Time-Space Lower Bounds for Satisfiability*

Lance Fortnow† University of Chicago

Richard Lipton‡ Georgia Institute of Technology

Dieter van Melkebeek§ University of Wisconsin

Anastasios Viglas¶ University of Sydney

August 1, 2005

Abstract

We establish the first polynomial time-space lower bounds for satisfiability on general models of computation. We show that for any constant c less than the golden ratio there exists a positive constant d such that no deterministic random-access Turing machine can solve satisfiability in time n^c and space n^d, where d approaches 1 when c does. On conondeterministic instead of deterministic machines, we prove the same for any constant c less than $\sqrt{2}$.

Our lower bounds apply to nondeterministic linear time and almost all natural NP-complete problems known. In fact, they even apply to the class of languages that can be solved on a nondeterministic machine in linear time and space $n^{1/c}$.

Our proofs follow the paradigm of indirect diagonalization. We also use that paradigm to prove time-space lower bounds for languages higher up in the polynomial-time hierarchy.

*This paper combines extended abstracts that appeared in the *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science* [LV99] and in the *Proceedings of the 15th IEEE Conference on Computational Complexity* [FvM00].

†Research performed while at the NEC Research Institute.

‡Research performed while at Princeton University.

§Research partly performed while at DIMACS and the Institute for Advanced Study and partly supported by NSF Career award CCR-0133693.

¶Research performed while at Princeton University.

1 Introduction

Proving lower bounds remains the most difficult of tasks in computational complexity theory. Even for satisfiability, the seminal NP-complete problem of deciding whether a given propositional formula has at least one satisfying assignment, little is known. While we expect satisfiability to take exponential time in the worst case and to require a linear exponent in the number of variables of the formula, we do not even know how to rule out the existence of a linear-time algorithm on a random-access Turing machine. Obviously, linear time is needed since we have to look at the entire formula in the worst case. To date no better time lower bound than the trivial one is known on unrestricted random-access machines.

However, if we limit the amount of work space a machine solving satisfiability is allowed to use then we *can* establish nontrivial lower bounds on its running time. Fortnow [For00b] established a slightly super-linear time lower bounds for such machines: For any positive constant ϵ, satisfiability cannot be solved on a deterministic random-access Turing machine in time $n^{1+o(1)}$ and space $n^{1-\epsilon}$. In this work, we improve Fortnow's result and obtain a polynomial lower bound of constant degree larger than 1 for the same type of machines; the degree approaches the golden ratio $\phi \approx 1.618$ when the space bound becomes subpolynomial. More precisely, we obtain the following result.

Theorem 1.1 *Let $\phi \doteq (\sqrt{5}+1)/2$ denote the golden ratio. For any constant $c < \phi$ there exists a positive constant d such that satisfiability cannot be solved on a deterministic random-access Turing machine in time n^c and space n^d. Moreover, d approaches 1 from below when c approaches 1 from above.*

Deterministic time-space lower bounds for satisfiability relate to the P-versus-NP problem. Similarly, in the context of the NP-versus-coNP problem, one can establish time-space lower bounds for satisfiability on conondeterministic machines. In fact, Fortnow's lower bound also holds for conondeterministic machines [For00b]. Our techniques allow us to improve that lower bound from slightly superlinear to a polynomial of constant degree larger than 1, albeit a smaller degree than in the deterministic case.

Theorem 1.2 *For any constant $c < \sqrt{2}$ there exists a positive constant d such that satisfiability cannot be solved on a conondeterministic random-access Turing machine in time n^c and space n^d. Moreover, d approaches 1 from below when c approaches 1 from above.*

Our results as well as the earlier ones by Fortnow exploit the tight relationship between satisfiability and nondeterministic linear time. Satisfiability is complete for nondeterministic quasi-linear time under reductions that use quasi-linear time and logarithmic space [Coo88]. Time-space lower bounds for satisfiability and for nondeterministic linear time are equivalent up to polylogarithmic factors. The equivalence holds not just for satisfiability but for all natural NP-complete problems we know of. Thus, our lower bounds for satisfiability actually apply to all these problems.

Our proofs can be characterized as indirect diagonalization arguments. By way of contradiction, we assume we can simulate nondeterministic linear time efficiently on deterministic (in the case of Theorem 1.1) or conondeterministic (in the case of Theorem 1.2) machines. We view that hypothesis as an unlikely inclusion of complexity classes and use it to derive more and more unlikely inclusions of complexity classes up to the point where we reach a contradiction with a direct diagonalization result. Kannan [Kan84] used a similar approach in his investigation of the relationship between nondeterministic and deterministic linear time. His results do not imply lower bounds for satisfiability but we will cast his argument in such a way that we can use it as a starting point for our exposition.

We can get lower bounds for nondeterministic linear time even if we restrict the amount of space the nondeterministic machine can use. For example, we show that for every constant c less than the golden ratio there exists a positive constant d and a language computable by a nondeterministic machine in linear time and $n^{1/c}$ space that cannot be computed by deterministic machines that run in time n^c and space n^d. A similar lower bound for conondeterministic instead of deterministic simulations holds for every constant c less than $\sqrt{2}$. These are instantiations of more general tradeoffs between the various time and space parameters. They constitute the first nontrivial time-space lower bounds for nondeterministic linear-time computations with sublinear space.

Our approach applies to languages higher up in the polynomial-time hierarchy as well. We show that $\Sigma_\ell\mathsf{TIME}[n]$ cannot be solved in deterministic time n^c and space n^d for $c < \ell$ and some positive constant d (depending on c).

1.1 Organization

Section 2 contains some preliminaries, including a description of the machine model we focus on in our proofs, the constructibility conventions we use throughout the paper, and the direct diagonalization results we need.

In Section 3, we argue that time-space lower bounds for satisfiability and for nondeterministic linear time are equivalent up to polylogarithmic factors and that the same holds for all natural NP-complete problems we know of. Whereas in this section we have stated our main results in terms of satisfiability, in the rest of the paper we will think in terms of nondeterministic linear time.

Section 4 describes the proof paradigm of indirect diagonalization in more detail. It provides the structure and basic ingredients for all our arguments.

We derive our lower bounds for nondeterministic linear time on deterministic machines in Section 5. We start out by casting Kannan's result about nondeterministic versus deterministic linear time as an indirect diagonalization argument. We then develop our indirect diagonalization approach and derive explicit relationships between the constants c and d in the time and space bounds of Theorem 1.1, as well as further strengthenings. In Section 6, we do the same for our lower bounds for nondeterministic linear time on conondeterministic machines as stated in Theorem 1.2. Section 7 contains the application of our approach to complexity classes other than nondeterministic linear time.

Finally, we discuss some related subsequent work and directions for further research in Section 8.

2 Preliminaries

In this section, we describe our machine model, the constructibility conventions we assume in the rest of the paper, and the direct diagonalization results we need. We refer the reader to the textbooks by Balcázar, Díaz and Gabarró [BDG95], and by Papadimitriou [Pap94] for general background on computational complexity and for notation.

2.1 Machine model

Up to polylogarithmic factors, our results are robust with respect to the choice of random-access machine model. Our arguments work for all models we know; for some models extra polylogarithmic factors arise in the analysis due to simulations of machines within the model. In this paper, we choose the particular model below. It has the advantage that simulations can be done with only constant factor overheads in time and space, which keeps the analysis clean.

We use random-access Turing machines with an arbitrary number of tapes. A machine can have two types of tapes: non-index tapes and index tapes. Every non-index tape T except the output tape has an associated index tape I. The machine can move the head on T in one step to the position indexed by the contents of I. The contents of I is erased in such a step.

The input tape is read-only and has an associated index tape; the output tape is write-only, has no index-tape, and is one-directional. The input and output tapes do not count towards the space usage of the machine. Non-index tapes contribute the largest position ever read (indexed) to the space usage.

Note that according to our definition the space can be exponential in the running time. However, by using an appropriate data structure to store the contents of the tape cells accessed, we can prevent the space from being larger than the running time without blowing up the running time by more than a polylogarithmic factor.

A *configuration* of a machine M consists of the internal state of M, the contents of the work and index tapes, and the head positions on the index tapes. We use the notation $C \vdash^{\tau}_{M,x} C'$ to denote that machine M on input x can go from configuration C to configuration C' in τ steps. The *computation tableau* of M on input x is a representation of the entire computation. It consists of a table in which successive rows describe the successive configurations of M on input x, starting from the initial configuration of M. If M runs in time t and space s, each configuration has size $O(s)$ and the computation tableau contains at most t rows.

Based on our machine model, we define *complexity classes* in the standard way. Apart from the usual acronyms, we also employ the following notation: For functions t and s, $\mathsf{DTISP}[t,s]$ denotes the class of languages that can be accepted by a deterministic machine running in time $O(t)$ and space $O(s)$, and $\mathsf{NTISP}[t,s]$ denotes the corresponding nondeterministic class. $\Sigma_\ell\mathsf{TIME}[t]$ stands for the class of languages computed by Σ_ℓ-machines that run in time $O(t)$. Using Π_ℓ-machines instead of Σ_ℓ-machines yields the class $\Pi_\ell\mathsf{TIME}[t]$.

2.2 Constructibility Issues

Unless stated otherwise, a function f will always denote a function from the natural numbers to the natural numbers which is at least logarithmic, nondecreasing and does not grow too fast in the sense that $f(O(n)) \subseteq O(f(n))$. A sublinear function that acts as a time bound is in principle allowed. For example, the length of the input can be determined in logarithmic time in our computation model. Typically, though, a time bound has to be at least linear for our results to be meaningful. For example, several of our theorems have hypotheses of the form $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t,s]$ or $\mathsf{NTIME}[n] \subseteq \mathsf{coNTISP}[t,s]$; for sublinear t, these hypotheses trivially fail and therefore the underlying theorem vacuously holds. We use the term *subpolynomial* for a function f such that $f(n) = n^{o(1)}$, i.e., $f(n) = O(n^\epsilon)$ for every positive constant ϵ.

Regarding constructibility, let f, t, and s be functions. We say that f is (t,s) time-space constructible if there exists a deterministic machine that outputs $1^{f(|x|)}$ on input x and runs in time $O(t)$ and space $O(s)$. We call f constructible if f is (f, f) time-space constructible.

Constructibility conditions are needed when we do time and/or space bounded simulations, as in padding results like the following. We include a proof sketch for completeness.

Theorem 2.1 *Let f, t, s, s', and σ be functions such that $f(n) \geqslant n+1$ and is (f, s') time-space constructible and $s' = O(\sigma)$. Then*

$$\mathsf{NTISP}[n, \sigma] \subseteq \mathsf{coNTISP}[t, s]$$

implies

$$\mathsf{NTISP}[f, \sigma(f)] \subseteq \mathsf{coNTISP}[t(f) + f, s(f) + s'].$$

Proof: Let L be a language that is accepted by a nondeterministic machine M that runs in time $O(f)$ and space $O(\sigma(f))$.

Consider the padded language $L' = \{x10^{f(|x|)-|x|-1} \mid x \in L\}$. The language L' is accepted by a nondeterministic machine that acts as follows on an input y of length N. First, the machine verifies that y is of the form $y = x10^k$ for some string x and integer k, determines the length n of x, stores n in binary, and verifies that $f(n) = N$. The time-space constructibility of f allows us to verify the latter condition in time linear in N and space $O(\log N + \max_{0 \leqslant n \leqslant N} s'(n)) = O(\sigma(N))$. Second, we run M on input x, which takes time $f(n)$ and space $\sigma(f(n))$. Overall, the resulting nondeterministic machine for L' runs in time $O(N)$ and space $O(\sigma(N))$. By our hypothesis, there also exists a conondeterministic machine M' that accepts L' and runs in time $O(t(N))$ and space $O(s(N))$.

We then construct the following conondeterministic machine accepting L. On input x of length n, we use the time-space constructibility of f to compute $N = f(n)$ and write n and N down in binary. This step takes time $O(f(n))$ and space $O(s'(n) + \log f(n))$. Next, we simulate a run of M' on input $y = x10^{f(n)-n-1}$. We do so without storing y explicitly, using the input x, the values of n and N in memory, and comparing indices on the fly. The second step takes time $O(t(f(n)))$ and space $O(s(f(n)))$, resulting in overall requirements of $O(t(f(n)) + f(n))$ for time and $O(s(f(n)) + s'(n) + \log f(n)) = O(s(f(n)) + s'(n))$ for space. $\square$

We will not explicitly state the constructibility conditions outside of Section 2. We tacitly assume that the bounds in the general formulations of our results satisfy these requirements. The bounds we use in our concrete results will be polynomials, i.e., functions of the form n^c for some positive constant c; polynomials are sufficiently smooth and meet all the above conditions.

2.3 Diagonalization results

Most of our indirect diagonalization arguments use the following straightforward direct diagonalization result. It states that, for a fixed number of alternations, switching from universal to existential initial states and allotting a little bit more time allows us to do something that we could not do before. We include a proof sketch for completeness.

Theorem 2.2 *Let ℓ be a positive integer and t a time constructible function. Then*

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

Proof: Fix a positive integer ℓ. Paul, Prauß, and Reischuk [PPR80] show, for the multitape Turing machine model, that every Σ_ℓ-machine running in time τ with an arbitrary number of tapes can be simulated in time $O(\tau)$ by a Σ_ℓ-machine with a fixed number of tapes. The same holds for our model of computation.

Let M denote the Σ_ℓ-machine that takes an input x and runs the above simulation of the Σ_ℓ-machine described by x on input x; clock M such that it runs in time t. The language L defined by M lies in $\Sigma_\ell\mathsf{TIME}[t]$.

Consider any Π_ℓ-machine N that runs in time $o(t)$. There are infinitely many strings x that describe a Σ_ℓ-machine that does the opposite of N and runs in the same time as N. For large enough strings x in that sequence, M will finish its computation on input x before the clock kicks in, and therefore do the opposite of what N does on that input. Since M accepts L, N cannot accept L. Thus, L is not in $\Pi_\ell\mathsf{TIME}[o(t)]$. □

Except in Section 7, we will apply Theorem 2.2 with $\ell = 1$, i.e., we will use the fact that $\mathsf{NTIME}[t] \not\subseteq \mathsf{coNTIME}[o(t)]$.

A similar result to Theorem 2.2 also holds in the time-space bounded setting. We will only use the instance for $\ell = 1$:

Theorem 2.3 *Let t and s be functions that are (t, s) time-space constructible. Then*

$$\mathsf{NTISP}[t, s] \not\subseteq \mathsf{coNTISP}[o(t), o(s)].$$

Proof: Fortnow and Lund [FL93] argue that the result of Paul, Prauß, and Reischuk for multitape Turing machines also holds in the time-space bounded setting. Their proof carries over for our model of computation. The rest of the argument is the same as in the proof of Theorem 2.2. □

One of our results (part of Theorem 6.2) relies on a more intricate direct diagonalization result, namely the nondeterministic time hierarchy theorem.

Theorem 2.4 ([SFM78, Ž83]) *Let t_1 and t_2 be functions with t_2 time constructible. If $t_1(n+1) \in o(t_2(n))$ then*

$$\mathsf{NTIME}[t_1] \subsetneq \mathsf{NTIME}[t_2].$$

3 Satisfiability versus Nondeterministic Linear Time

This section discusses the tight connection between satisfiability and nondeterministic linear time as far as time-space lower bounds are concerned.

Cook's Theorem states that satisfiability (SAT) is NP-complete. Gurevich and Shelah [GS89] show that, in fact, satisfiability is complete for nondeterministic quasi-linear time under quasi-linear reductions. Recall that "quasi-linear" means $O(n \log^{O(1)} n)$. Therefore, proving time lower bounds for satisfiability and for nondeterministic linear time are equivalent up to polylogarithmic factors.

We are interested in simultaneous time and space bounds, though. Since satisfiability lies in nondeterministic quasi-linear time, time-space lower bounds for satisfiability also hold for nondeterministic linear time modulo polylogarithmic factors. The converse is also true but does not immediately follow from the completeness result by Gurevich and Shelah. The problem is that we do not have the space to store the result of the reduction of a problem in nondeterministic quasi-linear time to satisfiability, nor the time to redo the whole reduction each time we need a piece of it. The way around it is to construct a reduction each bit of which can be computed on the fly in polylogarithmic time using logarithmic work space. This leads to the following result.

Theorem 3.1 *There exists a constant r such that the following holds. If*

$$\mathrm{SAT} \in \mathsf{DTISP}[t, s],$$

then

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t(n \log^r n) \cdot \log^r n, s(n \log^r n)].$$

We include a detailed proof of Theorem 3.1 in Section 3.1.

Theorem 3.1 allows us to translate time-space lower bounds for nondeterministic linear time into the same time-space lower bounds for satisfiability up to polylogarithmic factors. In particular, for polynomial bounds we obtain:

Corollary 3.2 *Let c and d be constants. If*

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^d],$$

then for any constants $c' < c$ and $d' < d$

$$\mathrm{SAT} \notin \mathsf{DTISP}[n^{c'}, n^{d'}].$$

The simple reductions to satisfiability that underly Theorem 3.1 also exist to all of the standard natural NP-complete problems. In fact, to the best of the authors' knowledge, all known natural NP-complete problems in nondeterministic quasi-linear time share the latter property. Consequently, Theorem 3.1 and Corollary 3.2 hold if we replace satisfiability by any of these problems. They also hold if we replace DTISP by $\mathsf{coNTISP}$ in both hypothesis and conclusion.

From now on, our goal will be to obtain time-space lower bounds for nondeterministic linear time. Results for satisfiability and other NP-complete problems then follow from Theorem 3.1 or Corollary 3.2 and their variants.

3.1 Proof of Theorem 3.1

Gurevich and Shelah [GS89] show how to simulate nondeterministic random-access machines by multitape Turing machines. Their proof builds on Schnorr's result [Sch78] that one can sort in quasi-linear time on a nondeterministic multitape Turing machine.

Theorem 3.3 (Gurevich-Shelah [GS89]) *There exists a constant r such that every language that is accepted by a nondeterministic random-access Turing machine using time t is also accepted by a nondeterministic multitape Turing machine using time $O(t \log^r t)$.*

Because of Theorem 3.3 the following theorem finishes the proof of Theorem 3.1.

Theorem 3.4 *There exists a constant r such that the following holds. If*

$$\mathrm{SAT} \in \mathsf{DTISP}[t, s],$$

then every language accepted by a linear-time nondeterministic multitape Turing machine belongs to

$$\mathsf{DTISP}[t(n \log^2 n) \cdot \log^r n, s(n \log^2 n)].$$

Proof: Let M be a deterministic random-access Turing machine deciding satisfiability in time t and space s.

Consider an arbitrary language L accepted by a linear-time nondeterministic multitape Turing machine. Hennie and Stearns [HS66] show that there exists an oblivious 2-tape nondeterministic Turing machine that accepts L in time $O(n \log n)$. Cook [Coo88], building on work by Pippenger and Fischer [PF79], uses this result to construct for a given input x of length n, a Boolean formula ϕ such that $\phi \in \mathrm{SAT}$ iff $x \in L$. The formula ϕ has size $m = dn \log^2 n$ for some constant d depending on L (the additional factor of $\log n$ over the Hennie-Stearns simulation comes from writing down the indices of the variables in the formula), only depends on n, and uses the bits x_i of the input x as well as some additional Boolean variables y. More precisely, ϕ is of the form $(\wedge_{i=1}^{n} x_i = y_i) \wedge \psi$ where ψ only uses the variables y. Given a pointer to a bit of ψ, we can compute that bit in simultaneous time $O(\log^{r_1} m) = O(\log^{r_1} n)$ and space $O(\log m) = O(\log n)$ for some constant r_1 independent of L.

The following algorithm decides L in time $O(t(n \log^2 n) \cdot \log^r n)$ and space $O(s(n \log^2 n))$ on a deterministic random-access Turing machine for some constant r independent of L. We will simulate running M on input ϕ without storing ϕ in memory and without recomputing all of ϕ each time we have to access one of its bits. When given ϕ, running M on ϕ takes time $t(dn \log^2 n)$ and space $s(dn \log^2 n)$. Whenever M needs a bit from ψ, we compute that individual bit from scratch in time $O(\log^{r_1} n)$ and space $O(\log n)$, without moving the input tape head above x. During the periods when M is accessing the part of ϕ that depends on the input x, the easy structure of that part allows us to compute the bit of ϕ we need in time $O(\log^{r_2} n)$ and space $O(\log n)$ for some constant r_2 independent of L. This operation may require moving the input tape head above x. All together, we can simulate M on ϕ with a multiplicative time overhead of $O(\log^r n)$ for $r = \max(r_1, r_2)$, and an additive space overhead of $O(\log n) = O(s(n \log^2 n))$. □

4 Structure of Our Arguments

All time-space lower bounds for nondeterministic linear time to date share the same high-level structure, which can be characterized as indirect diagonalization. This section describes the paradigm and the main ingredients for its application in this paper. We refer to the survey paper by Fortnow [For00a] for other applications of indirect diagonalization.

Indirect diagonalization is a technique to separate complexity classes. In our case, we would like to obtain separations of the form $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[t, s]$ for some interesting values of the parameters t and s (t should be at least linear and s at least logarithmic).

The proofs go by contradiction and have the following outline:

1. We assume that the separation does not hold, i.e., we assume the unlikely inclusion $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t, s]$.

2. Next, using our hypothesis, we derive more and more unlikely inclusions of complexity classes.

3. Finally, we derive a contradiction with a direct diagonalization result.

We first describe the two techniques we will apply to derive more inclusions in step 2, namely trading alternations for time and trading time for alternations. Then we will see how to combine these techniques to obtain a contradiction, and thereby refute the hypothesis made in step 1.

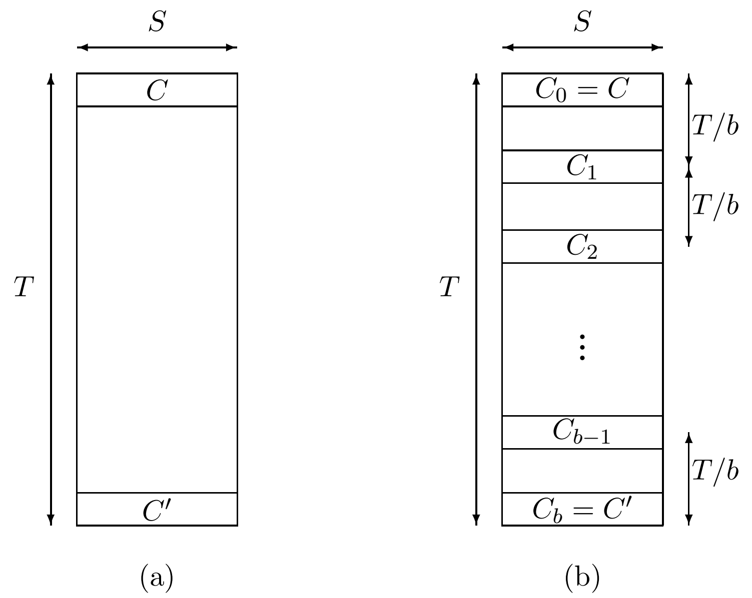

Figure 1: Tableaus of a computation using time T and space S

4.1 Trading Alternations for Time

Trading alternations for time means reducing the running time by allowing more alternations. We know how to do this in general for space-bounded computations using a divide-and-conquer strategy. The technique has been known for a long time and has been applied extensively in computational complexity, for example, in the proof of Savitch's theorem.

Suppose we have a deterministic machine M that runs in space S. We are given two configurations C and C' of M on an input x, and would like to know whether M goes from C to C' in T steps. One way to do this is to run the machine for T steps from configuration C and check whether we end up in configuration C'. In other words, we fill in the whole tableau in Figure 1(a) row by row.

Using the power of alternation, we can speed up this process as follows. We can break up the tableau into b equal blocks, guess the configurations $C_1, C_2, \ldots, C_{b-1}$ at the common borders of the blocks, treat each of the blocks i, $1 \leqslant i \leqslant b$, as a subtableau and verify that M on input x goes from configuration C_{i-1} to C_i in T/b steps. See Figure 1(b).

In terms of logical formulas, we are using the following property of configurations:

$$C \vdash^T C' \Leftrightarrow (\exists\, C_1, C_2, \ldots, C_{b-1})(\forall\, 1 \leqslant i \leqslant b)\, C_{i-1} \vdash^{T/b} C_i, \tag{1}$$

where $C_0 \doteq C$ and $C_b \doteq C'$. We can perform this process on a Σ_2-machine using time $O(bS)$ for guessing the $b-1$ intermediate configurations of size S each in the existential phase, time $O(\log b)$ to guess the block i we want to verify in the universal phase, and time $O(T/b)$ to deterministically run M for T/b steps to verify the ith block. Since the $O(\log b)$ term can be ignored, we obtain

$$\mathsf{DTISP}[T, S] \subseteq \Sigma_2\mathsf{TIME}[bS + T/b]. \tag{2}$$

The running time of the Σ_2-machine is minimized (up to a constant) by choosing $b = \sqrt{T/S}$, resulting in:

Lemma 4.1 *For any functions T and S,*

$$\mathsf{DTISP}[T,S] \subseteq \Sigma_2\mathsf{TIME}[\sqrt{TS}].$$

The final deterministic phase of our simulation consists of an easier instance of our original problem. Therefore, we can apply the divide-and-conquer strategy again, and again. Each application increases the number of alternations by 2. k recursive applications with block numbers $b_1, b_2, \ldots, b_k$, respectively, yield:

$$\mathsf{DTISP}[T,S] \subseteq \Sigma_{2k}\mathsf{TIME}[(\sum_i b_i)S + T/(\prod_i b_i)]. \tag{3}$$

The running time of the Σ_{2k}-machine is minimized (up to a constant) by picking the block numbers all equal to $(T/S)^{1/(k+1)}$. We obtain:

$$\mathsf{DTISP}[T,S] \subseteq \Sigma_{2k}\mathsf{TIME}[(TS^k)^{1/(k+1)}]. \tag{4}$$

We point out for later reference that minimizing the running time of the Σ_{2k}-machine may not be the best thing to do if this simulation is just an intermediate step in a derivation. In particular, in the proofs of Theorems 1.1 and 1.2 the block numbers are not all equal.

4.2 Trading Time for Alternations

The other tool we will use to derive more unlikely inclusions of complexity classes from our hypothesis $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t,s]$ consists of the opposite of what we just did. We will now see how we can trade time for alternations, i.e., how we can get rid of alternations by – moderately – increasing the running time.

In general, we only know how to remove one alternation at an exponential cost in running time. However, our hypothesis implies that $\mathsf{NTIME}[n]$ is included in $\mathsf{DTIME}[t]$. If t is small, this means that we can simulate nondeterminism deterministically and thus eliminate alternations at a moderate expense.

For example, we have the following:

Lemma 4.2 *Let t and τ be functions. If*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}[t]$$

then

$$\Sigma_2\mathsf{TIME}[\tau] \subseteq \Sigma_1\mathsf{TIME}[t(n+\tau)]. \tag{5}$$

Proof: Consider a Σ_2-machine running in time τ on an input x of length n. Its acceptance criterion can be written as

$$(\exists\, y_1 \in \{0,1\}^\tau)\ \underbrace{(\forall\, y_2 \in \{0,1\}^\tau)\, R(x, y_1, y_2)}_{(\alpha)}, \tag{6}$$

where R denotes a predicate computable in deterministic linear time. Part (α) of (6) defines a conondeterministic computation on input x and y_1. The running time is $O(\tau)$, which is linear in the input length $n+\tau$. Therefore, our hypothesis implies that we can transform (α) into a deterministic computation on input x and y_1 taking time $O(t(n+\tau))$. All together, (6) then describes a nondeterministic computation on input x of time complexity $O(\tau + t(n+\tau))$, which is

$O(t(n+\tau))$ since we can assume that $t(n) \leqslant n$ (if $t(n) < n$, the premise of the lemma trivially fails and therefore the lemma vacuously holds). $\square$

Note that (5) also follows from the weaker hypothesis $\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[t]$. Then ($\alpha$) in (6) can only be transformed into a nondeterministic instead of a deterministic computation running in time $O(t(n+\tau))$, but (6) as a whole still remains a nondeterministic computation taking $O(t(n+\tau))$ time. The same argument also works for a larger number of alternations.

Lemma 4.3 *Let $a \geqslant 2$ be an integer and t and τ functions. If*

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[t]$$

then

$$\Sigma_a\mathsf{TIME}[\tau] \subseteq \Sigma_{a-1}\mathsf{TIME}[t(n+\tau)].$$

In particular, in case t is of the form $t(n) = n^c$ for some constant c and τ is at least linear, we can eliminate an alternation at the cost of raising the running time to the power c.

4.3 Obtaining a Contradiction

So far we have seen techniques:

1. to trade alternations for time, and
2. to trade time for alternations.

What remains is to combine them in the right way so as to reduce both resources enough and obtain a contradiction with a direct diagonalization result.

The two most obvious ways of combining the techniques are to apply the first one and then the second one, or vice versa.

- Fortnow [For00b] first traded time for alternations, and then alternations for time. We refer the reader to a survey paper by Van Melkebeek [vM04] for this view of Fortnow's approach.
- Kannan [Kan84] did it the other way around. His approach forms the basis for our results and we will discuss it in the next section.

5 Lower Bounds for Nondeterministic Linear Time on Deterministic Machines

This section covers our time-space lower bounds for nondeterministic linear time on deterministic machines. Our starting point is an indirect diagonalization argument used by Kannan [Kan84]. We show how to modify his argument and add some more ingredients to obtain the golden ratio time lower bound for subpolynomial space bounds. We then derive our time lower bound for general space bounds, quantify the tradeoff in Theorem 1.1, and further strengthen it.

5.1 Kannan's Argument

Kannan [Kan84] investigates the relationship between deterministic time $O(t)$ and nondeterministic time $O(t)$ for various time bounds t, in particular for polynomials. In the case of linear t, he shows that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n, o(n)]$ using essentially[1] the following argument. We cast the argument in the indirect diagonalization paradigm presented at the beginning of Section 4.

Step 1 We assume by way of contradiction that

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n, o(n)]. \tag{7}$$

Step 2 Consider the class $\mathsf{DTISP}[\tau, o(\tau)]$ for some super-linear function τ. By first trading alternations for time as in Lemma 4.1 and then time for alternations as in Lemma 4.2, we obtain the following unlikely inclusion:

$$\mathsf{DTISP}[\tau, o(\tau)] \subseteq \Sigma_2\mathsf{TIME}[o(\tau)] \subseteq \mathsf{NTIME}[o(\tau)]. \tag{8}$$

The last inclusion follows from (5) since τ is super-linear. Note that Step 2 does not rely on our hypothesis (7).

Step 3 The hypothesis (7) padded to time τ, closure under complementation of DTISP classes, and (8) yield:

$$\mathsf{NTIME}[\tau] \subseteq \mathsf{DTISP}[\tau, o(\tau)] = \mathsf{coDTISP}[\tau, o(\tau)] \subseteq \mathsf{coNTIME}[o(\tau)].$$

This is a contradiction with Theorem 2.2 for, say, $\tau(n) = n^2$.

We point out that for multitape Turing machines without random access, Paul et al. [PPST83] use indirect diagonalization to separate nondeterministic linear time from deterministic linear time. However, their argument breaks down in case of random access. Kannan's argument works with random access but needs the additional space restriction of $o(n)$. Kannan uses his argument to derive other results about the relationship between $\mathsf{DTIME}[t]$ and $\mathsf{NTIME}[t]$ for nonlinear t. We do not discuss these results but move on to our modification of the argument.

5.2 The Road to The Golden Ratio

We want to rule out deterministic simulations of nondeterministic linear time that use more time but less space than in Kannan's original setting. For ease of exposition, we focus on subpolynomial space bounds first. We present the general analysis in Section 5.3. Thus, we would like to establish results of the form

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}], \tag{9}$$

where c is a constant larger than 1.

Let us run through the argument of Section 5.1 with the modified parameters.

Step 1 Assume that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$.

Step 2 For any function $\tau(n) \geqslant n^2$,

$$\mathsf{DTISP}[\tau, \tau^{o(1)}] \subseteq \Sigma_2\mathsf{TIME}[\tau^{\frac{1}{2}+o(1)}] \subseteq \mathsf{NTIME}[\tau^{\frac{c}{2}+o(1)}].$$

[1] Kannan uses the nondeterministic time hierarchy theorem instead of Theorem 2.2.

Step 3 For any function $\tau(n) \geqslant n^{2/c}$,

$$\mathsf{NTIME}[\tau] \subseteq \mathsf{DTISP}[\tau^c, \tau^{o(1)}] = \mathsf{coDTISP}[\tau^c, \tau^{o(1)}] \subseteq \mathsf{coNTIME}[\tau^{c^2/2+o(1)}].$$

We obtain a contradiction with Theorem 2.2 as long as $c^2/2 < 1$.

We conclude that (9) holds for any constant $c < \sqrt{2}$.

In order to establish (9) for constants $c \geqslant \sqrt{2}$, one might try to improve Step 2 by applying the divide-and-conquer strategy of Section 4.1 recursively. That is, we use more alternations, as in (3), to reduce the running time further and then remove them using Lemma 4.3 repeatedly. We obtain the following substitute for Step 2 by choosing the block numbers b_i in (3) optimally.

Lemma 5.1 *Suppose that*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}[n^c]$$

for some constant $c \geqslant 1$. Then for any function T and any positive integer k

$$\mathsf{DTISP}[T, T^{o(1)}] \subseteq \mathsf{NTIME}[T^{e_k+o(1)}]$$

provided $T^{e_k}(n) \geqslant n^{c^{2k-1}}$, where

$$\begin{aligned} e_1 &= c/2 \\ e_{k+1} &= c^2 e_k/(1+ce_k). \end{aligned}$$

The sequence $(e_k)_k$ converges monotonically to the positive fixed point of the transformation $e \to c^2 e/(1+ce)$, i.e., to $e_\infty \doteq c - \frac{1}{c}$.

Unfortunately[2], using Lemma 5.1 in Step 2 does not yield stronger results. Indeed, for k levels of recursion we obtain in Step 3 that for any sufficiently large polynomial τ

$$\mathsf{NTIME}[\tau] \subseteq \mathsf{DTISP}[\tau^c, \tau^{o(1)}] = \mathsf{coDTISP}[\tau^c, \tau^{o(1)}] \subseteq \mathsf{coNTIME}[\tau^{ce_k+o(1)}].$$

We reach a contradiction with Theorem 2.2 as long as $c \cdot e_k < 1$ for some positive integer k. Because of the monotonicity of the sequence $(e_k)_k$ we only have to check the starting point $e_1 = c/2$, which we already dealt with, and the limit value $e_\infty = c - \frac{1}{c}$. However,

$$c \cdot e_\infty < 1 \quad \Leftrightarrow \quad c^2 < 2 \quad \Leftrightarrow \quad c \cdot e_1 < 1.$$

In other words, recursion does not help. Each additional level of recursion allows us to further reduce the running time of the intermediate alternating machine. The latter also uses two more alternations, though. We have to eliminate these alternations subsequently, which involves, for each extra alternation, raising the running time of the simulation to the power c. Both effects even out.

However, we can achieve the same savings in the running time of the intermediate alternating machine with only half as many alternations. In order to do so, we exploit the closure under complementation of deterministic complexity classes, or equivalently, the following property of deterministic computations.

$$C \vdash^T C' \Leftrightarrow (\forall C'' \neq C')\, C \nvdash^T C''. \tag{10}$$

[2]For this reason, we omit the proof of Lemma 5.1. The interested reader can construct a proof along the lines of the proof of Lemma 5.3 but using one more application of Lemma 4.3. See also the proof of Lemma 6.1.

That is, a deterministic machine M goes from a configuration C to a configuration C' in T steps iff for every configuration C'' different from C', M cannot reach C'' from C in T steps. To verify the latter we use the divide-and-conquer strategy of Section 4.1. We replace the matrix of (10) by the negation of the right-hand side of (1) and rename C'' to C_b for convenience.

$$C \vdash^T C' \Leftrightarrow (\forall\, C_b \neq C')(\forall\, C_1, C_2, \ldots, C_{b-1})(\exists\, 1 \leqslant i \leqslant b)\, C_{i-1} \nvdash^{T/b} C_i, \tag{11}$$

where C_0 denotes C. In terms of the tableau of Figure 2, M reaches C' from C in T steps iff

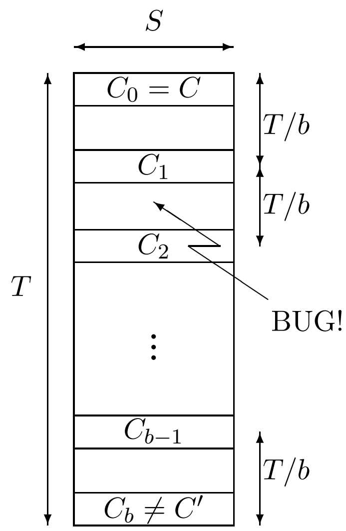

Figure 2: Saving alternations

the following holds. If we break up the tableau into b blocks then for every choice of intermediate configurations C_i, $1 \leqslant i \leqslant b-1$, and of a final configuration C_b other than C', there has to be a block i that cannot be completed in a legitimate way.

Applying this idea recursively amounts to replacing the matrix $C_{i-1} \nvdash^{T/b} C_i$ of the Π_2-formula (11) by a Σ_2-formula which is the negation of a formula of the same type as the whole right-hand side of (11). The existential quantifiers merge and the resulting formula is of type Π_3. In general, k recursive applications result in a Π_{k+1}-formula. If we denote the block numbers for the successive recursive applications by $b_1, b_2, \ldots, b_k$, we conclude in a similar way as in Section 4.1 that

$$\mathsf{DTISP}[T, S] \subseteq \mathsf{\Pi}_{k+1}\mathsf{TIME}[(\sum_i b_i)S + T/(\prod_i b_i)]. \tag{12}$$

So, we achieve the same speed-up as in (3) but with only half as many alternations. An improvement of Lemma 5.1 follows.

Lemma 5.2 *Suppose that*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}[n^c]$$

for some constant $c \geqslant 1$. Then for any function T and any integer $k \geqslant 0$

$$\mathsf{DTISP}[T, T^{o(1)}] \subseteq \mathsf{coNTIME}[T^{f_k + o(1)}]$$

provided $T^{f_k}(n) \geqslant n^{c^k}$*, where*

$$\begin{array}{rcl} f_0 & = & 1 \\ f_{k+1} & = & c \cdot f_k/(1+f_k). \end{array} \tag{13}$$

For future reference, we point out some properties of the sequence $(f_k)_k$, the first few terms of which are $f_0 = 1$, $f_1 = c/2$, $f_2 = c^2/(2+c)$, etc. The fact that the transformation $x \to x/(1+x)$ is monotone increasing for $x \geqslant 0$ implies that (i) for each $k \geqslant 1$, f_k is a monotone increasing function of c for $c \geqslant 1$, and (ii) for each fixed $c \geqslant 1$, the sequence $(f_k)_k$ monotonically converges to the fixed point of the transformation $f \to c \cdot f/(1+f)$, i.e., to $f_\infty \doteq c-1$. The sequence $(f_k)_k$ is decreasing iff $f_0 > f_1$, or equivalently, iff $c < 2$. An explicit expression for the sequence is:

$$f_k = \begin{cases} 1/(k+1) & \text{if } c = 1 \\ (c-1) \cdot \frac{1}{1-\frac{2-c}{c^k}} & \text{if } c > 1. \end{cases}$$

Since f_k has value $1/(k+1) \leqslant 1$ for $c = 1$ and monotonically grows unbounded for $k \geqslant 1$, the transformation $c \to 1/f_k$ has a unique fixed point $c \geqslant 1$. The fixed point depends on k and monotonically increases to the fixed point $c \geqslant 1$ of $c \to 1/f_\infty$, which is the golden ratio ϕ as ϕ satisfies $\phi(\phi - 1) = 1$.

We will prove a generalization of Lemma 5.2 in Section 5.3. Applying Lemma 5.2 in a similar way as Lemma 5.1, Step 3 reads

$$\mathsf{NTIME}[\tau] \subseteq \mathsf{DTISP}[\tau^c, \tau^{o(1)}] \subseteq \mathsf{coNTIME}[\tau^{cf_k+o(1)}],$$

for any integer $k \geqslant 0$ and sufficiently large polynomial τ. Thus, we obtain a contradiction with Theorem 2.2 as long as $c \cdot f_k < 1$ for some positive integer k. The latter is the case iff $c \cdot f_\infty = c(c-1) < 1$. We conclude that

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$$

for any constant $c < \phi$.

5.3 The General Lower Bound

We now derive our time-space lower bounds for nondeterministic linear time on deterministic machines in their most general form and establish Theorem 1.1 as well as some other results.

We start with a version of Lemma 5.2 for arbitrary space bounds. We also weaken its hypothesis somewhat, reflecting the fact that trading time for alternations only requires simulations of nondeterministic computations on conondeterministic (as opposed to deterministic) machines.

Lemma 5.3 *Suppose that*

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[n^c]$$

for some constant $c \geqslant 1$*. Then for any functions* T *and* S *and any integer* $k \geqslant 0$

$$\mathsf{DTISP}[T, S] \subseteq \mathsf{coNTIME}[(T \cdot S^k)^{f_k} + (n+S)^{c^k}],$$

where f_k *is given by (13).*

The proof of Lemma 5.3 essentially uses (12) and optimizes the number of blocks b_1, b_2, …, b_k at the successive levels of recursion. We obtained (4) by setting these numbers the same at every level. Intuitively, this isn't optimal because the subsequent trading of time for alternations raises the contributions of later blocks to the running time more often to the power c than the contributions of earlier blocks. Thus, later levels should have fewer blocks than earlier ones. The precise balance is determined in the next proof.

Proof of Lemma 5.3: We give a proof by induction on k. The base case $k = 0$ trivially holds.

For the induction step $k \to k+1$, since $\mathsf{DTISP}[T, S]$ is closed under complementation, it suffices to show that

$$\mathsf{DTISP}[T, S] \subseteq \mathsf{NTIME}[(T \cdot S^{k+1})^{f_{k+1}} + (n+S)^{c^{k+1}}] \tag{14}$$

assuming that statement of the lemma holds for k.

Consider a deterministic machine M that runs in time T and space S on an input x of length n. Let us analyze the simulation defined by (1) when we apply the induction hypothesis to the matrix:

$$\underbrace{(\exists\, C_1, C_2, \ldots, C_{b-1})\, \underbrace{(\forall\, 1 \leqslant i \leqslant b)\ \underbrace{C_{i-1} \vdash^{T/b} C_i}_{(\alpha)}}_{(\beta)}}_{(\gamma)}.$$

Part (α) corresponds to a deterministic computation on input x, C_{i-1}, and C_i with the following parameters:

$$\begin{aligned} \text{input size:}\quad & n + 2S \\ \text{running time:}\quad & O(T/b) \\ \text{space used:}\quad & O(S). \end{aligned}$$

By the induction hypothesis, we can turn (α) into a conondeterministic computation running in time

$$O((T/b \cdot S^k)^{f_k} + (n+S)^{c^k}).$$

Thus (β) becomes a conondeterministic computation on input x and $C_0, C_1, \ldots, C_b$ with the following parameters:

$$\begin{aligned} \text{input size:}\quad & n + (b+1)S \\ \text{running time:}\quad & O(\log b + (T/b \cdot S^k)^{f_k} + (n+S)^{c^k}). \end{aligned}$$

Along the lines of Section 4.2, the hypothesis of the lemma allows us to transform this conondeterministic computation into a nondeterministic one resulting in a nondeterministic simulation of (γ) taking time

$$O\left((b-1)S + \left(n + (b+1)S + \log b + (T/b \cdot S^k)^{f_k} + (n+S)^{c^k}\right)^c\right).$$

The latter expression simplifies to big O of

$$(bS + (T/b \cdot S^k)^{f_k})^c + (n+S)^{c^{k+1}}, \tag{15}$$

where the constant hidden in the big O depends on c and on k. Our goal is to minimize (15) up to constant factors by picking b appropriately. Note that the first term in (15) increases with b

whereas the second one decreases with b; the other terms are independent of b. Thus, without any constraints on b, (15) is minimized up to a factor of 2 by equating the first two terms. If the resulting value b^* is smaller than 1, the best we can do is setting $b = 1$. In that case, the first term dominates the second one and is dominated itself by the term independent of b. If b^* exceeds T, the resulting simulation becomes trivial since it runs in time more than $bS \geqslant T$. If b^* lies in the range $[1, T]$, rounding b^* to the nearest integer affects the value of the objective function by only a constant factor. Thus, in each case we can obtain a nondeterministic simulation that runs in time a constant factor times the value of (15) for $b = b^*$.

Solving $bS = (T/b \cdot S^k)^{f_k}$ for b results in the following expression for (15):

$$2 \cdot (T \cdot S^{k+1})^{cf_k/(1+f_k)} + (n + S)^{c^{k+1}}.$$

This establishes (14) since f_{k+1} is given by (13). □

Plugging in Lemma 5.3 into our indirect diagonalization approach yields the following general result.

Theorem 5.4 *For any constant $c \geqslant 1$ and functions t and s,*

$$\mathsf{NTIME}[n] \nsubseteq \mathsf{coNTIME}[n^c] \cap \mathsf{DTISP}[t, s] \tag{16}$$

if for some integer $k \geqslant 0$

$$(t \cdot s^k)^{f_k} + s^{c^k} = o(n), \tag{17}$$

where f_k is given by (13).

Proof: Assume that

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[n^c] \cap \mathsf{DTISP}[t, s] \tag{18}$$

for some integer $k \geqslant 0$ such that (17) holds. For any function $\tau(n) \geqslant n$, we have

$$\begin{aligned}
\mathsf{NTIME}[\tau] &\subseteq \mathsf{DTISP}[t(\tau), s(\tau)] \text{ (by padding (18))} \\
&\subseteq \mathsf{coNTIME}[(t(\tau) \cdot s(\tau)^k)^{f_k} + (n + s(\tau))^{c^k}] \text{ (by Lemma 5.3)} \\
&= \mathsf{coNTIME}[(t(\tau) \cdot s(\tau)^k)^{f_k} + s(\tau)^{c^k}] \text{ (provided } t(\tau(n))^{f_k} \geqslant n^{c^k}) \\
&\subseteq \mathsf{coNTIME}[o(\tau)] \text{ (by (17))},
\end{aligned}$$

which is a contradiction with Theorem 2.2. Note that we can assume without loss of generality that $t(n) \geqslant n$ so there exists a function τ such that $t(\tau(n))^{f_k} \geqslant n^{c^k}$, e.g., a polynomial of sufficiently large degree. □

One instantiation of Theorem 5.4 states that either $\mathsf{NTIME}[n]$ is not in logspace or else there exists a constant $c > 1$ such that $\mathsf{NTIME}[n]$ is not contained in $\mathsf{coNTIME}[n^c]$. Fortnow [For00b] already proved that instantiation. In general, Theorem 5.4 implies quantitative improvements over Fortnow's results but in this particular case the improvement is hidden in the statement, namely in the dependence of the constant c on the running time of the presumed logspace algorithm for nondeterministic linear time. Say that running time is $O(n^a)$ for some constant a. A careful analysis of Fortnow's technique rules out any c such that $c < \sqrt[2k-1]{(k+1)/a}$ for some integer $k \geqslant 1$. Theorem 5.4 rules out any $c < 1 + 1/a$, which is a stronger statement.

We can use Theorem 5.4 to obtain the following lower bound on the time-space product of any deterministic simulation of nondeterministic linear time.

Corollary 5.5 *If* $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t,s]$ *then* $ts = \Omega(n^{\sqrt{2}})$.

Proof: Assume that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t,s]$. Let $c \geqslant 1$ be any constant and $k \geqslant 0$ an integer. We consider three cases:

- If $t \neq O(n^c)$ then $ts = \Omega(n^c)$.
- If $s^{c^k} = \Omega(n)$ then $ts = \Omega(n^{1+1/c^k})$ since without loss of generality $t(n) \geqslant n$.
- If $t = O(n^c)$ and $s^{c^k} = o(n)$ then by Theorem 5.4 $ts^k = \Omega(n^{1/f_k})$ and therefore $ts = \Omega(n^{1/(kf_k)}t^{1-1/k})$ and thus $ts = \Omega(n^{1-1/k+1/(kf_k)})$.

We conclude that $ts = \Omega(n^a)$ for $a = \max_{c\geqslant 1} \min(c, 1 + 1/c^k, 1 - 1/k + 1/(kf_k))$. The latter expression turns out to be maximized for $k = 1$, yielding $a = \max_{c\geqslant 1} \min(c, 1 + 1/c, 2/c) = \max_{c\geqslant 1} \min(c, 2/c) = \sqrt{2}$. □

Theorem 5.4 naturally gives lower bounds on ts^k for $\mathsf{DTISP}[t,s]$-simulations of nondeterministic linear time for values of k larger than 1. As an example, we state the result for $k = 2$.

Corollary 5.6 *If* $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t,s]$ *then* $ts^2 = \Omega(n^a)$*, where* a *is the positive solution of* $a(a^2 - 1) = 2$*, about 1.521.*

Proof: The same approach as in the proof of Corollary 5.5 leads to optimal results for $k = 2$. It results in the lower bound $ts^2 = \Omega(n^a)$ for $a = \max_{c\geqslant 1} \min(c, 1 + 2/c^2, (2 + c)/c^2)$. Since $(2 + c)/c^2 \leqslant 1 + 2/c^2$ for $c \geqslant 1$, a is the positive solution to $c = (2 + c)/c^2$. □

Similar statements follow from Theorem 5.4 for larger values of k. Letting k grow to infinity leads to the result we obtained at the end of Section 5.2, namely that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ for any constant $c < \phi$. In fact, we can show the stronger statement given in Theorem 1.1, the quantitative proof of which we embark on next.

We can simplify the statement of the general lower bound of Theorem 5.4 as follows in the case where $t(n) = O(n^c)$.

Theorem 5.7 *For any constant* $c \geqslant 1$ *and integer* $k \geqslant 1$*,*

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, o(n^d)],$$

where

$$d = \min((1/f_k - c)/k, 1/c^k) \tag{19}$$

and f_k *is defined by (13).*

We observe that $d = 1$ for $c = 1$, giving Kannan's result from Section 5.1. For a given value of k, the right-hand side of (19) is positive as long as c is less than the fixed point of $c \to 1/f_k$ for $c \geqslant 1$. Note also that $d < 1$ for $c > 1$.

Proof of Theorem 5.7: Theorem 5.4 with $t = n^c$ and $s = o(n^d)$ gives us the separation as long as $n^{(c+kd)f_k} + n^{dc^k} = O(n)$. The latter condition is equivalent to d being at most the right-hand side of (19). □

Which value of k realizes the maximum of (19) depends on the value of c. The bound for $k = 2$ turns out to be always at least as good as the one for $k = 1$. We explicitly state the bound obtained for $k = 2$.

Corollary 5.8 *For any constant $c \geqslant 1$,*

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

Note that Corollary 5.8 yields nontrivial statements as long as $\frac{c+2}{c^2} - c > 0$, or equivalently, c is less than the fixed point of $c \to 1/f_2$. This condition works out to $c(c^2-1) < 2$, or approximately, $c \leqslant 1.521$.

Since the fixed point of $c \to 1/f_k$ converges to the golden ratio for $k \to \infty$, Theorem 5.7 gives interesting results for c up to the golden ratio, as stated in Theorem 1.1.

Proof of Theorem 1.1: The statement trivially holds for $c < 1$. For $c \geqslant 1$ the proof follows from Theorem 5.7. The expression (19) is positive as long as there exists some k such that $c \cdot f_k < 1$. Since f_k monotonically decreases to $f_\infty = c - 1$, the latter condition is equivalent to $c(c-1) < 1$, i.e., to $c < \phi$.

By considering, for example, $k = 1$, one can see that the value of d approaches 1 when c does. □

Theorem 5.7 also lets us improve the lower bound on the time-space product of Corollary 5.5 in the case of polynomial time bounds.

Corollary 5.9 *If $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t, s]$ for some t of the form $t = n^c$ then $ts = \Omega(n^{1.573})$.*

Proof: A trivial bound is $ts = \Omega(n^c)$. Theorem 5.7 tells us that for every integer $k \geqslant 1$, $ts = \Omega(n^a)$ where $a = c + \min((1/f_k - c)/k, 1/c^k)$. This gives us a better bound than the trivial one as long as $c \cdot f_k < 1$. The minimum over $c \geqslant 1$ of the resulting compound bound is maximized for $k = 4$, namely at $c \approx 1.4572$, resulting in $a \approx 1.5738$. □

6 Lower Bounds for Nondeterministic Linear Time on Cononde-terministic Machines

Our indirect diagonalization approach lends itself to establishing time-space lower bounds for nondeterministic linear time on conondeterministic machines as well. The results are somewhat weaker than for deterministic machines. We derive them in this section.

As the first step we assume by way of contradiction that

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[n^c] \cap \mathsf{coNTISP}[t, s] \tag{20}$$

for some constant c. The two techniques we developed in Sections 4.1 and 4.2 to derive more unlikely inclusions apply to the nondeterministic setting, too.

In particular, the divide-and-conquer strategy (1) works for nondeterministic machines M. Breaking up the computation into b equal blocks leads to the inclusion

$$\mathsf{NTISP}[T, S] \subseteq \Sigma_3\mathsf{TIME}[bS + T/b].$$

Note that we have one more alternation than in (2) because the matrix predicate on the right-hand side of (1) becomes Σ_1 in case of nondeterministic machines M. This is one reason why we obtain weaker results.

As in Section 4.1, we can apply the divide-and-conquer strategy recursively. Corresponding to (3) we obtain

$$\mathsf{NTISP}[T,S] \subseteq \Sigma_{2k+1}\mathsf{TIME}[(\sum_i b_i)S + T/(\prod_i b_i)].$$

In Section 5.2, we showed how to achieve the same speed-up as in (3) using only half the number of alternations. However, (12) does not carry over to the nondeterministic setting because NTIME classes are not known to be closed under complementation. In terms of the exposition in Section 5.2, property (10) fails for a generic nondeterministic machine because nondeterministic machines may be able to reach more than one configuration on a given input in T steps. This is the other reason why we cannot quite match the deterministic results of the previous section.

There are no complications as far as trading time for alternations is concerned. We observed in Section 4.2 and already used in Section 5.3 that (5) follows from the hypothesis that $\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[t]$.

Combining these ingredients as before we obtain the following counterpart to Lemma 5.3.

Lemma 6.1 *Suppose that*

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[n^c]$$

for some constant $c \geqslant 1$. *Then for any functions* T *and* S *and any integer* $k \geqslant 0$

$$\mathsf{NTISP}[T,S] \subseteq \mathsf{NTIME}[(T \cdot S^k)^{g_k} + (n+S)^{c^{2k}}],$$

where

$$\begin{aligned} g_0 &= 1 \\ g_{k+1} &= c^2 g_k/(1+cg_k). \end{aligned} \tag{21}$$

The sequence $(g_k)_k$ starts out with $g_0 = 1$, $g_1 = c^2/(1+c)$, etc. Similar arguments as the ones we made for the sequence $(f_k)_k$ exhibit the following properties: For each $k \geqslant 1$, g_k is a monotone increasing function of c for $c \geqslant 1$. For each fixed $c \geqslant 1$, the sequence $(g_k)_k$ monotonically converges to the fixed point of the transformation $g \rightarrow c^2g/(1+cg)$, i.e., to $g_\infty \doteq c - 1/c$. The sequence $(g_k)_k$ is decreasing iff $g_0 > g_1$, or equivalently, iff $c < \phi$. For each k, the transformation $c \rightarrow 1/g_k$ has a unique fixed point $c \geqslant 1$, which monotonically increases to the fixed point of $c \rightarrow 1/g_\infty$, namely $\sqrt{2}$. An explicit expression for the sequence is:

$$g_k = \begin{cases} 1/(k+1) & \text{if } c = 1 \\ (c - \frac{1}{c}) \cdot \frac{1}{1 - \frac{1+c-c^2}{c^{2k+1}}} & \text{if } c > 1. \end{cases}$$

Proof of Lemma 6.1: The proof goes by induction on k. As usual, the base case trivially holds.

For the induction step $k \rightarrow k+1$, consider a nondeterministic machine M that runs in time T and space S on an input x of length n. As in the deterministic case, we analyze the simulation defined by (1) when we apply the induction hypothesis to the matrix:

$$\underbrace{(\exists\, C_1, C_2, \ldots, C_{b-1})\, \underbrace{(\forall\, 1 \leqslant i \leqslant b)\ \underbrace{C_{i-1} \vdash^{T/b} C_i}_{(\alpha)}.}_{(\beta)}}_{(\gamma)}$$

Part (α) defines a nondeterministic computation on input x, C_{i-1}, and C_i with the following parameters:

$$\begin{aligned} \text{input size:} &\quad n + 2S \\ \text{running time:} &\quad O(T/b) \\ \text{space used:} &\quad O(S). \end{aligned}$$

By the induction hypothesis combined with the hypothesis of the lemma, we can turn (α) into a conondeterministic computation running in time

$$O(((T/b \cdot S^k)^{g_k} + (n+S)^{c^{2k}})^c).$$

This way, (β) becomes a conondeterministic computation on input x and $C_0, C_1, \ldots, C_b$ with the following parameters:

$$\begin{aligned} \text{input size:} &\quad n + (b+1)S \\ \text{running time:} &\quad O(\log b + ((T/b \cdot S^k)^{f_k} + (n+S)^{c^k})^c). \end{aligned}$$

Applying the hypothesis of the lemma once more, we can transform this conondeterministic computation into a nondeterministic one and thus obtain a nondeterministic simulation of (γ) taking time

$$O\left((b-1)S + \left(n + (b+1)S + \log b + ((T/b \cdot S^k)^{f_k} + (n+S)^{c^k})^c\right)^c\right).$$

The latter expression simplifies to big O of

$$(bS + (T/b \cdot S^k)^{cg_k})^c + (n+S)^{c^{2(k+1)}}. \tag{22}$$

A similar argument as in the proof of Lemma 5.3 shows that the best running time we can realize is within a constant factor of (22) when we equate the first two terms, i.e., when we set $bS = (T/b \cdot S^k)^{cg_k}$. That value equals

$$2 \cdot (T \cdot S^{k+1})^{c^2 g_k/(1+cg_k)} + (n+S)^{c^{2(k+1)}},$$

which finishes the induction step as g_{k+1} satisfies the recurrence (21). $\square$

Lemma 6.1 corresponds to Step 2 of our indirect diagonalization approach from Section 5.1. In Step 3, we aim for a contradiction with a direct diagonalization result. So far we used Theorem 2.2 to do so. We could equally well have used the nondeterministic time hierarchy theorem (Theorem 2.4). In the setting of this section, the application of our two direct diagonalization tools leads to different results. The one using Theorem 2.2 will turn out to be critical for the proof of Theorem 1.2 but we state both.

Theorem 6.2 *For any constant $c \geqslant 1$ and functions t and s,*

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{coNTIME}[n^c] \cap (\mathsf{NTISP}[t,s] \cup \mathsf{coNTISP}[t,s])$$

if for some integer $k \geqslant 0$

$$(t \cdot s^k)^{g_k} + s^{c^{2k}} = o(n), \tag{23}$$

where g_k is given by (21).

Proof: We first prove the $\mathsf{coNTISP}$ result by obtaining a contradiction with Theorem 2.2 from the assumption that

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[n^c] \cap \mathsf{coNTISP}[t, s] \tag{24}$$

for some integer $k \geqslant 0$ for which (23) holds. For any function $\tau(n) \geqslant n$, we have

$$\begin{aligned} \mathsf{NTIME}[\tau] &\subseteq \mathsf{coNTISP}[t(\tau), s(\tau)] \text{ (by padding (24))} \\ &\subseteq \mathsf{coNTIME}[(t(\tau) \cdot s(\tau)^k)^{g_k} + (n + s(\tau))^{c^{2k}}] \text{ (by Lemma 6.1)} \\ &= \mathsf{coNTIME}[(t(\tau) \cdot s(\tau)^k)^{g_k} + s(\tau)^{c^{2k}}] \text{ (provided } t(\tau(n))^{g_k} \geqslant n^{c^{2k}}) \\ &\subseteq \mathsf{coNTIME}[o(\tau)] \text{ (by (23)).} \end{aligned} \tag{25}$$

Note that there exists a function τ such that $t(\tau(n))^{g_k} \geqslant n^{c^{2k}}$. For example, a polynomial of sufficiently large degree will do since we can assume without loss of generality that $t(n) \geqslant n$. Thus, we obtain the contradiction with Theorem 2.2 we need.

The NTISP result is similar but starts with an NTISP simulation instead of a $\mathsf{coNTISP}$ simulation in step (25) and obtains a contradiction with the nondeterministic time hierarchy theorem instead of Theorem 2.2. $\square$

The instantiation of Theorem 6.2 for logarithmic space s implies that either $\mathsf{NTIME}[n]$ is not in nondeterministic logspace or else there exists a constant $c > 1$ such that $\mathsf{NTIME}[n]$ is not contained in $\mathsf{coNTIME}[n^c]$. As in the deterministic case, Fortnow [For00b] already proved this instantiation and we get an improvement in the dependence of the constant c on the running time, $O(n^a)$, of the presumed nondeterministic logspace algorithm for nondeterministic linear time. A careful analysis of Fortnow's technique rules out any c such that $c < \sqrt[2k]{(k+1)/a}$ for some integer $k \geqslant 1$, whereas letting k grow to infinity in Theorem 6.2 and using the fact that g_k converges to $c - 1/c$ rules out any c for which $c < 1/c + 1/a$.

Similar to Corollary 5.5 for deterministic simulations, we obtain the following bound on the time-space product of any conondeterministic simulation of nondeterministic linear time.

Corollary 6.3 *If* $\mathsf{NTIME}[n] \subseteq \mathsf{coNTISP}[t, s]$ *then* $ts = \Omega(n^a)$, *where* a *is the positive solution of* $a(a^2 - 1) = 1$, *about 1.324.*

Proof: A similar analysis as in the proof of Corollary 5.5 leads to the lower bound $ts = \Omega(n^a)$ where $a = \max_{c \geqslant 1} \min(c, 1 + 1/c^{2k}, 1 - 1/k + 1/(kg_k))$. The latter expression turns out to be maximized for $k = 1$, yielding $a = \max_{c \geqslant 1} \min(c, 1 + 1/c^2, (1 + c)/c^2)$. Since the second term is at least as large as the third one for $c \geqslant 1$, we have that a is the positive solution to $c = (1+c)/c^2$. $\square$

Multiple applications of the divide-and-conquer strategy (1) allow us to establish larger lower bounds on ts^k where k denotes the number of applications.

For conondeterministic simulations with running times of the form $t(n) = O(n^c)$ for some constant c, Theorem 6.2 implies the following simpler statement.

Theorem 6.4 *For any constant* $c \geqslant 1$ *and integer* $k \geqslant 1$,

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{coNTISP}[n^c, o(n^d)],$$

where

$$d = \min((1/g_k - c)/k, 1/c^{2k}) \tag{26}$$

and g_k *is defined by (21).*

Note that $d = 1$ for $c = 1$ and that $d < 1$ for $c > 1$. A given value of k yields nontrivial results iff c is less than the fixed point of $c \to 1/g_k$ for $c \geqslant 1$.

Proof of Theorem 5.7: Analogous to the proof of Theorem 5.7 but use the $\mathsf{coNTISP}$ part of Theorem 6.2 instead of Theorem 5.4. $\square$

Considering the case $k = 1$ in (26), we have:

Corollary 6.5 *For any constant $c \geqslant 1$,*

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{coNTISP}[n^c, o(n^{\frac{1+c}{c^2} - c})].$$

Depending on c, larger values of k may give stronger results than Corollary 6.5. In particular, Corollary 6.5 only gives nontrivial results in case $\frac{1+c}{c^2} - c > 0$. The latter condition is equivalent to c being less than the fixed point of $c \to 1/g_1$, i.e., to $c(c^2 - 1) < 1$, and implies an upper bound for c of about 1.324.

Since the fixed point of $c \to 1/g_k$ converges to $\sqrt{2}$, Theorem 6.4 gives interesting results for values of c up to $\sqrt{2}$.

Proof of Theorem 1.2: The proof is similar to the proof of Theorem 1.1 but uses the sequence g_k instead of f_k. Since the sequence g_k is monotone, the condition that there exists an integer k for which $c \cdot g_k < 1$ is equivalent to the condition that $c \cdot g_0 < 1$ or $c \cdot g_\infty < 1$. Since $g_\infty = c - 1/c$, we end up with the condition $c < \sqrt{2}$. $\square$

In particular, Theorem 1.2 implies that $\mathsf{NTIME}[n] \not\subseteq \mathsf{coNTIME}[n^c, n^{o(1)}]$ for any $c < \sqrt{2}$.

We point out the crucial role of Theorem 2.2 in the results of this section. If our strategy was to obtain a contradiction with the nondeterministic time hierarchy theorem instead of Theorem 2.2 in the proof of Theorem 5.7, we would have needed one more application of our hypothesis that $\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[n^c]$ to transform the final conondeterministic simulation into a nondeterministic one. Since the latter transformation raises the running time to the power c, the condition in the proof of Theorem 1.2 would become $c^2 \cdot g_\infty < 1$, or equivalently, $c(c^2 - 1) < 1$. The use of Theorem 2.2 allows us to relax the latter condition to $c^2 - 1 < 1$, resulting in an increase in the bound on c from about 1.324 to $\sqrt{2} \approx 1.414$.

Similar to the deterministic case, Theorem 6.4 allows us to improve the lower bound on the time-space product of Corollary 6.3 in the case of polynomial time bounds.

Corollary 6.6 *If $\mathsf{NTIME}[n] \subseteq \mathsf{coNTISP}[t, s]$ for some t of the form $t = n^c$ then $ts = \Omega(n^{1.406})$.*

Proof: The result follows from the solution of a similar optimization problem as in the proof of Corollary 5.9. Replacing the term f_k by g_k and c^k by c^{2k} in that proof leads to optimal choices of $k = 4$ and $c \approx 1.3841$ resulting in the lower bound $ts = \Omega(n^a)$ for $a = 1.4064$. $\square$

7 Lower Bounds for Other Complexity Classes

In this section, we apply our indirect diagonalization argument to obtain time-space lower bounds for complexity classes other than nondeterministic linear time. We first strengthen some of the lower bounds of Sections 5 and 6 by showing that they hold not just for $\mathsf{NTIME}[n]$ but even for $\mathsf{NTISP}[n, n^a]$ for some $a < 1$. Then we apply our technique to classes higher up in the polynomial-time hierarchy.

7.1 Nondeterministic Linear Time and Sublinear Space

The crux of our lower bounds for nondeterministic linear time in Sections 5 and 6 are careful simulations that follow from hypotheses like $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^d]$ or $\mathsf{NTIME}[n] \subseteq \mathsf{coNTISP}[n^c, n^d]$. As we will see next, the amount of space we need for these simulations is typically only about $t^{1/c}$, where t denotes the running time of the simulation. It follows that several of our previous lower bounds even hold for $\mathsf{NTISP}[n, n^{1/c}]$.

We revisit both the deterministic and the conondeterministic lower bounds for nondeterministic linear time from this perspective. We start by analyzing the space needed for the transformations underlying Lemma 5.3.

Lemma 7.1 *Suppose that*

$$\mathsf{NTISP}[n, \sigma] \subseteq \mathsf{coNTISP}[n^c, \rho] \tag{27}$$

for some constant $c \geqslant 1$ and functions σ and ρ. Then for any functions T and S and any integer $k \geqslant 1$

$$\mathsf{DTISP}[T, S] \subseteq \mathsf{coNTISP}[R, R^{1/c} + \rho(R^{1/c})],$$

where $R \doteq (T \cdot S^k)^{f_k} + (n+S)^{c^k}$ and f_k is defined by (13), provided that

$$\begin{aligned} S &= O(\sigma(R^{1/c^k})) \\ n^{1/c} + \rho(n^{1/c}) &= O(\sigma(n)). \end{aligned} \tag{28}$$

Condition (28) is not required for $k = 1$.

Proof: We follow the induction proof of Lemma 5.3 but start with $k = 1$ as the base case. For easy reference, we recall the key notation from that proof:

$$\underbrace{(\exists\, C_1, C_2, \ldots, C_{b-1})\, \underbrace{(\forall\, 1 \leqslant i \leqslant b)\, \underbrace{C_{i-1} \vdash^{T/b} C_i}_{(\alpha)}}_{(\beta)}}_{(\gamma)}.$$

For $k = 1$, we view (β) as a conondeterministic computation on input x and $C_0, C_1, \ldots, C_b$ with the following parameters:

$$\begin{aligned} \text{input size:}\quad & n + (b+1)S \\ \text{running time:}\quad & O(T/b) \\ \text{space:}\quad & O(S). \end{aligned}$$

Setting $bS = T/b$ as in the proof of Lemma 5.3 and noting that $R = (TS)^{c/2} + (n+S)^c$, (β) becomes a conondeterministic computation on an input of size $O(R^{1/c})$ that runs in time $O(R^{1/c})$ and space $O(S)$. Provided $S = O(\sigma(R^{1/c}))$, we can apply (27) to transform (β) into a nondeterministic computation taking time $O(R)$ and space $O(\rho(R^{1/c}))$. This way, (γ) becomes a nondeterministic computation that runs in time $O(bS + R) = O(R)$ and space $O(bS + \rho(R^{1/c})) = O(R^{1/c} + \rho(R^{1/c}))$, the additional time and space being for guessing and storing the intermediate configurations. This takes care of the base case $k = 1$.

For the induction step $k \to k+1$, we first need to transform (α) into a conondeterministic computation. We can use the induction hypothesis to do so provided $S = O(\sigma(\tilde{R}^{1/c^k}))$, where

$\tilde{R} = ((T/b) \cdot S^k)^{f_k} + (n+S)^{c^k}$. We thereby turn ($\beta$) into a conondeterministic computation with the following parameters:

$$\begin{aligned} \text{input size: } & n + (b+1)S \\ \text{running time: } & O(\tilde{R}) \\ \text{space: } & O(\tilde{R}^{1/c} + \rho(\tilde{R}^{1/c})). \end{aligned}$$

Provided the latter space bound is $O(\sigma(n + bS + \tilde{R}))$, we can apply (27) to turn (β) into a nondeterministic computation that runs in time $O((n + bS + \tilde{R})^c)$ and space $O(\rho(n + bS + \tilde{R}))$. With the settings as in the proof of Lemma 5.3, we have that $R = \Theta((n + bS + \tilde{R})^c)$. Thus, (28) guarantees that the provision for applying (27) is met. We end up with a nondeterministic simulation of (γ) that runs in time $O(bS + R) = O(R)$ and space $O(bS + \rho(R^{1/c})) = O(R^{1/c} + \rho(R^{1/c}))$. This finishes the induction step. $\square$

In Section 5, we combined Lemma 5.3 with Theorem 2.2 to obtain separations of the form (16). Similarly, by combining Lemma 7.1 with Theorem 2.3 we can derive separations of the form

$$\mathsf{NTISP}[n, \sigma] \not\subseteq \mathsf{coNTISP}[n^c, \rho] \cap \mathsf{DTISP}[t, s] \tag{29}$$

for some constants $c \geqslant 1$ and interesting functions σ, ρ, s, and t. This leads to a strengthening of Theorem 5.4. We refrain from stating that general result because it involves too many parameters. Instead, we immediately state the corresponding strengthening of Theorem 5.7, where we restrict attention to time bounds $t(n) = O(n^c)$ and space bounds $s = O(\rho)$.

How much we can restrict the space bound σ depends on the choice of the space bound ρ. Because condition (28) is not needed for $k = 1$, the case $k = 1$ allows for a more relaxed relationship between σ and ρ than other values of k. We state the result for $k = 1$ first.

Theorem 7.2 *For any constants $c \geqslant 1$ and $d \leqslant 2/c - c$,*

$$\mathsf{NTISP}[n, n^{\frac{c+d}{2}}] \not\subseteq \mathsf{DTISP}[n^c, o(n^d)]. \tag{30}$$

Proof: Assume for contradiction that $\mathsf{NTISP}[n, n^a] \subseteq \mathsf{DTISP}[n^c, o(n^d)]$. By Lemma 7.1 with $k = 1$, $\sigma(n) = n^a$ and $\rho(n) = o(n^d)$ we have that for a sufficiently large polynomial τ

$$\mathsf{NTISP}[\tau, \tau^a] \subseteq \mathsf{DTISP}[\tau^c, o(\tau^d)] \subseteq \mathsf{coNTISP}[o(R), o(R^{1/c})] \tag{31}$$

provided that $\tau^d = O(R^{a/c})$, where $R = \tau^{(c+d)c/2}$. We obtain a contradiction with Theorem 2.3 as long as $R = O(\tau)$ and $R^{1/c} = O(\tau^a)$. The conditions we need are equivalent to $d \leqslant 2/c - c$ and $a \geqslant (c+d)/2$. $\square$

The interesting values of c in Theorem 7.2 are in the range $1 \leqslant c < \sqrt{2}$. For large values of k, we can let c approach the golden ratio ϕ.

Theorem 7.3 *For any integer $k \geqslant 1$ and constants $c \geqslant 1$ and $d \leqslant \min(\frac{1}{k}(\frac{1}{f_k} - c), \frac{cf_k}{c^k - kf_k})$,*

$$\mathsf{NTISP}[n, n^a] \not\subseteq \mathsf{DTISP}[n^c, o(n^d)],$$

where $a = \max(\frac{1}{c}, \frac{c^k d}{(c+kd)f_k})$ and f_k is given by (13).

Proof: We follow the same proof outline as for Theorem 7.2. Inclusion (31) now follows provided that $\tau^d = O(R^{a/c^k})$ and $n^{1/c} = O(n^a)$ where $R = \tau^{(c+kd)f_k} + \tau^{c^k d}$. The contradiction requirements $R = O(\tau)$ and $R^{1/c} = O(\tau^a)$ remain. Let $r = \max((c + kd)f_k, c^k d)$. The conditions we get are equivalent to $r \leqslant 1$, $a \geqslant 1/c$ and $a \geqslant c^k d/r$. Note that if $c^k d > (c+kd)f_k$ then $r = c^k d$ and the last requirement for a implies that $a \geqslant 1$. In that case, we do not get an improvement over Theorem 5.7. Therefore, we additionally impose the condition $c^k d \leqslant (c + kd)f_k$. In that case, the requirement $r \leqslant 1$ simplifies to $(c+kd)f_k \leqslant 1$. Solving for d yields the bound in the statement of the theorem. □

As the above proof shows, the discrepancy in the bound on d in Theorems 5.7 and 7.3 for a given value of k is solely due to our desire to obtain a simpler expression for a in the case where we can obtain a strengthening of Theorem 5.7. In cases where the bounds effectively differ, our approach does not allow us to derive lower bounds for $\mathsf{NTISP}[n, \sigma]$ for sublinear σ.

An analysis of the expressions involved in the statement of Theorem 7.3 shows that for $k \leqslant 3$, the bounds on d in Theorems 5.7 and 7.3 coincide and equal the fixed point of $c \to 1/f_k$. For $k \leqslant 2$, the expression for a in Theorem 7.3 simplifies to $a = 1/c$ for the entire range of d. For $k = 1$, Theorem 7.2 allows a value of $a = (c + d)/2$, which can be seen to be at most $1/c$. Therefore, Theorem 7.2 is stronger than Theorem 7.3 for $k = 1$. Using the simplified bounds for $k = 2$, Theorem 7.3 implies the following strengthening of Corollary 5.8.

Corollary 7.4 *For any constant $c \geqslant 1$,*

$$\mathsf{NTISP}[n, n^{1/c}] \not\subseteq \mathsf{DTISP}[n^c, o(n^{\frac{1}{2}(\frac{c+2}{c^2}-c)})].$$

For higher values of k, the range of d may be more restricted (although still nontrivial) and the value of a may be larger than $1/c$ for a subrange of d. However, for small enough d (depending on k), the value of a in Theorem 7.3 equals $1/c$. Thus, we can strengthen Theorem 1.1 as follows.

Corollary 7.5 *For any constant $c < \phi$ there exists a positive constant d such that*

$$\mathsf{NTISP}[n, n^{1/c}] \not\subseteq \mathsf{DTISP}[n^c, n^d]. \tag{32}$$

Moreover, d approaches 1 from below when c approaches 1 from above.

Proof: Since the statement trivially holds for $c < 1$, we only need to consider the case where $c \geqslant 1$. For $1 \leqslant c < \phi$, there exists an integer k such that $d^* \doteq 1/f_k - c > 0$. By picking $d \leqslant d^*$ small enough but still positive, we can ensure that $c^k d/((c + kd)f_k) \leqslant 1/c$. Then Theorem 7.3 implies (32).

The fact that d approaches 1 when c does follows from the proof of Theorem 1.1 since there are no more constraints on the relationship between c and d than in the proof of that theorem for $k = 1$. □

We can also strengthen the results from Section 6 to lower bounds for conondeterministic simulations of $\mathsf{NTISP}[n, \sigma]$ for sublinear σ.

A space analysis of Lemma 6.1 shows the following stronger result. The proof is analogous to the one of the strengthening of Lemma 5.3 given in Lemma 7.1.

Lemma 7.6 *Suppose that*

$$\mathsf{NTIME}[n, \sigma] \subseteq \mathsf{coNTIME}[n^c, \rho] \tag{33}$$

for some constant $c \geqslant 1$ and functions σ and ρ. Then for any functions T and S and any integer $k \geqslant 1$

$$\mathsf{NTISP}[T, S] \subseteq \mathsf{NTISP}[R, R^{1/c} + \rho(R^{1/c})], \tag{34}$$

where $R \doteq (T \cdot S^k)^{g_k} + (n+S)^{c^{2k}}$ and g_k is defined by (21), provided that

$$\begin{aligned} S &= O(\sigma(R^{1/c^{2k}})) \\ \rho(n^{1/c}) &= O(\sigma(n)) \\ n^{1/c} &= O(\sigma(n)). \end{aligned}$$

The last condition is not required for $k = 1$.

The NTISP hierarchy theorem [BM80] implies that $\mathsf{NTISP}[t, s] \not\subseteq \mathsf{NTISP}[o(t), o(s)]$ provided t and s are sufficiently nice functions that don't grow too fast. Note that (34) by itself does not contradict the NTISP hierarchy theorem for $c \geqslant 1$. For some settings of the parameters in Lemma 7.6, the time on the right-hand side of (34) is little o of the time on the left-hand side, but the space on the right-hand side always exceeds the space on the left-hand side. Under the hypothesis (33), Lemma 7.6 lets us simulate nondeterministic time T and space S in less time but somewhat more space. The hypothesis (33) itself allows us to do the opposite – increase time but reduce space – at the cost of switching from nondeterministic to conondeterministic computations. By combining the two we get a contradiction with Theorem 2.3 and thereby refute hypothesis (33).

This is how we can strengthen our conondeterministic lower bounds from Section 6. We do not state the strengthening of Theorem 6.2 in its full generality because there are too many parameters involved. We do spell out the strengthening of Theorem 6.4 and some corollaries. The proofs are analogous to those of the corresponding deterministic results we just covered.

For the same reason as before, we single out the case $k = 1$.

Theorem 7.7 *For any constants $c \geqslant 1$ and $d \leqslant (1+c)/c^2 - c$,*

$$\mathsf{NTISP}[n, n^{c(c+d)/(c+1)}] \not\subseteq \mathsf{coNTISP}[n^c, o(n^d)].$$

As a corollary, we can strengthen Corollary 6.5 as follows.

Corollary 7.8 *For any constant $c \geqslant 1$,*

$$\mathsf{NTISP}[n, n^{1/c}] \not\subseteq \mathsf{coNTISP}[n^c, o(n^{\frac{1+c}{c^2} - c})].$$

For general k, we obtain the following counterpart to Theorem 7.7.

Theorem 7.9 *For any integer $k \geqslant 1$ and constants $c \geqslant 1$ and $d \leqslant \min(\frac{1}{k}(\frac{1}{g_k} - c), \frac{cg_k}{c^{2k} - kg_k})$,*

$$\mathsf{NTISP}[n, n^a] \not\subseteq \mathsf{coNTISP}[n^c, o(n^d)],$$

where $a = \max(\frac{1}{c}, \frac{c^{2k}d}{(c+kd)g_k})$ and g_k is given by (21).

Analogous remarks as we made about the phrasing of Theorem 7.3 are in place here. An analysis of the expressions involved shows that for $k \leqslant 2$, the bound on d in Theorem 7.9 coincides with the one given in Theorem 6.4 and equals the fixed point of $c \to 1/g_1$. For $k = 1$, the value of a in Theorem 7.9 equals $1/c$ for the entire range of d. Since $c(c+d)/(c+1) \leqslant 1/c$ for $d \leqslant (1+c)/c^2 - c$, this implies that Theorem 7.7 is stronger than the general Theorem 7.9 for $k = 1$.

Theorem 7.9 allows us to strengthen Theorem 1.2.

Corollary 7.10 *For any constant $c < \sqrt{2}$ there exists a positive constant d such that*

$$\mathsf{NTISP}[n, n^{1/c}] \not\subseteq \mathsf{coNTISP}[n^c, n^d].$$

Moreover, d approaches 1 from below when c approaches 1 from above.

7.2 Higher Levels of The Polynomial-Time Hierarchy

We end with a simple application of our indirect diagonalization argument to linear-time classes in higher levels of the polynomial-time hierarchy.

Theorem 7.11 *For any integer $\ell \geqslant 2$, if $\Sigma_\ell\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[t, s]$ then $ts^{\ell-1} = \Omega(n^\ell)$.*

Proof: Assume that $\Sigma_\ell\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[t, s]$. Applying (12) with $k = \ell - 1$, $T = t$ and $S = s$, we have that

$$\Sigma_\ell\mathsf{TIME}[n] \subseteq \Pi_\ell\mathsf{TIME}[(\sum_{i=1}^{k} b_i)s + t/(\prod_{i=1}^{k} b_i)].$$

The running time of the Π_ℓ simulation is minimized up to a constant factor by picking b_i equal to b where b is such that $b \cdot s = t/b^k$, i.e., $b = (t/s)^{1/(k+1)}$. This setting results in a running time of $O((t \cdot s^k)^{1/(k+1)})$. Thus, we get that

$$\Sigma_\ell\mathsf{TIME}[n] \subseteq \Pi_\ell\mathsf{TIME}[(ts^{\ell-1})^{1/\ell}],$$

which contradicts Theorem 2.2 as long as $ts^{\ell-1} = o(n^\ell)$. □

Corollary 7.12 *For any integer $\ell \geqslant 2$ and constant $\epsilon > 0$,*

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

8 Further Research

The obvious open problem is to improve the quantitative results we get, in particular the golden ratio exponent in Theorem 1.1 and the $\sqrt{2}$ exponent in Theorem 1.2. There seems to be room for improvement as the ingredients we use in our indirect diagonalization argument are rather simple and we combine them in a straightforward way.

On the other hand, some complexity theorists feel that improving the golden ratio exponent beyond 2 would require a breakthrough. One can ask about the limits of indirect diagonalization in showing time-space lower bounds for NP-complete problems, or more generally, about its limitations as a proof technique in computational complexity. Note that indirect diagonalization provides the opportunity to exploit nonrelativizing inclusion results. In that sense, it does not suffer from oracle objections as direct diagonalization does.

Another direction for further research are time-space lower bounds for satisfiability and other NP-complete problems on randomized machines with bounded two-sided error. Theorem 1.2 immediately implies time-space lower bounds for satisfiability on randomized machines that only err on the no-side. This is because the latter type of machine is a conondeterministic machine. However, we currently do not see how to apply our techniques to randomized computations with two-sided error or with one-sided error on the yes-side.

We should point out that Beame et al. [BSSV03], building on earlier work by Ajtai [Ajt99], establish nonuniform time-space lower bounds for a problem in P based on binary quadratic forms. They show that any branching program for that problem that uses only space $n^{1-\epsilon}$ for some positive constant ϵ takes time

$$\Omega(n \cdot \sqrt{\log n / \log\log n}), \tag{35}$$

and manage to extend their lower bounds to randomized computations with two-sided error. As we argued in Section 3, time-space lower bounds for problems that are easier than satisfiability imply time-space lower bounds for satisfiability provided the problems have the property that each bit of the translation to satisfiability can be computed on the fly in a time and space efficient way. The latter is the case for all problems in nondeterministic linear time. However, it does not seem like the results of Beame et al. carry over to satisfiability in this way. First, since their bound (35) is only slightly superlinear, an extremely efficient reduction of the problem they consider to satisfiability is needed in order to obtain nontrivial lower bounds for satisfiability. The reduction we described in Section 3 (Theorem 3.1) would not do. Moreover, their problem does not appear to be in nondeterministic quasi-linear time. As far as we know, nothing nontrivial is known about time-space lower bounds for satisfiability on randomized machines with two-sided error.

Another related work is by Allender et al. [AKR$^+$01], in which these authors establish time-space lower bounds on randomized machines with unbounded error but for a problem that is harder than satisfiability. They consider the problem MAJ-MAJ-SAT consisting of all Boolean formulas $\psi(x, y)$ such that for a majority of assignments to x, a majority of the assignments to y satisfies $\psi(x, y)$. They show that no randomized machine with unbounded error can solve MAJ-MAJ-SAT in time $n^{1+o(1)}$ and space $n^{1-\epsilon}$ for any positive constant ϵ. More generally, they establish time-space lower bounds for simulations of the second level of the counting hierarchy on randomized machines with unbounded error. Their proofs use our indirect diagonalization paradigm. Roughly speaking, whereas our deterministic results can be viewed as lower bounds for complete problems for the first level of the polynomial-time hierarchy on machines that correspond to the zero-th level of the polynomial-time hierarchy, the results by Allender et al. can be regarded as lower bounds for complete problems of the second level of the counting hierarchy on machines that correspond to the first level of the counting hierarchy.

Time-space lower bounds for satisfiability on nonuniform models form another topic for further investigation. Time-space lower bounds for deterministic machines straightforwardly translate into size-width lower bounds for sufficiently uniform circuits, and into depth-logarithm-of-the-size lower bounds for sufficiently uniform branching programs. Logtime uniformity is good enough for all of our results to carry over without any changes in the parameters. Fortnow shows how to apply his technique to logspace-uniform NC^1 circuits [For00b]. Allender et al. [AKR$^+$01] extend this result to logspace-uniform SAC^1 circuits and their negations, and Fortnow [For00b] states it for logspace-uniform branching programs. Van Melkebeek [vM04] derives all these circuit results as instantiations of a general theorem, and shows directly that in each case $\mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$ uniformity for any positive constant ϵ suffices. Tourlakis [Tou01] argues that the arguments of Fortnow [For00b] carry through when the machines receive subpolynomial advice. The same holds for our results.

An appealing feature of the lower bounds in this paper is their robustness – they hold for virtually any reasonable general model of computation. On the other hand, they do not yield any lower bounds for algorithms for satisfiability that use linear space, such as algorithms that explicitly store an assignment to the given formula. Another line of research tries to establish pure time lower bounds for satisfiability but on more restricted models of computation. Van Melkebeek and Raz [vMR04] build on our techniques to obtain such bounds on Turing machines with one d-dimensional work tape and random access to the input. They show that no such machine can solve satisfiability in time n^c for $c < \sqrt{(d+2)/(d+1)}$ deterministically, and for c such that $c^3 < 1 + c/(d+1)$ conondeterministically.

Acknowledgments

DvM would like to thank Scott Diehl, Rahul Santhanam, Madhu Sudan, and Iannis Tourlakis for discussions about the topic of this paper and/or comments on earlier drafts. He is grateful to Steve Cook for bringing him into contact with Iannis Tourlakis. Thanks also to the anonymous referees for their suggestions.

References

[Ajt99] M. Ajtai. A non-linear time lower bound for Boolean branching programs. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 60–70. IEEE, 1999.

[AKR+01] 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.

[BDG95] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[BM80] A. Bruss and A. Meyer. On time-space classes and their relation to the theory of real addition. *Theoretical Computer Science*, 11:59–69, 1980.

[BSSV03] 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.

[Coo88] S. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26:269–270, 1988.

[FL93] L. Fortnow and C. Lund. Interactive proof systems and alternating time-space complexity. *Theoretical Computer Science*, 113:55–73, 1993.

[For00a] L. Fortnow. Diagonalization. *Bulletin of the European Association for Theoretical Computer Science*, 71:102–112, 2000.

[For00b] L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60:337–353, 2000.

[FvM00] 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.

[GS89] Y. Gurevich and S. Shelah. Nearly-linear time. In *Proceedings, Logic at Botik '89*, volume 363 of *Lecture Notes in Computer Science*, pages 108–118. Springer-Verlag, 1989.

[HS66] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13:533–546, 1966.

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

[LV99] 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.

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

[PF79] N. Pippenger and M. Fischer. Relations among complexity measures. *Journal of the ACM*, 26:361–381, 1979.

[PPR80] W. Paul, E. Prauß, and R. Reischuk. On alternation. *Acta Informatica*, 14:243–255, 1980.

[PPST83] W. Paul, N. Pippenger, E. Szemeridi, and W. Trotter. On determinism versus nondeterminism and related problems. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 1983.

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

[SFM78] J. Seiferas, M. Fischer, and A. Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25:146–167, 1978.

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

[vM04] 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.

[vMR04] D. van Melkebeek and R. Raz. A time lower bound for satisfiability. In *Proceedings of the 31st International Colloquium On Automata, Languages and Programming*, pages 971–982. Springer-Verlag, 2004.

[Ž83] S. Žàk. A Turing machine time hierarchy. *Theoretical Computer Science*, 26:327–333, 1983.