, since this can be achieved with only a constant overhead from a machine with any error bounded away from 1/2. The amplified machine M' given by Theorem 9 interprets its random string r' of length $O(R)$ as an initial vertex in a Gabber-Galil graph followed by $O(R)$ edge labels (of constant size) specifying the edges on a walk in the graph. Formally, this graph is described as follows:

Definition (Gabber-Galil graph [10]). *The Gabber-Galil graph* $\mathrm{GG}(m)$ *is a graph with vertices* $\mathbb{Z}_m \times \mathbb{Z}_m$ *of degree 5 where the vertices adjacent to* $(x, y) \in \mathbb{Z}_m \times \mathbb{Z}_m$ *are the five pairs* (x', y') *obtained from the matrix multiplication* $[x', y', 1]^T = A_i[x, y, 1]^T$ *(over* $\mathbb{Z}_m$*) for* $1 \leq i \leq 5$*, where*

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, A_4 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We now state a useful lemma which says that the vertex at the end of a path of length p in $\mathrm{GG}(2^k)$ can be found in time quasi-linear in p and k, an improvement on the naïve bound of $O(pk)$. We leave the proof for later.

Lemma 23. *Given a vertex (x, y) of the graph* $\mathrm{GG}(2^k)$ *and a path π of length p indicated by edge labels $(e_1, e_2, \ldots, e_p)$, where $1 \leq e_i \leq 5$ for $1 \leq i \leq p$, the vertex connected to (x, y) by π can be determined in time $O(m \operatorname{polylog}(m))$ and space $O(m)$ where $m = k + p$.*

For the purposes of Theorem 9, the vertices of the Gabber-Galil graph must be described by R bits corresponding to the possible random strings for M. To this end, we choose $\mathrm{GG}(2^{R/2})$. Given input x and random string r', M' proceeds by deterministically carrying out the walk of length $p = O(R)$ on $\mathrm{GG}(2^{R/2})$ indicated by r'. Every β steps, where β is some constant, M' simulates M on input x and random string corresponding to the label of the current vertex on the walk. M' accepts if a majority of these trials accept. As each edge relation can be computed by simple arithmetic in time $O(R)$ and space $O(R)$, and running M requires time T and space S, M' runs in time $O(R^2 + RT) = O(RT)$ and space $O(R + S)$. Cohen and Wigderson [5] and Impagliazzo and Zuckerman [13] show that when β is large enough and M has error smaller than some constant δ, the trials specified by a randomly chosen r' are close enough to uniform so that M' only errs with probability 2^{-R}. This establishes Theorem 9.

We now prove Lemma 12, giving a more time-efficient manner to determine if M' accepts on x and r' at the cost of using alternations.

Proof of Lemma 12. To arrive at Lemma 12, we show how to use alternations to verify if there is a majority of trials on the walk given by the random string r' where M accepts, in such a way that the final deterministic phase only needs to simulate M once. Specifically, given input x and random string r', we can express the acceptance condition of M' as

$$(\exists Z \subseteq \{1, 2, \ldots, R'\}, |Z| = \lceil R'/2 \rceil)(\forall i \in Z) M(x, r_i) \text{ accepts}, \tag{47}$$

where R' is the number of trials of M specified by r', and r_i is the random string produced for the i^{th} trial. Observe that this describes a Σ_2-computation which accepts if and only if M' accepts. The initial existential phase guesses the characteristic string of the set Z consisting of $\lceil R'/2 \rceil$ indices of the R' trials, for a total of $R' = O(R)$ bits. The universal phase guesses $\log R' = O(\log R)$ bits to determine the index of a trial to verify. The final deterministic stage must first determine r_i and then run M on input x and random string r_i. Once the former has been computed, the latter task takes time T and space S. Since r_i corresponds to the label of the $(\beta i)^{\text{th}}$ vertex on the walk in $\mathrm{GG}(2^{R/2})$ specified by r', Lemma 23 shows that r_i can be computed in time $O(R \operatorname{polylog}(R))$. Therefore, the final deterministic stage takes time $O(T + R \operatorname{polylog}(R))$ and space $O(R + S)$. All told, we have shown that (47) is a computation in

$$\exists^R \forall^{\log R} \mathrm{DTISP}[T + R \operatorname{polylog}(R), R + S]$$

which accepts if and only if M' accepts. This completes the proof. $\square$

All that remains is to establish Lemma 23, which follows from a divide-and-conquer strategy to efficiently evaluate a product of p matrices A_i.

Proof of Lemma 23. Throughout this proof, we use the fact that multiplication of b bit integers can be done in time $O(b \operatorname{polylog}(b))$ and space $O(b)$. This follows from the fast Fourier transform techniques of Schönhage and Strassen [11, Theorem 8.24 on page 240]. Let

$$A \doteq A_{e_p} A_{e_{p-1}} \cdots A_{e_1}.$$

Then the vertex (x', y') connected to (x, y) by π satisfies $[x', y', 1]^T = A[x, y, 1]^T \bmod 2^k$. Therefore, computing (x', y') reduces to computing A and multiplying by the vector $[x, y, 1]^T$ modulo 2^k.

We accomplish the latter with a divide-and-conquer strategy. Namely, we split the product approximately in half and recursively compute the subproducts $A_{e_p} A_{e_{p-1}} \cdots A_{e_{\lfloor p/2 \rfloor + 1}} \doteq B$ and $A_{e_{\lfloor p/2 \rfloor}} A_{e_{\lfloor p/2 \rfloor - 1}} \cdots A_{e_1} \doteq C$. It can be shown by induction that any product of p matrices A_i has entries bounded by 2^{p-1}, since each column of any matrix A_i has at most two non-zero entries, which are ones. This shows that once B and C are computed, A can be computed as $A = BC$ by $O(1)$ multiplications and additions of integers of bit length $p/2$, so we require time $O(p \text{ polylog}(p))$ in addition to the recursive calls. Thus, by following this strategy, we can see that at each recursive call to compute the product of q matrices, we solve two subproblems of size $q/2$ and do an additional amount of work which is quasi-linear in q and uses space $O(q)$. Thus, A can be computed in total time $O(p \operatorname{polylog}(p))$ and space $O(p)$.

By our observation on the size of the product matrix entries, we also know that the entries of the matrix A are at most 2^{p-1}. Therefore, computing the product $A[x, y, 1]^T$ and reducing modulo 2^k can be done in time $O(m \operatorname{polylog}(m))$ and space $O(m)$ as desired. ☐

We point out that another natural way to arrive at Theorem 13 is to use expander walks to generate the shift vectors for Lautemann's simulation in lieu of amplifying the confidence of the simulated algorithm as above. While the effect of the latter is to reduce the *number* of shift vectors needed, the former allows a large number of "good" shifts to be described by *very few bits*. Briefly, the hitting property of expanders guarantees that the shifts satisfy the needed property (i.e., the shifts of the accepting set cover the entire set of random strings) with approximately the same probability when they are chosen by a random walk on a Gabber-Galil graph as when they are chosen independently. Thus, we can generate a set of $O(R)$ good shifts in the initial stage of the simulation with only $O(R)$ bits. Furthermore, each shift can be computed efficiently by Lemma 23, so we can avoid the $O(R)$ blowup in the running time of the final deterministic stage by using an additional alternation to verify that M accepts on some shift. This approach leads to a simulation that matches the parameters of Theorem 13.