The Computational Complexity Column

by

Lance FORTNOW

NEC Research Institute
4 Independence Way, Princeton, NJ 08540, USA
fortnow@research.nj.nec.com
http://www.neci.nj.nec.com/homepages/fortnow/beatcs

Despite the many exciting results in computational complexity, we still have not made significant progress in proving lower bounds on general models of computation like a random-access Turing machine. Over the past couple of years we have seen two approaches that make some small progress showing that some natural problems require either a nontrivial amount of time or space.

Ajtai with follow up work by Beame, Saks, Sun and Vee have taken a combinatorial approach showing some problems computable in polynomial-time require either nonlinear time or nearly linear space on nonuniform or probabilistic branching programs.

Dieter van Melkebeek, in this month's column, describes another approach using old-fashioned simulation and diagonalization to get time-space tradeoffs for Boolean formulae satisfiability, the seminal NP-complete problem.

I hope to see all of my loyal readers at the 16th Annual Conference on Computational Complexity that will be held June 18-21 at DePaul University in Chicago. There will be a special session of talks before the conference celebrating Alan Selman's 60th birthday. A link to the conference is available from the URL above.

Time-Space Lower Bounds for Satisfiability

Dieter van Melkebeek[1]

Abstract

We survey the recent lower bounds on the running time of general-purpose random-access machines that solve satisfiability in a small amount of work space, and related lower bounds for satisfiability in nonuniform models.

1 Introduction

Satisfiability is the problem of deciding whether a given propositional formula has at least one satisfying assignment. It constitutes the seminal NP-complete problem and is of major practical importance. Complexity theorists widely believe that satisfiability takes exponential time in the worst case, and requires a linear exponent in the number of variables of the formula. On the other hand, 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

[1]Institute for Advanced Study, Princeton, NJ 08540, USA Email: dieter@ias.edu

in the worst case. To date no better lower bound than the trivial one is known. The same situation holds for the size of circuits deciding satisfiability.

However, if we restrict the amount of work space a machine solving satisfiability is allowed to use then we *can* establish non-trivial lower bounds on its running time. We have seen considerable progress on such time-space lower bounds in recent years. They form the topic of this survey. Various space bounds have been considered. All are sublinear, which in particular implies that the machine is not able to store an assignment to the formula.

The first such result was obtained by Fortnow. He established slightly super-linear time lower bounds for machines solving satisfiability in space $n^{1-\epsilon}$.

Theorem 1.1 (Fortnow [For00b]) *For any positive constant ϵ, satisfiability cannot be solved on a deterministic random-access Turing machine in time $n^{1+o(1)}$ and*

space $n^{1-\epsilon}$.

Lipton and Viglas considered smaller space bounds, namely polylogarithmic ones, and obtained a polynomial lower bound of degree larger than 1 on the running time, namely essentially $n^{\sqrt{2}}$.

Theorem 1.2 (Lipton-Viglas [LV99]) *For any positive constant ϵ, satisfiability cannot be solved on a deterministic random-access Turing machine in time $n^{\sqrt{2}-\epsilon}$ and*

space $\log^{O(1)} n$.

Fortnow and Van Melkebeek captured and improved both previous results. They established a time lower bound of essentially n^{ϕ} for machines using subpolynomial space, where ϕ denotes the golden ratio, about 1.618. More precisely, they obtained the following:

Theorem 1.3 (Fortnow-Van Melkebeek [FvM00]) *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 proved Theorem 1.1 for conondeterministic machines [For00b]. Fortnow and Van Melkebeek managed to improve the time lower bound in this version of Theorem 1.1 from slightly superlinear to a polynomial of degree larger than 1, and obtained time lower bounds of essentially $n^{\sqrt{2}}$ for subpolynomial space conondeterministic machines.

Theorem 1.4 (Fortnow-Van Melkebeek [FvM00]) *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.

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 the above results to carry over without any changes in the parameters. Buhrman showed how to apply Fortnow's technique to logspace-uniform NC^1 circuits [For00b]. Allender et al. [AKR$^+$00] extended this result to logspace-uniform SAC^1 circuits and their negations, and Fortnow [For00b] stated it for logspace-uniform branching programs. We will derive all these circuit results as instantiations of a general theorem, and show directly that in each case $\mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$ uniformity for any positive constant ϵ suffices. A family $\{C_n\}_n$ of circuits is $\mathsf{NTISP}[t,s]$-uniform if there exists a nondeterministic Turing machine that on input 1^n runs in time $O(t(n))$ and space $O(s(n))$ and outputs C_n on every accepting computation path, of which there is at least one.

Theorem 1.5 *For any positive constant ϵ, satisfiability cannot be solved by $\mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$-uniform families of any of the following types:*

- *circuits of size $n^{1+o(1)}$ and width $n^{1-\epsilon}$,*
- *branching programs of size $n^{1+o(1)}$,*
- SAC^1 *circuits with $n^{1+o(1)}$ connections, and negations of such circuits.*

Recall that SAC^1 circuits are circuits of logarithmic depth with bounded fan-in AND's, unbounded fan-in OR's, and negations only on the inputs. NC^1 circuits of size $n^{1+o(1)}$ are a special type of SAC^1 circuits with $n^{1+o(1)}$ connections. Negations of SAC^1 circuits are equivalent to circuits of logarithmic depth with bounded fan-in OR's, unbounded fan-in AND's, and negations only on the inputs.

Tourlakis [Tou00] argued that the arguments of Fortnow and of Lipton and Viglas carry through when the machines receive subpolynomial advice. The same holds for the Fortnow-Van Melkebeek results.

1.1 Scope

This paper provides an overview of the known time-space lower bounds for satisfiability on general-purpose random-access machines and nonuniform models. Up to polylogarithmic factors, all results are robust with respect to the choice of random-access machine model, so the specifics do not really matter. In the arguments we present we will have a multitape Turing machine in mind with random access to all tapes. See the paper by Fortnow and Van Melkebeek [FvM00] for details.

We will not cover time-space lower bounds for problems other than satisfiability, even though they may be based on similar techniques. Examples include problems higher up in the linear-time hierarchy [Woo86, FvM00], the polynomial-time hierarchy [Kan84, Tou00], and the counting hierarchy [AKR$^+$00]. Time-space lower bounds for problems that efficiently reduce to satisfiability *are* relevant provided they have the property that each bit of the translation to satisfiability can be computed on the fly in a time and space efficient way. Then time-space lower bounds for satisfiability follow. As we will see in Section 2, problems in nondeterministic quasi-linear time are precisely those that have this property in a strong sense. However, none of the known time-space lower bounds work for such problems. In particular, the recent non-uniform time-space lower bounds by Ajtai [Ajt99] and their improvements by Beame et al. [BSSV00] do not yield time-space lower bounds for satisfiability. These authors considered a problem in P based on a binary quadratic form, and showed that any branching program for it that uses only space $n^{1-\epsilon}$ for some positive constant ϵ takes time

$$\Omega(n \cdot \sqrt{\log n / \log \log n}). \tag{1}$$

An extremely efficient reduction of the problem they considered to satisfiability is needed in order to obtain nontrivial lower bounds for satisfiability, since the bound (1) is only slightly super-linear.

The reduction we will describe in Section 2 (Theorem 2.1) would not do. Moreover, their problem does not appear to be in nondeterministic quasi-linear time.

We will also not consider time-space lower bounds for satisfiability on restricted computation models. In particular, we will not discuss multitape Turing machines, in which each tape head can move over only one tape cell during a computation step. In that model Duris and Galil [DG84] established a lower bound of $\Omega(n^2)$ on the product of the time and space needed by any machine deciding palindromes. An efficient reduction of the language of palindromes to satisfiability immediately implies a lower bound of $\Omega(n^2/\log^c n)$ for some constant c on the product of the time and space needed by any multitape Turing machine deciding satisfiability [San99]. In particular, subpolynomial space multitape Turing machines for satisfiability have to run for at least about n^2 steps. This result, in contrast to the ones for random-access machines mentioned above, does not rely on the inherent difficulty of nondeterministic computation. It rather exploits an artifact of the multitape Turing machine model – that the machine may have to waste a lot of time in moving its tape head between both ends of the input tape. On random-access machines palindromes can be decided simultaneously in quasi-linear time and logarithmic space.

1.2 Organization

We present the known results in a unified way by distilling out what they have in common. We focus on the ideas involved and ignore several details. In particular, we will not be very precise about constructibility issues. We refer to the original papers for more details.

In Section 2, we will argue that time-space lower bounds for satisfiability and for nondeterministic linear time are equivalent up to polylogarithmic factors. Whereas in this section we have stated all results in terms of satisfiability, in the rest of the paper we will think in terms of nondeterministic linear time.

Section 3 describes the proof techniques and the tools involved in proving time-space lower bounds for nondeterministic linear time. It turns out that all the proofs have a very similar high-level structure, which can be denoted as indirect diagonalization. We will describe how it works, what the ingredients are, and two ways to combine them: one due to Fortnow [For00b] and one due to Kannan [Kan84].

Kannan's technique forms the subject of Section 4. Kannan developed it to investigate the relationship between deterministic time $O(t)$ and nondeterministic time $O(t)$. Lipton and Viglas employed it to obtain time-space lower-bounds for nondeterministic linear time. We will sketch Kannan's original argument and see how Lipton and Viglas used it to prove Theorem 1.2. Fortnow and Van Melkebeek applied the technique recursively in two different ways and obtained Theorems 1.3 and 1.4, the proofs of which we will describe in some detail.

Although the latter results constitute the state-of-the-art time-space lower bounds for satisfiability on deterministic and conondeterministic machines, Fortnow's approach is still interesting because it yields circuit lower bounds that do not seem to follow from Kannan's approach. Section 5 is devoted to Fortnow's technique. We will first describe the machine lower bound of Theorem 1.1 and then show how to modify and extend it to obtain the circuit lower bounds of Theorem 1.5.

Finally, in Section 6 we will propose some directions for further research.

2 Satisfiability versus Nondeterministic Linear Time

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

We are interested in simultaneous time and space bounds, though. Since SAT lies in nondeterministic quasi-linear time, time-space lower bounds for SAT 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 SAT, 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.

Recall that $\mathsf{DTISP}[t, s]$ denotes the class of languages decidable by deterministic random-access Turing machines running in time $O(t)$ and space $O(s)$. $\mathsf{NTISP}[t, s]$ symbols the corresponding nondeterministic class.

Theorem 2.1 *There exists a constant r such that the following holds: If*

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

then

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

A detailed proof of Theorem 2.1 can be found in the paper by Fortnow and Van Melkebeek [FvM00]. Theorem 2.1 also holds in the nonuniform setting, and if we replace DTISP by $\mathsf{coNTISP}$ in both hypothesis and conclusion. It allows us to translate time-space lower bounds for nondeterministic linear time into the same time-space lower bounds for SAT up to polylogarithmic factors. In particular, for polynomial bounds we obtain:

Corollary 2.2 *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 $d' < d$

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

Again, Corollary 2.2 also holds in the nonuniform setting, and 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 SAT then follow from Theorem 2.1 or Corollary 2.2 and their variants.

3 Tools

This section describes the tools used in the proofs of the time-space lower bounds for nondeterministic linear time. All results to date share the same high-level structure, which can be described as 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). We refer to the survey paper by Fortnow [For00a] for other applications of indirect diagonalization.

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. We keep on doing this until we reach a contradiction with a direct diagonalization result.

In this section, we will first list the direct diagonalization results we will use for step 3. Then we will describe the two techniques we will apply to derive more inclusions in step 2, namely trading alternations for time and trading time for alternations. Finally, we will see how to combine these techniques to obtain a contradiction, and thereby refute the hypothesis made in step 1.

3.1 Direct Diagonalization Results

We do not attempt to define in general what direct diagonalization results are. Instead, we just list the ones we will use.

The first one is a straightforward hierarchy theorem for alternating computations. It states that one more alternation and a little bit more time allow us to do more.

Theorem 3.1 *For any positive integer a and constructible function $t(n)$*

$$\Sigma_a \mathsf{TIME}[o(t)] \subsetneq \Sigma_{a+1} \mathsf{TIME}[t].$$

Theorem 3.1 is the direct diagonalization result Fortnow used to prove Theorem 1.1.

Lipton and Viglas used a more involved direct diagonalization result, namely the standard hierarchy theorem for nondeterministic time.

Theorem 3.2 (Seiferas-Fischer-Meyer [SFM78]) *Let $t_1(n)$ and $t_2(n)$ be functions with $t_2(n)$ constructible. If $t_1(n+1) \in o(t_2(n))$ then*

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

In case $t_1(n) = n^{c_1}$ and $t_2(n) = n^{c_2}$ where c_1 and c_2 are constants, Theorem 3.2 states that nondeterministic machines can do strictly more in time t_2 than in time t_1 if $c_2 > c_1$. Theorem 3.2 also holds for $\Sigma_a \mathsf{TIME}$ instead of NTIME where a is an arbitrary integer larger than 1. This extension of Theorem 3.2 strengthens Theorem 3.1 in the polynomial time range.

It turns out that an easier direct diagonalization result than Theorem 3.2 suffices for the Lipton-Viglas argument.

Theorem 3.3 *For any positive integer a and constructible function $t(n)$*

$$\Sigma_a \mathsf{TIME}[t] \not\subseteq \Pi_a \mathsf{TIME}[o(t)].$$

Theorem 3.3 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 what we could not do before. We may not be able to do everything we could before – Theorem 3.3 is not a hierarchy result – but we can do something new. Theorem 3.3 strengthens Theorem 3.1. Its proof is equally straightforward. Nevertheless, Theorem 3.3 is powerful enough a direct diagonalization result for all indirect diagonalization arguments in this survey.

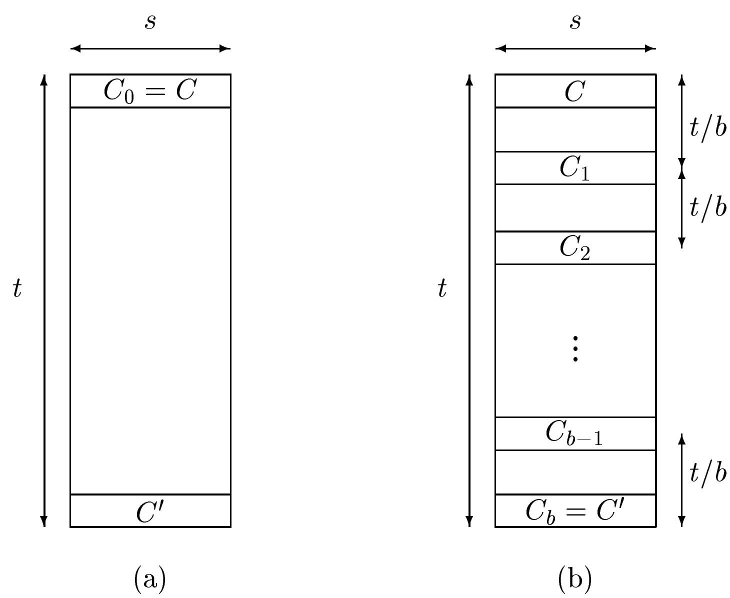

Figure 1: Tableaus of a computation using time t and space s

3.2 Trading Alternations for Time

We now start our discussion of the tools we will use to derive more inclusions of complexity classes from our hypothesis that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t, s]$. The first one is trading alternations for time, i.e., reducing the running time by allowing more alternations. We know how to do this in general for space-bounded computations. The technique consists of a divide-and-conquer strategy. It has been known for a long time and has been applied extensively in computational complexity, among others 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:

$$C \vdash^t C' \Leftrightarrow (\exists\, C_1, C_2, \ldots, C_{b-1})(\forall\, 1 \leqslant i \leqslant b)\, C_{i-1} \vdash^{t/b} C_i, \tag{2}$$

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

$$\mathsf{DTISP}[t, s] \subseteq \Sigma_2\mathsf{TIME}[bs + t/b]. \tag{3}$$

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

$$\mathsf{DTISP}[t,s] \subseteq \Sigma_2\mathsf{TIME}[\sqrt{ts}]. \tag{4}$$

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:

$$\mathsf{DTISP}[t,s] \subseteq \Sigma_{2k}\mathsf{TIME}[(\sum_i b_i)s + t/(\prod_i b_i)]. \tag{5}$$

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:

$$\mathsf{DTISP}[t,s] \subseteq \Sigma_{2k}\mathsf{TIME}[(ts^k)^{1/(k+1)}]. \tag{6}$$

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 our applications the optimal block numbers will not all be equal.

One application of (6) is Nepomnjascii's theorem. It states that $\mathsf{DTISP}[n^{O(1)}, n^{1-\epsilon}]$ is included in the linear-time hierarchy for any positive constant ϵ.

Theorem 3.4 (Nepomnjascii [Nep70]) *For any positive constants ϵ and ℓ there exists a constant k such that*

$$\mathsf{DTISP}[n^\ell, n^{1-\epsilon}] \subseteq \Sigma_k\mathsf{TIME}[n].$$

In fact, Nepomnjascii showed that Theorem 3.4 even holds for NTISP instead of DTISP. Indeed, the key ingredient (2) works for nondeterministic machines M as well. Kannan [Kan84] extended Nepomnjascii's result to an arbitrary but constant number of alternations.

Theorem 3.5 (Kannan [Kan84]) *For any positive constants ϵ and ℓ, and any nonnegative integer a there exists a constant k such that*

$$\Sigma_a\mathsf{TISP}[n^\ell, n^{1-\epsilon}] \subseteq \Sigma_k\mathsf{TIME}[n].$$

3.3 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 $\mathsf{DTIME}[t]$. Since t will be small, this means that we can simulate nondeterminism deterministically and thus eliminate alternations at a moderate expense.

For example, it follows that for any constructible function $\tau(n) \geqslant n$

$$\Sigma_2\mathsf{TIME}[\tau] \subseteq \Sigma_1\mathsf{TIME}[t \circ \tau]. \tag{7}$$

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

$$(\exists\, y_1 \in \{0,1\}^\tau)\ \underbrace{(\forall\, y_2 \in \{0,1\}^\tau)\ R(x, y_1, y_2)}_{(\alpha)}, \tag{8}$$

where R denotes a predicate computable in deterministic linear time. Part (α) of (8) defines a conondeterministic computation on input x and y_1. The running time is $O(\tau)$, which is linear in the input length since $\tau \geqslant n$. Therefore, our hypothesis implies that we can transform (α) into a deterministic computation on input x and y_1 taking time $O(t \circ \tau)$. All together, (8) then describes a nondeterministic computation on input x of time complexity $O(\tau + t \circ \tau) = O(t \circ \tau)$. □

Note that (7) also follows from the weaker hypothesis $\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[t]$. Then ($\alpha$) in (8) can only be transformed into a nondeterministic instead of a deterministic computation running in time $O(t \circ \tau)$, but (8) as a whole still remains a nondeterministic computation taking $O(t \circ \tau)$ time. The same argument also works for a larger number of alternations.

Lemma 3.6 *Let a be a positive integer and $t(n)$ and $\tau(n)$ constructible functions such that $\tau(n) \geqslant n$. If*

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

then

$$\Sigma_a\mathsf{TIME}[\tau] \subseteq \Sigma_{a-1}\mathsf{TIME}[t \circ \tau].$$

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

3.4 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 trades time for alternations, and then alternations for time. We will discuss his approach in Section 5.

- Kannan [Kan84] did it the other way around. His approach forms the basis for the Lipton-Viglas [LV99] and Fortnow-Van Melkebeek [FvM00] results. We will cover Kannan's argument and its applications in the next section.

4 Kannan's Approach

This section covers the time-space lower bounds for nondeterministic linear time that are based on Kannan's indirect diagonalization argument. We first sketch Kannan's argument. Then we will show how Lipton and Viglas used it to obtain Theorem 1.2. Finally we will describe the recursive applications by Fortnow and Van Melkebeek yielding Theorems 1.3 and 1.4.

4.1 Kannan's Argument

Kannan [Kan84] investigated 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 showed that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n, o(n)]$ using the following argument. We cast it in the indirect diagonalization paradigm presented at the beginning of Section 3.

Step 1 We assume by way of contradiction that

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n, o(n)]. \tag{9}$$

Step 2 Consider the class $\mathsf{DTISP}[\tau, o(\tau)]$ for some super-linear function $\tau(n)$. By first trading alternations for time as in (4) and then time for alternations as in (7), we obtain the following unlikely inclusion:

$$\mathsf{DTISP}[\tau, o(\tau)] \subseteq \Sigma_2\mathsf{TIME}[o(\tau)] \subseteq \mathsf{NTIME}[o(\tau)]. \tag{10}$$

Step 3 The hypothesis (9) padded to time τ and combined with (10) yields:

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

This is a contradiction with the nondeterministic time hierarchy theorem (Theorem 3.2) for functions τ that do not grow too fast, e.g., $\tau(n) = n^2$.

Kannan used this argument to derive other results about the relationship between $\mathsf{DTIME}[t]$ and $\mathsf{NTIME}[t]$ for nonlinear t. We will not state these results. Instead, we will move on and see how Lipton and Viglas employed Kannan's argument.

4.2 The Lipton-Viglas Result

We would like to establish results of the form

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n^c, n^{o(1)}] \tag{11}$$

where c is a constant larger than 1. That is, we want to rule out deterministic simulations of nondeterministic linear time that use more time but less space than in Kannan's original setting. Lipton and Viglas restricted the space even further, to polylogarithmic, but their proof works for subpolynomial space as well.

Let us run through the argument of Section 4.1 with the modified parameters. In Step 1 we assume that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$. Step 2 becomes

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

for any function $\tau(n) \geqslant n^2$. Using padding we get in Step 3 that

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

for any function $\tau(n) \geqslant n^{2/c}$. We obtain a contradiction with the nondeterministic time hierarchy theorem as long as $c^2/2 < 1$.

We conclude that (11) holds for any constant $c < \sqrt{2}$. This implies Theorem 1.2.

4.3 The Deterministic Fortnow-Van Melkebeek Result

In order to establish (11) for constants $c \geqslant \sqrt{2}$, one might try to improve Step 2 by applying the divide-and-conquer strategy of Section 3.2 recursively. That is, we use more alternations as in (5) to reduce the running time further and then remove them using Lemma 3.6 repeatedly. We obtain the following substitute for Step 2 by choosing the block numbers b_i in (5) optimally.

Lemma 4.1 *Suppose that*

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

for some constant c. *Then for any constructible function* $\tau(n)$ *and any positive integer* k

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

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

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

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, using Lemma 4.1 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

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

We reach a contradiction with the nondeterministic time hierarchy theorem 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 machine with only *one* extra alternation instead of two. We exploit the following property of deterministic computations.

$$C \vdash^t C' \Leftrightarrow (\forall C'' \neq C')\, C \nvdash^t C''. \tag{12}$$

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 3.2. We replace the matrix of (12) by the negation of the right-hand side of (2) and rename C'' to C_b for convenience.

$$C \vdash^t C' \Leftrightarrow (\forall\, C_b \neq C')(\forall\, C_1, C_2, \ldots, C_{b-1})(\exists\, 1 \leqslant i \leqslant b)\, C_{i-1} \nvdash^{t/b} C_i, \tag{13}$$

where C_0 denotes C. In terms of the tableau of Figure 2, M reaches C' from C in t steps iff the following holds: If we break up the tableau into b blocks then for every choice of intermediate

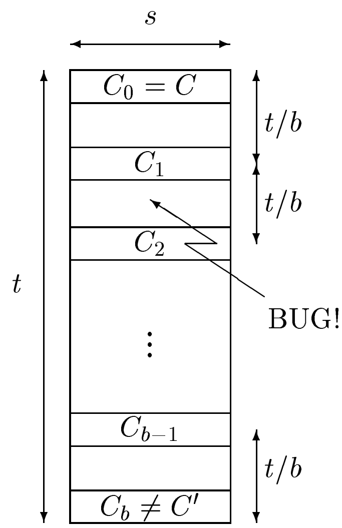

Figure 2: Saving alternations

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 $C_{i-1} \not\vdash^{t/b} C_i$ of the Π_2-formula (13) by a Σ_2-formula which is the negation of a formula of the same type as the whole right-hand side of (13). 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 3.2 that

$$\mathsf{DTISP}[t, s] \subseteq \Pi_{k+1}\mathsf{TIME}[(\sum_i b_i)s + t/(\prod_i b_i)]. \tag{14}$$

So, we achieve the same speed-up as in (5) but with only half as many alternations. An improvement of Lemma 4.1 follows.

Lemma 4.2 *Suppose that*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}] \tag{15}$$

for some constant c. Then for any constructible function $\tau(n)$ and any positive integer k

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

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

$$\begin{aligned} f_1 &= c/2 \\ f_{k+1} &= c \cdot f_k/(1+f_k). \end{aligned} \tag{16}$$

We will prove Lemma 4.2 in a moment. Note that the sequence $(f_k)_k$ converges monotonically to the fixed point of the transformation $f \to c \cdot f/(1+f)$, i.e., to $f_\infty \doteq c-1$.

Applying Lemma 4.2 in the same way as Lemma 4.1, we reach in Step 3 a contradiction with the nondeterministic time hierarchy theorem as long as $c \cdot f_k < 1$ for some positive integer k. The latter is the case iff $cf_\infty = c(c-1) < 1$. Since $c(c-1) = 1$ defines the golden ratio ϕ, we conclude that

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

for any constant $c < \phi$. A more careful analysis yields the slightly stronger Theorem 1.3.

We now give the proof of Lemma 4.2 in some detail. In particular, we will determine how to choose the block numbers b_i in (14) optimally so as to minimize the running time of the final nondeterministic simulation of $\mathsf{DTISP}[\tau, \tau^{o(1)}]$.

The proof goes by induction on k. We covered the base case in Section 4.2. We now argue the induction step $k \to k+1$.

Consider a deterministic machine M that runs in time t and space $s \in t^{o(1)}$ on an input x of length n. Let us analyze the simulation defined by (13):

$$\underbrace{(\forall\, C_b \neq C')(\forall\, C_1, C_2, \ldots, C_{b-1})\, \underbrace{(\exists\, 1 \leqslant i \leqslant b)\, \underbrace{C_{i-1} \nvdash^{t/b} C_i}_{(\alpha)}.}_{(\beta)}}_{(\gamma)} \tag{17}$$

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

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

Provided

$$b \leqslant t^{1-\epsilon} \tag{18}$$

for some positive constant ϵ, s is in $(t/b)^{o(1)}$, which makes (α) a $\mathsf{DTISP}[\tau, \tau^{o(1)}]$ computation for $\tau = t/b$. We can apply the induction hypothesis to turn (α) into a nondeterministic computation taking less time provided $(t/b)^{f_k} \geqslant (n+2s)^{c^k}$. Using (18) the latter constraint is equivalent (up to constant factors) to

$$(t/b)^{f_k} \geqslant n^{c^k}. \tag{19}$$

Under these conditions we can transform (α) into a nondeterministic computation taking time $(t/b)^{f_k+o(1)}$. Thus (β) becomes a nondeterministic computation on input x and $C_0, C_1, \ldots, C_b$ with the following parameters:

$$\begin{aligned} \text{input size: } & n + (b+1)s \\ \text{running time: } & (t/b)^{f_k+o(1)}. \end{aligned}$$

Provided

$$(t/b)^{f_k+o(1)} \geqslant n + (b+1)s \tag{20}$$

hypothesis (15) allows us to simulate (β) deterministically in time $(t/b)^{cf_k+o(1)}$. Thus, (γ) becomes a conondeterministic computation taking time (neglecting an $O(\log b)$ term)

$$bs + (t/b)^{cf_k+o(1)}. \tag{21}$$

Our goal is to minimize (21). Note that condition (20) implies that $(t/b)^{f_k+o(1)} \geqslant bs$, which means that the second term in (21) is dominant. Therefore, the smaller b, the better. Equality in $(t/b)^{f_k} \geqslant b$ determines the smallest value of b one could hope for, namely $b = t^{f_k/(1+f_k)}$. This would yield a conondeterministic simulation of (γ) taking time

$$(t/b)^{cf_k+o(1)} = b^{c+o(1)} = t^{cf_k/(1+f_k)+o(1)}.$$

All conditions (18), (19), and (20) turn out to be met provided $t^{f_{k+1}} \geqslant n^{c^{k+1}}$. We conclude that for such t, $\mathsf{DTISP}[t, t^{o(1)}] \subseteq \mathsf{coNTIME}[t^{f_{k+1}+o(1)}]$, where f_{k+1} is given by (16). This finishes the induction step since $\mathsf{DTISP}[t, t^{o(1)}]$ is closed under complementation.

4.4 The Conondeterministic Fortnow-Van Melkebeek Result

Kannan's 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.

As the first step in the derivation we assume by way of contradiction that

$$\mathsf{NTIME}[n] \subseteq \mathsf{coNTISP}[t, s] \tag{22}$$

where $t(n) = n^c$ for some constant $c \geqslant 1$ and $s(n) \in n^{o(1)}$. The two techniques we developed in Sections 3.2 and 3.3 to derive more unlikely inclusions apply to the nondeterministic setting, too.

Regarding trading alternations for time, we already pointed out in Section 3.2 that the divide-and-conquer strategy (2) works for nondeterministic machines M as well. 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 (3) because the matrix predicate on the right-hand side of (2) becomes Σ_1 in case of nondeterministic machines M. The is one reason why we obtain weaker results.

As in Section 3.2, we can apply the divide-and-conquer strategy recursively. Corresponding to (5) we obtain

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

In Section 4.3 we showed how to achieve the same speed-up as in (5) using only half the number of alternations. However, (14) does not carry over to the nondeterministic setting. Since nondeterministic machines may be able to reach more than one configuration on a given input in t steps, property (12) fails for a generic nondeterministic machine. This is the other reason why we cannot quite match the deterministic results in this section.

There are no complications as far as trading time for alternations is concerned. We observed in Section 3.3 that (7) follows from the hypothesis that $\mathsf{NTIME}[n] \subseteq \mathsf{coNTIME}[t]$. See also Lemma 3.6.

Combining these ingredients as before we obtain the following counterpart to Lemma 4.1.

Lemma 4.3 *Suppose that*

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

for some constant c. Then for any constructible function $\tau(n)$ and any nonnegative integer k

$$\mathsf{NTISP}[\tau, \tau^{o(1)}] \subseteq \mathsf{NTIME}[\tau^{g_k + o(1)}]$$

provided $\tau^{g_k}(n) \geqslant n^{c^{2k}}$, where

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

We omit the proof of Lemma 4.3. It is similar to the one of Lemma 4.2. The sequence $(g_k)_k$ converges monotonically. Since it satisfies the same recurrence as the sequence $(e_k)_k$ of Lemma 4.1, is also has the same limit value $g_\infty \doteq c - \frac{1}{c} = e_\infty$. The starting point is worse due to the additional alternation: $g_1 = c^2/(1 + c)$ is larger than $e_1 = c/2$. As a result, whereas recursion did not benefit us in the first part of Section 4.3, it will aid us here.

Lemma 4.3 corresponds to Step 2 of Section 4.1. In Step 3 we aim for a contradiction with a direct diagonalization result. So far we used the nondeterministic time hierarchy theorem (Theorem 3.2) to do so. We could equally well have used Theorem 3.3 instead. In the nondeterministic setting the use of Theorem 3.3 turns out to be crucial. Theorem 3.1 would yield weaker lower bounds.

The hypothesis $\mathsf{NTIME}[n] \subseteq \mathsf{coNTISP}[n^c, n^{o(1)}]$ padded to time τ and Lemma 4.3 imply that for any nonnegative integer k and any sufficiently large polynomial τ

$$\mathsf{NTIME}[\tau] \subseteq \mathsf{coNTISP}[\tau^c, \tau^{o(1)}] \subseteq \mathsf{coNTIME}[\tau^{cg_k+o(1)}].$$

This contradicts Theorem 3.3 if $c \cdot g_k < 1$, which is the case if $cg_\infty = c(c - \frac{1}{c}) < 1$ and k is sufficiently large.

We conclude that

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{coNTISP}[n^c, n^{o(1)}] \tag{23}$$

for any constant $c < \sqrt{2}$. A more careful analysis leads to Theorem 1.4. Note that one level of recursion ($k = 1$) only allows us to establish (23) for values of c satisfying $c(c^2 - 1) < 1$, i.e., up to about 1.324.

5 Fortnow's Approach

In this section we explain Fortnow's indirect diagonalization argument. It yields time-space lower bounds for nondeterministic linear time on deterministic and conondeterministic machines, as well as on certain classes of circuits. The machine-based applications are superseded by the Fortnow-Van Melkebeek results but the circuit results do not seem to follow from Kannan's approach. We first sketch Fortnow's result for deterministic machines because it paves the way for the circuit results. Then we discuss the latter.

5.1 Machine-Based Result

Fortnow's approach uses the same ingredients as Kannan's. Whereas Kannan first trades alternations for time and then time for alternations, Fortnow does it the other way around. The idea is the following.

Suppose that $\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[t, s]$ for some polynomial t and some small function s. Then the polynomial-time hierarchy collapses to P, i.e., we can efficiently simulate an arbitrary but constant number of alternations. Moreover, if s is sufficiently small, our hypothesis combined with Nepomnjascii's Theorem implies that P lies in the linear-time hierarchy, i.e., we can speed up arbitrary polynomial-time deterministic computations to linear time on an alternating machine with a constant number of alternations. This does not give us a contradiction yet. However, if we somehow manage to simulate *more than a constant* number of alternations in P using our hypothesis then we obtain a contradiction with the hierarchy theorem for alternating machines.

Dealing with an unbounded number of alternations requires some more care than in the bounded case, in particular while trading time for alternations. We postpone delving into the details and just state the result for now.

Lemma 5.1 *If*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}[n^{1+o(1)}] \tag{24}$$

then for every positive constant δ there exists an unbounded function $a(n)$ such that

$$\Sigma_a\mathsf{TIME}[n \log n] \subseteq \mathsf{NTIME}[n^{1+\delta}]. \tag{25}$$

The time bound $t(n) \in n^{1+o(1)}$ on the right-hand side of (24) is precisely what we need in the proof to obtain the inclusion $\Sigma_a^{\mathrm{p}} \subseteq \mathsf{P}$ for some *unbounded* function $a(n)$. If $t(n)$ is sufficiently constructible, so will be $a(n)$ but we will not worry about this issue. The precise form of the time bound $n \log n$ on the left-hand side of (25) is not that important. Any nice slightly super-linear function would do for the argument below.

We now work out the optimal parameters for Fortnow's proof. We develop it following the indirect diagonalization paradigm of Section 3.

Step 1 We assume by way of contradiction that

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTISP}[n^{1+o(1)}, n^{1-\epsilon}] \tag{26}$$

for some positive constant ϵ.

Step 2 By trading time for alternations and setting $\delta = \epsilon$ in Lemma 5.1, we have that

$$\Sigma_a\mathsf{TIME}[n \log n] \subseteq \mathsf{NTIME}[n^{1+\epsilon}] \tag{27}$$

for some unbounded function $a(n)$.

Next we would like to trade alternations for time. We need space bounds in order to do so. Our hypothesis provides them:

$$\mathsf{NTIME}[n^{1+\epsilon}] \subseteq \mathsf{DTISP}[n^{1+\epsilon+o(1)}, n^{1-\epsilon^2}]. \tag{28}$$

Now we can apply Nepomnjascii's Theorem, which says that

$$\mathsf{DTISP}[n^{1+\epsilon+o(1)}, n^{1-\epsilon^2}] \subseteq \Sigma_k\mathsf{TIME}[n] \tag{29}$$

for some constant k.

Step 3 Combining (27), (28), and (29), we get

$$\Sigma_a\mathsf{TIME}[n \log n] \subseteq \Sigma_k\mathsf{TIME}[n],$$

a contradiction with the direct diagonalization result of Theorem 3.1.

We have established Theorem 1.1. As a matter of fact, the above argument also allows us to prove the same time-space lower bound for nondeterministic linear time on conondeterministic machines. However, our interest lies in extending these bounds to circuits, and we only need the deterministic result for that.

Before switching to circuits we still have to prove Lemma 5.1. In order to eliminate an unbounded number of alternations we will explicitly construct a logical formula with alternating quantifiers that expresses the acceptance criterion of the alternating machine. Then we will manipulate this formula and eliminate the alternations one by one in a uniform way.

We need a couple of tools to do so. The first one is a lemma that captures nondeterministic computations in short Σ_1-formulas. It was proved by Cook [Coo88] for multitape Turing machines. The result for random-access machines follows from the quasi-linear simulation by Gurevich and Shelah [GS89] of nondeterministic random-access Turing machines by nondeterministic multitape Turing machines.

Lemma 5.2 *There exists a constant r such that the following holds for any language $L \in \mathsf{NTIME}[t]$ where $t(n)$ is a constructible function. Given n, we can construct in time $O(t(n)\log^r t(n))$ a Σ_1-formula φ_n with free variables $x = x_1x_2\ldots x_n$ such that for any setting of $x \in \{0,1\}^n$, $\varphi_n(x)$ holds iff $x \in L$.*

The next tool provides the key for eliminating alternations in a uniform way.

Lemma 5.3 *Suppose that*

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

for some constructible function $t(n)$. There exists a constant r such that we can transform any π_1-formula of size τ into a logically equivalent Σ_1-formula of size $\tau' \doteq t(\tau\log^r\tau)\cdot\log^r t(\tau\log^r\tau)$. The transformation takes time $\tau'\log^r\tau'$.

Note that the formulas in Lemma 5.3 can have free variables. Being logically equivalent means sharing the same free variables and having the same value for each setting of the free variables.

Proof of Lemma 5.3. Consider the language L consisting of all binary strings $\psi z 0 1^{|z|}$ such that ψ is a Π_1-formula with $|z|$ free variables and $\psi(z)$ is true. The language L can be decided in conondeterministic quasi-linear time. Therefore, by our hypothesis, $L \in \mathsf{DTIME}[t(n\log^{O(1)} n)] \subseteq \mathsf{NTIME}[t(n\log^{O(1)} n)]$. We apply Lemma 5.2 to L with $t(n\log^{O(1)} n)$ as time bound.

Given a Π_1-formula ψ of size τ with m free variables, construct the formula $\varphi \doteq \varphi_{\tau+2m+1}$ provided by Lemma 5.2. Fixing in φ the free variables corresponding to ψ and $|z|$, leaving only z free, yields a Σ_1-formula with the required properties. ☐

We are now ready to prove Lemma 5.1. Assume that

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

The acceptance criterion of a Σ_a-machine M that runs in time τ on inputs x of length n can be expressed as

$$(\exists\, y_1 \in \{0,1\}^\tau)(\exists\, y_2 \in \{0,1\}^\tau)\ldots(Q_{a-1}\, y_{a-1} \in \{0,1\}^\tau)\,\underbrace{(Q_a\, y_a \in \{0,1\}^\tau)\, R(x, y_1, y_2, \ldots, y_a)}_{(\alpha)}, \quad (30)$$

where Q_b denotes an existential quantifier if b is odd and a universal one otherwise, and R is a predicate computable deterministically in time linear in the size of its arguments. Because of Lemma 5.2 we can assume without loss of generality that R actually denotes a Σ_1-formula of size quasi-linear in $n + a\tau$ in case a is odd, and a Π_1-formula of the same size in case a is even. The resulting entire formula ψ_0 has size τ_0, where τ_0 is quasi-linear in $n+a\tau$. Moreover, we can compute ψ_0 in time quasi-linear in its length.

Using Lemma 5.3 we can replace (α) in (30) by a formula in quantifier prefix form with only quantifiers of type Q_{a-1}, thus obtaining a Σ_{a-1}-formula ψ_1 that is logically equivalent to ψ_0. We can merge the last two quantifiers, both of type type Q_{a-1}, in ψ_1, and apply Lemma 5.3 again, yielding a Σ_{a-2}-formula ψ_2 logically equivalent to ψ_1. We repeat this process $a-1$ times. The ith application results in a Σ_{a-i}-formula ψ_i of size τ_i. Lemma 5.3 states that

$$\tau_i = t(\tau_{i-1}\log^r\tau_{i-1})\cdot\log^r t(\tau_{i-1}\log^r\tau_{i-1})$$

for some constant r, and that we can compute ψ_i from ψ_{i-1} in time $\tau_i\log^r\tau_i$. We can make sure τ_{a-1} remains bounded by $n^{1+\delta}$ and at the same time let a grow unbounded provided $t(n) \in n^{1+o(1)}$. Since all computations, including the final nondeterministic evaluation of ψ_{a-1}, can be done in time quasi-linear in the size of the formulas involved, Lemma 5.1 follows.

5.2 Circuit Results

If $\mathsf{NP} = \mathsf{P}$ then any constant number of alternations collapses to P. In Section 5.1 we showed that the stronger assumption $\mathsf{NTIME}[n] \subseteq \mathsf{DTIME}[n^{1+o(1)}]$ allows us to collapse an unbounded number of alternations to P. In the nonuniform setting, Karp and Lipton [KL82] showed that if NP has polynomial-size circuits then the polynomial-time hierarchy collapses to Σ_2^{p}. Under the stronger hypothesis that $\mathsf{NTIME}[n]$ has circuits of size $n^{1+o(1)}$ their construction collapses an unbounded number of alternations to Σ_2^{p}. More precisely, the following equivalent of Lemma 5.1 holds.

Lemma 5.4 *If*

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

then for every positive constant δ there exists an unbounded function $a(n)$ such that

$$\Sigma_a\mathsf{TIME}[n \log n] \subseteq \Sigma_2\mathsf{TIME}[n^{1+\delta}].$$

The proof is the same as for Lemma 5.1 modulo the replacement of Lemma 5.3 by the following careful analysis of the Karp-Lipton argument.

Lemma 5.5 *Suppose that*

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

for some constructible function $t(n)$. There exists a constant r such that we can transform any π_2-formula of size τ into a logically equivalent Σ_2-formula of size $\tau' \doteq t(\tau \log^r \tau) \cdot \log^r t(\tau \log^r \tau)$. The transformation takes time $\tau' \log^r \tau'$.

Along the lines of Section 5.1, Lemma 5.4 shows us that at least one of the following must be false:

- $\mathsf{NTIME}[n] \subseteq \mathsf{SIZE}[n^{1+o(1)}]$, or
- for some positive δ and integer k, $\Sigma_2\mathsf{TIME}[n^{1+\delta}] \subseteq \Sigma_k\mathsf{TIME}[n]$.

The next lemma provides a strengthening of the former statement that implies the latter one.

Lemma 5.6 *Suppose that there exists a positive constant ϵ and a positive integer ℓ such that* $\mathsf{NTIME}[n]$

- *has $\mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$-uniform circuits of size $n^{1+o(1)}$, and*
- *lies in $\Sigma_\ell\mathsf{TISP}[n^{O(1)}, n^{1-\epsilon}]$.*

Then there exists an integer k such that

$$\Sigma_2\mathsf{TIME}[n^{1+\epsilon}] \subseteq \Sigma_k\mathsf{TIME}[n].$$

Proof. Let L be an arbitrary language in $\Sigma_2\mathsf{TIME}[n^{1+\epsilon}]$. There exists a language $L' \in \mathsf{NTIME}[n']$ such that for any string x of length n

$$x \in L \;\Leftrightarrow\; (\exists y \in \{0,1\}^{n^{1+\epsilon}})\, \langle x, y \rangle \in L'.$$

The pair $\langle x, y \rangle$ has length $n' \in O(n^{1+\epsilon})$. By the first hypothesis there exists a circuit C^* of size $m \in (n')^{1+o(1)}$ that decides L' on inputs of length n'. Moreover, there exists a nondeterministic machine M' that computes C^* and runs in time $(n')^{O(1)}$ and space $(n')^{1-\epsilon}$.

Consider the language L'' consisting of all pairs $\langle x, C\rangle$ such that x is a string of length n, C is the description of a circuit of size m, and

$$(\exists y \in \{0,1\}^{n^{1+\epsilon}})\, C(\langle x, y\rangle) = 1.$$

Let n'' denote the length of $\langle x, C\rangle$. The language L'' can be decided in nondeterministic quasi-linear time. By the second hypothesis there exists a Σ_ℓ-machine M'' that decides L'' in time $(n'')^{O(1)}$ and space $(n'')^{1-\epsilon} \log^{O(1)} n''$.

Note that $x \in L$ iff $\langle x, C^*\rangle \in L''$. In order to decide whether $x \in L$ we will run the machine M'' on input $\langle x, C^*\rangle$ without first computing and storing C^*. Each time M'' needs a bit from C^* and is in an existential state, we run M'. On rejecting paths of M' we reject and halt; on accepting paths of M' we distill the bit M'' needs and continue running M''. Whenever M'' needs a bit from C^* but is in a universal state, we do the same but accept and halt on rejecting paths of M'. This results in a Σ_ℓ-algorithm for deciding L that takes time $(n'n'')^{O(1)} = n^{O(1)}$ and space $(n'')^{1-\epsilon} \log^{O(1)} n'' + (n')^{1-\epsilon} = n^{(1+\epsilon)(1-\epsilon)+o(1)} = n^{1-\epsilon^2+o(1)}$. Theorem 3.5 finishes the proof. □

We conclude that the hypothesis of Lemma 5.6 cannot hold.

Theorem 5.7 *Let ϵ be a positive constant and ℓ a positive integer.* $\mathsf{NTIME}[n]$ *cannot both*

- *have* $\mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$*-uniform circuits of size $n^{1+o(1)}$, and*
- *be in* $\Sigma_\ell\mathsf{TISP}[n^{O(1)}, n^{1-\epsilon}]$.

Finally, we argue the instantiations of Theorem 5.7 given in Theorem 1.5. Circuits of size s and width w can be evaluated simultaneously in time $s \log^{O(1)} s$ and space $O(w \log s)$. Branching programs of size s can be evaluated in space $O(\log s)$. SAC^1 circuits can be evaluated in $\mathsf{NTISP}[n^{O(1)}, \log^2 n]$ [Ruz80]. It follows that if, for some positive constant ϵ, nondeterministic linear time has $\mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$-uniform

- circuits of size $n^{1+o(1)}$ and width $n^{1-\epsilon}$, or
- branching programs of size $n^{1+o(1)}$, or
- SAC^1 circuits with $n^{1+o(1)}$ connections, or negations of SAC^1 circuits with $n^{1+o(1)}$ connections,

then

$$\mathsf{NTIME}[n] \subseteq \mathsf{NTISP}[n^{O(1)}, n^{1-\epsilon/2}].$$

Theorem 5.7 then concludes the proof of Theorem 1.5.

6 Future Directions

The obvious open problem is to improve the quantitative results we get, in particular the golden ratio exponent in Theorem 1.3. It seems plausible that this is possible. After all, we have only used rather old and relatively easy tools, and combined them in the most straightforward way possible. There ought to be a way to do better, maybe merely by combining the same ingredients in a smarter way.

On the other hand, some people 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 satisfiability, 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 on randomized machines with bounded two-sided error. Theorem 1.4 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, the techniques we discussed do not seem to apply to two-sided error or to one-sided error on the yes-side. Beame et al. [BSSV00] managed to extend their time-space lower bounds to the two-sided error setting. However, as we explained in Section 1.1, their results do not carry over to satisfiability. Nothing nontrivial is known about time-space lower bounds for satisfiability on randomized machines with two-sided error.

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+00] E. Allender, M. Koucky, D. Ronneburger, S. Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. Manuscript, 2000.

[BSSV00] P. Beame, M. Saks, X. Sun, and E. Vee. Super-linear time-space tradeoff lower bounds for randomized computation. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, pages 169–179. IEEE, 2000.

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

[DG84] P. Dúriś and Z. Galil. A time-space tradeoff for language recognition. *Mathematical Systems Theory*, 17:3–12, 1984.

[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.

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

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982. A preliminary version appeared in STOC 1980.

[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.

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

[Ruz80] W. Ruzzo. Tree-size bounded alternation. *Journal of Computer and System Sciences*, 21:218–235, 1980.

[San99] R. Santhanam, October 1999. Personal communication.

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

[Tou00] I. Tourlakis. Time-space lower bounds for SAT on uniform and non-uniform machines. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 22–33. IEEE, 2000.

[Woo86] A. Woods. Bounded arithmetic formulas and Turing machines of constant alternation. In *Logic Colloquium '84*, pages 355–377. Elsevier, 1986.