# Time-Space Tradeoffs for Satisfiability

Lance Fortnow*
University of Chicago
Department of Computer Science
1100 E. 58th. St.
Chicago, IL 60637

**Abstract**

We give the first nontrivial model-independent time-space tradeoffs for satisfiability. Namely, we show that $\overline{\mathbf{SAT}}$ cannot be solved simultaneously in $n^{1+o(1)}$ time and $n^{1-\epsilon}$ space for any $\epsilon > 0$ on general random-access nondeterministic Turing machines. In particular, **SAT** cannot be solved deterministically by a Turing machine using quasilinear time and $\sqrt{n}$ space.

We also give lower bounds for log-space uniform $\mathbf{NC}^1$ circuits and branching programs.

Our proof uses two basic ideas. First we show that if $\overline{\mathbf{SAT}}$ can be solved nondeterministically with a small amount of time then we can collapse a nonconstant number of levels of the polynomial-time hierarchy. We combine this work with a result of Nepomnjaščiĭ that shows that a nondeterministic computation of super linear time and sublinear space can be simulated in alternating linear time. A simple diagonalization yields our main result.

We discuss how these bounds lead to a new approach to separating the complexity classes **NL** and **NP**. We give some possibilities and limitations of this approach.

## 1 Introduction

Separating complexity classes remains the most important and difficult of problems in theoretical computer science. Circuit complexity and other techniques on finite functions have seen some exciting early successes (see [BS90]) but have yet to achieve their promise of separating complexity classes above logarithmic space. Other techniques based on logic and geometry also have given us separations only on very restricted models.

We should turn back to a traditional separation technique—diagonalization. In this paper, we would like to argue that diagonalization might yet help us separating two common classes, nondeterministic logarithmic space (**NL**) and nondeterministic polynomial-time (**NP**).

We have no inherent reason to believe that diagonalization will not succeed in separating **NP** from **NL**. Relativization results for space-bounded classes are hard to interpret (see Fortnow [For94]). While there are relativization models that make **NL** = **NP**, these same models also can collapse **NL** and **AP** (alternating polynomial time) even though we know these classes differ (**PSPACE** = **AP** [CKS81], **NL** $\subseteq$ **DSPACE**$[\log^2 n]$ [Sav70] and **PSPACE** strictly contains **DSPACE**$[\log^2 n]$ [HS65]).

*Research done while on leave at the Centrum voor Wiskunde en Informatica in Amsterdam. URL: http://www.cs.uchicago.edu/~fortnow. Email: fortnow@cs.uchicago.edu. Supported in part by NSF grant CCR 92-53582, the Dutch Foundation for Scientific Research (NWO) and a Fulbright Scholar award.

Diagonalization over uniform classes also avoids the limits of combinatorial proofs described by Razborov and Rudich [RR97].

We make partial progress by giving some new time-space tradeoffs for satisfiability. We prove a general result: For $r(n)$ any unbounded function such that $r(n) = O(\frac{\log n}{\log\log n})$ and $\epsilon > 0$,

$$\overline{\mathbf{SAT}} \notin \mathbf{NTIME}[n^{1+\frac{1}{r(n)}}] \cap \mathbf{NTISP}[n^{o(r(n))}, n^{1-\epsilon}]$$

where $\mathbf{NTISP}[t(n), s(n)]$ are the languages accepted by nondeterministic Turing machines using $t(n)$ time and $s(n)$ space.

From this result we get many interesting consequences. For any $\epsilon > 0$,

$$\begin{aligned}
\overline{\mathbf{SAT}} &\notin \mathbf{NTISP}[n^{1+o(1)}, n^{1-\epsilon}] \\
\mathbf{SAT} &\notin \mathbf{DTISP}[n^{1+o(1)}, n^{1-\epsilon}] \\
\overline{\mathbf{SAT}} &\notin \mathbf{NTIME}[n^{1+o(1)}] \cap \mathbf{NL} \\
\overline{\mathbf{SAT}} &\notin \mathbf{NTIME}[n \log^{O(1)} n] \cap \mathbf{NSPACE}[o(\tfrac{\log^2 n}{\log\log n})] \\
\mathbf{NQL} &\not\subseteq \mathbf{coNQL} \cap \mathbf{NL}
\end{aligned}$$

where $\mathbf{NQL}$ is nondeterministic quasilinear $(n \log^{O(1)} n)$ time.

All of these results hold for the general Turing machine model where we allow random access to the input. This follows since converting from a nondeterministic RAM machine to a nondeterministic multitape machine will only increase the time by a polylogarithmic factor [GS89, Sch78]. These are the first nontrivial machine-independent time-space tradeoffs for satisfiability.

We also give lower bounds for circuits and branching programs. We show, with the help of Buhrman, that satisfiability does not have log-space uniform $\mathbf{NC}^1$ circuits of size $n^{1+o(1)}$ or log-space uniform branching programs of size $n^{1+o(1)}$.

Our proof uses surprisingly simple and well-established techniques. We first show that if $\overline{\mathbf{SAT}}$ can be solved with slightly more than linear time then we can collapse more than a constant number of levels of the polynomial-time hierarchy to a small amount of nondeterministic time.

We then consider an extension of the work of Nepomnjaščiĭ [Nep70] that shows that any language computable in nondeterministic time $n^{\alpha(n)}$ and space $n^{1-\epsilon}$ can be solved in $\alpha(n)$ alternations and linear time for any $\alpha(n) = n^{o(1)}$.

Our main results follow by combining these results with some straightforward diagonalization.

We can also use these techniques to try to separate $\mathbf{NL}$ from $\mathbf{NP}$. If $\mathbf{P} = \mathbf{NP}$ then every constant level of the polynomial-time hierarchy collapses to $\mathbf{P}$. Suppose one could show that if $\mathbf{P} = \mathbf{NP}$ then some nonconstant level of the polynomial-time hierarchy would collapse to $\mathbf{P}$. This will separate $\mathbf{NP}$ from $\mathbf{NL}$: If $\mathbf{NP} = \mathbf{NL}$ then $\mathbf{P} = \mathbf{NP}$, so a nonconstant level of the polynomial-time hierarchy would collapse to $\mathbf{P}$ and we could diagonalize against $\mathbf{NL}$ using the ideas above. However, we show some relativizable limits of this approach: We create a relativized world where for any $\epsilon > 0$, $\mathbf{SAT}$ is in $\mathbf{DTIME}[n^{1+\epsilon}]$ but every nonconstant level of the polynomial-time hierarchy does not collapse to $\mathbf{P}$.

Still one may try to hope for results like $\mathbf{NP} = \mathbf{NL}$ implies that a nonconstant level of $\mathbf{PH}$ collapses to $\mathbf{P}$. This will still separate $\mathbf{NP}$ from $\mathbf{NL}$. Stronger assumptions like $\mathbf{NP} = \mathbf{L}$ and $\mathbf{NP}$ in uniform $\mathbf{NC}^1$ will separate $\mathbf{NP}$ from $\mathbf{L}$ and $\mathbf{NP}$ from uniform $\mathbf{NC}^1$ respectively.

We also show that if nonconstant levels of the polynomial-hierarchy cannot be solved in super-logarithmic space then $\mathbf{NP} \neq \mathbf{NL}$.

Complexity theorists have devoted much effort to separating complexity classes like **NL** and **NP**. This paper shows that separating these classes might be not nearly as difficult as previously believed, perhaps considerably easier than separating **P** from **NP**.

### 1.1 Related Work

The question of time-space tradeoffs goes back to 1966 when Cobham [Cob66] showed that palindromes required quadratic time-space tradeoffs on a Turing machine with a single-head on a read-only input tape. Much of the work on time-space tradeoffs deals with restricted machine models such as comparison models (see [Yao94]) and JAG models (see [EP95]).

For Turing machines, lower bounds typically require restricted access to the input, usually with heads that read the input tape sequentially (see [GS90, DG84, Kar86]) but these results break down if random access to the input is allowed.

Kannan shows that $\mathbf{NTIME}[n] \not\subseteq \mathbf{DTIME}[n] \cap \mathbf{NL}$ in the sequential input model. Like our paper, Kannan makes use of Nepomnjaščiĭ's Theorem [Nep70]. Later Paul, Pippenger, Szemerédi and Trotter [PPST83] separate $\mathbf{NTIME}[n]$ and $\mathbf{DTIME}[n]$ using different techniques.

Kannan also proves the following curious related result.

**Theorem 1.1 (Kannan)** *There is a $k$ such that for all $t(n)$ such that $n^k = o(t(n))$ and $t(n)$ is bounded by a polynomial,* $\mathbf{NTIME}[t(n)]$ *strictly contains* $\mathbf{DTISP}[t(n), o(t^{\frac{1}{k}}(n))]$.

His proof does not construct the $k$, it merely proves one exists. This result gives an interesting contrast with our work since his result gives time-space trade-offs for large polynomial-time functions which do not fall out of our proof. However, we get time-space tradeoffs for Satisfiability that do not appear to follow from Theorem 1.1.

## 2 Preliminaries

Most of the complexity classes discussed in this paper like **NL**, **P**, **NP** and the polynomial-time hierarchy have been well studied. Definitions and basic results of these classes can be found in basic textbooks such as Hopcroft and Ullman [HU79] or Garey and Johnson [GJ79]. We use the multitape model of Turing machines though as we discuss in Section 3, our lower bounds for **SAT** hold in more general models.

The results given only hold if the time and space bounds have the appropriate constructivity requirements. Since virtually all natural functions are constructible we ignore these issues in this paper.

We need to generalize the polynomial-time hierarchy to have super-constant levels and more general time bounds.

**Definition 2.1** *The class $\Sigma_{s(n)}^{t(n)}$ consists of the set of languages accepted by alternating Turing machines using $O(t(n))$ time, start in an $\exists$ state and on input $x$, makes at most $s(|x|)-1$ alternations. The class $\Pi_{s(n)}^{t(n)}$ has a similar definition except that the computation starts in a $\forall$ state.*

*We use $\Sigma_{s(n)}^{p}$ to represent $\cup_k \Sigma_{s(n)}^{n^k}$ and $\Sigma_{s(n)}^{\mathbf{LIN}}$ to represent $\Sigma_{s(n)}^{n}$.*

Note that for constant functions $s(n) = k$, $\Sigma_{s(n)}^{p}$ corresponds to $\Sigma_k^p$, the $k$th level of the traditional polynomial-time hierarchy.

The class $\mathbf{NC}^1$ consists of languages accepted by a family circuits of bounded fan-in and logarithmic depth. A circuit family is log-space uniform if there exists a logarithmic space-bounded Turing machine that on input $1^n$ computes $C_n$, the circuit for inputs of length $n$. A circuit family is $t(n)$-time uniform if given $n$ and a pointer to one gate $g$ of $C_n$ one can in $t(n)$-time find the pointers to the gates connected to $g$.

The class $\mathbf{DTISP}[t(n), s(n)]$ consists of those languages accepted by Turing machines using simultaneously $t(n)$ time and $s(n)$ space. $\mathbf{NTISP}[t(n), s(n)]$ is the nondeterministic version of this class.

**SAT** consists of the set of satisfiable boolean formulae. Cook [Coo71] and Levin [Lev73] independently show that **SAT** is **NP**-complete. Later Cook [Coo88], building on work of Pippenger and Fischer [PF79] and Hennie and Stearns [HS66], shows how to reduce nondeterministic time to a satisfiability question of a small formula.

**Lemma 2.2 (Cook)** *Let $M$ be a nondeterministic Turing machine running in time $t(n)$. There is a $O(t(n)\log t(n))$ time and $O(\log t(n))$ space algorithm that maps inputs $x$ of length $n$ to formulae $\phi$ of size $O(t(n)\log t(n))$ such that*

$$x \in L \Leftrightarrow \phi \in \mathbf{SAT}$$

Robson [Rob91] shows that Lemma 2.2 holds even for random-access Turing machines.

Let the language $\mathbf{QBF}_{s(n)}$ consist of quantified boolean formulae restricted to $s(n) - 1$ alternations where the first quantifier is "$\exists$". Similar to the proof that **QBF** is **PSPACE**-complete, we have $\mathbf{QBF}_{s(n)}$ is $\Sigma^p_{s(n)}$-complete. In particular, using the ideas of the proof of Cook's Lemma (Lemma 2.2), we get that any language in $\Sigma^{t(n)}_{s(n)}$ can be reduced to a $\mathbf{QBF}_{s(n)}$ question of length $O(t(n)\log t(n))$.

Gurevich and Shelah [GS89] show how to simulate a nondeterministic random-access machines by multitape Turing machines. Their proof builds on Schnorr's result [Sch78] that one can sort $m$ elements on a multitape Turing machine in $O(m \log^{O(1)} m)$ time.

**Theorem 2.3 (Gurevich-Shelah-Schnorr)** *Every language that is accepted by a nondeterministic random-access machine using $t(n)$ time is also accepted by a nondeterministic multitape Turing machine using time $t(n)\log^{O(1)} t(n)$. This result generalizes to show that any random-access alternating Turing machine using time $t(n)$ can be simulated in time $t(n)\log^{O(1)} t(n)$ on a multitape machine using the same number of alternations.*

For space one can simulate such a RAM on a multitape machine by simply rereading the input and the entire memory used every time a memory call is made (see [Sv88]).

**Theorem 2.4** *Every language accepted by a random-access Turing machine using space $s(n)$ and time $t(n)$ can be simulated by a multitape Turing machine using $s(n)$ space and $t^2(n)$ time.*

We say a function $f(n)$ is *quasilinear* if $f(n) = O(n \log^k n)$ for some constant $k$. We define the class **NQL** as the set of languages accepted in nondeterministic quasilinear time. By Theorem 2.3 we have that **NQL** is a rather robust class with the same set of languages whether one uses a two-tape Turing machine or a random-access machine. By Lemma 2.2 we have that **SAT** is complete for **NQL** under quasi-linear time reductions.

We use $\mathbf{SAT}^A$ to represent a relativized version of satisfiability (see [GJ93]). Relativized $\mathbf{SAT}^A$ has several extra predicate $A_0, A_1, \ldots$ such that $A_m(x_1, \ldots, x_m)$ has the property that

$$x_1 \ldots x_m \in A \Leftrightarrow A_m(x_1, \ldots, x_m)$$

For every oracle $A$, $\mathbf{SAT}^A$ has the following properties:

1. $\mathbf{SAT}^A$ is $\mathbf{NP}^A$ complete.
2. $\mathbf{SAT}^A$ is in $\mathbf{NTIME}^A[n \log^{O(1)} n]$. [Sch78]
3. Whether $\phi$ is in $\mathbf{SAT}^A$ depends only on strings in $A$ of length less than $|\phi|$.

# 3 Main Results

In this section we present our main results.

**Theorem 3.1** *For any $\epsilon > 0$,*

$$\overline{\mathbf{SAT}} \notin \mathbf{NTISP}[n^{1+o(1)}, n^{1-\epsilon}]$$

This result follows from a more general result.

**Theorem 3.2** *For $r(n)$ any unbounded function such that $r(n) = O(\frac{\log n}{\log\log n})$ and $\epsilon > 0$,*

$$\overline{\mathbf{SAT}} \notin \mathbf{NTIME}[n^{1+\frac{1}{r(n)}}] \cap \mathbf{NTISP}[n^{o(r(n))}, n^{1-\epsilon}]$$

We prove Theorem 3.2 in Section 4 using a standard multitape Turing machine. By Theorems 2.3 and 2.4 the result applies even to random-access machines.

One could further generalize Theorem 3.2 by using constant alternating time-space bounds or by trading off alternations with time and space but we will not follow that path here.

Theorem 3.1 also gives us a deterministic lower bound for satisfiability.

**Corollary 3.3** *For any $\epsilon > 0$,*

$$\mathbf{SAT} \notin \mathbf{DTISP}[n^{1+o(1)}, n^{1-\epsilon}]$$

We also get a lower bound for the class $\mathbf{NQL}$.

**Corollary 3.4** *For any $\epsilon > 0$,*

$$\mathbf{NQL} \not\subseteq \mathbf{coNTISP}[n^{1+o(1)}, n^{1-\epsilon}]$$

**Proof:** Corollary 3.4 follows from Corollary 3.3 and Lemma 2.2. □

Theorem 3.1 tells us there cannot exist a single machine accepting $\overline{\mathbf{SAT}}$ using at most $n^{1+o(1)}$ time and $n^{1-\epsilon}$ space. One might ask the question of whether if $\overline{\mathbf{SAT}}$ is computable in a small amount of time then no other program can compute $\overline{\mathbf{SAT}}$ in a small amount of space. We can get these kinds of results from Theorem 3.2 although with weaker bounds.

**Corollary 3.5** *For $r(n)$ any unbounded function such that $r(n) = O(\frac{\log n}{\log\log n})$,*

$$\overline{\mathbf{SAT}} \notin \mathbf{NTIME}[n^{1+\frac{1}{r(n)}}] \cap \mathbf{NSPACE}[o(r(n)) \log n]$$

*In particular,*

$$\overline{\mathbf{SAT}} \notin \mathbf{NTIME}[n^{1+o(1)}] \cap \mathbf{NL}, \text{ and}$$

$$\overline{\mathbf{SAT}} \notin \mathbf{NTIME}[n \log^{O(1)} n] \cap \mathbf{NSPACE}[o(\frac{\log^2 n}{\log\log n})]$$

**Proof:** Corollary 3.5 follows from Theorem 3.2 and the fact that any $s(n)$ space bounded Turing machine uses at most $2^{O(s(n))}$ time or it will repeat a configuration. $\square$

Corollary 3.5 and Lemma 2.2 allow us to state a lower bound completely in terms of complexity classes.

**Corollary 3.6**

$$\mathbf{NQL} \not\subseteq \mathbf{coNQL} \cap \mathbf{NL}$$

We can also get lower bounds on the time-space product.

**Corollary 3.7** *If $M$ is a nondeterministic Turing machine accepting $\overline{\mathbf{SAT}}$ in time $t(n)$ and space $s(n)$ then for some $\epsilon > 0$, $t(n)s(n) = \Omega(n^{1+\epsilon})$.*

**Proof:** If $s(n) = \Omega(\sqrt{n})$ then we have $t(n)s(n) = \Omega(n^{1.5})$ since $M$ will need linear time to read the entire input. Otherwise by Theorem 3.1 we will have $t(n) = n^{1+\epsilon}$ for some $\epsilon > 0$. $\square$

In Section 5, we show how to get lower bounds for circuits and branching programs.

# 4 Proof of Main Theorem

We prove Theorem 3.2 using two lemmas.

**Lemma 4.1** *For any unbounded $r(n)$ such that $r(n) = O(\frac{\log n}{\log\log n})$, if $\overline{\mathbf{SAT}} \in \mathbf{NTIME}[n^{1+\frac{1}{r(n)}}]$ then for any $\gamma > 0$, there is a constant $\ell$ such that*

$$\Sigma^{n \log n}_{\frac{r(n)}{\ell}} \subseteq \mathbf{coNTIME}[n^{1+\gamma}].$$

We prove Lemma 4.1 in Section 4.1.

**Lemma 4.2** *For any $\beta > 0$, constant $c$ and $1 \leq \alpha(n) \leq n^{o(1)}$,*

$$\mathbf{NTISP}[n^{c\alpha(n)}, n^{1-\beta}] \subseteq \Sigma^{\mathbf{LIN}}_{O(\alpha(n))}$$

Lemma 4.2 is a generalization of Nepomnjaščiĭ's Theorem [Nep70] who proves the result for $\alpha(n) = 1$. Lemma 4.2 follows from the even greater generalization given by Reischuk [Rei90, p. 282].

**Lemma 4.3 (Reischuk)** *For functions $t_1(n)$, $t_2(n)$, $s(n)$ and $a(n)$, every language accepted by an alternating Turing machine using time $t_1(n)^{t_2(n)}$, space $s(n)$ and $a(n)$ alternations can be recognized by an alternating Turing machine using $O(a(n)s(n) + t_1(n)t_2(n)s(n))$ time, $O(t_1(n)s(n))$ space and $O(a(n) + t_2(n))$ alternations.*

For completeness, we give a proof of Lemma 4.2 in Section 4.2.

Now assume

$$\overline{\mathbf{SAT}} \in \mathbf{NTIME}[n^{1+\frac{1}{r(n)}}] \quad (1)$$

$$\overline{\mathbf{SAT}} \in \mathbf{NTISP}[n^{\alpha(n)}, n^{1-\epsilon}] \quad (2)$$

where $r(n)$ is any unbounded function such that $r(n) \leq \frac{\log n}{\log\log n}$, $\alpha(n) = o(r(n))$ and $\epsilon > 0$.

We will set $\gamma$ later.

By straightforward diagonalization we can construct a language $L$ in $\Sigma^{n\log n}_{\frac{r(n)}{\ell}} - \Sigma^{\mathbf{LIN}}_{O(\alpha(n))}$. By Equation (1) and Lemma 4.1 we have $\overline{L}$ in $\mathbf{NTIME}[n^{1+\gamma}]$. By Lemma 2.2 we can convert decision problems on $L$ to nonsatisfiability questions of formulae of size $O(n^{1+\gamma}\log n^{1+\gamma}) < n^{1+2\gamma}$ for sufficiently large $n$.

By Equation (2) we can solve these nonsatisfiability questions by a nondeterministic Turing machine using time $t(n) < (n^{1+2\gamma})^{\alpha(n^{1+2\gamma})}$ and space $s(n) < (n^{1+2\gamma})^{1-\epsilon}$. Since $\alpha(n) < \log n$ we have $t(n) = n^{O(\alpha(n))}$. Making $\gamma = \epsilon/4$ we have

$$s(n) < n^{(1+2\gamma)(1-\epsilon)} = n^{1-\frac{\epsilon(1+\epsilon)}{2}}.$$

By Lemma 4.2 we have $L \in \Sigma^{\mathbf{LIN}}_{O(\alpha(n))}$ a contradiction. $\square$

## 4.1 Proof of Lemma 4.1

For simplicity we give the proof of Lemma 4.1 for the case $r(n) = O(\frac{\log n}{\log\log n})$.

**Proof:** Fix $\gamma > 0$ and some integer $j$. We need to prove that that if $\overline{\mathbf{SAT}} \in \mathbf{NTIME}[n\log^j n]$ then for some $\ell$,

$$\Sigma^{n\log n}_{\frac{\log n}{\ell\log\log n}} \subseteq \mathbf{coNTIME}[n^{1+\gamma}].$$

Note that if $A \in \Sigma^{n\log n}_{\frac{\log n}{\ell\log\log n}}$ then $\overline{A} \in \Sigma^{n\log n}_{\frac{\log n}{\ell'\log\log n}}$ for any $\ell' < \ell$ so it is sufficient to show

$$\Sigma^{n\log n}_{\frac{\log n}{\ell\log\log n}} \subseteq \mathbf{NTIME}[n^{1+\gamma}].$$

Given the assumption we will show how to reduce a formula of $k-1$ alternations to $k-2$ alternations without the length of the formula growing significantly. Recursively applying this reduction gives us a polynomial-size formula with no alternations.

Fix $k$. Assume $k$ even.

Consider the formula

$$\psi_k = \exists\,\overline{x}_1\forall\,\overline{x}_2\ldots\exists\,\overline{x}_{k-1}\forall\,\overline{x}_k\ \phi(\overline{x}_1,\ldots,\overline{x}_k)$$

where $\overline{x}_i$ is a vector of variables. Assume $|\psi_k| = m_k$.

Now consider the formula

$$\tau(\overline{x}_1,\ldots,\overline{x}_{k-1}) = \forall\,\overline{x}_k\ \phi(\overline{x}_1,\ldots,\overline{x}_k).$$

The truth of $\tau$ is a $\overline{\mathbf{SAT}}$ question of an input of length bounded by $m_k$.

By assumption, determining whether $\tau(\overline{x}_1,\ldots,\overline{x}_{k-1})$ is in $\overline{\mathbf{SAT}}$ can be solved in nondeterministic time $m_k\log^j m_k$ time for some $j$. By Lemma 2.2 we can reduce a nondeterministic time $t(n)$ computation to the truth of the formula $\exists\,\overline{y}\ \sigma(\overline{x}_1,\ldots,\overline{x}_{k-1},\overline{y})$ of size $O(t(n)\log t(n))$.

The size of $\sigma$ is bounded by $m_k\log^{j+2} m_k$.

The truth of the formula $\psi_k$ is equivalent to the truth of

$$\psi_{k-1} = \exists\,\overline{x}_1\forall\,\overline{x}_2\ldots\exists\,\overline{x}_{k-1}\exists\,\overline{y}\ \sigma(\overline{x}_1,\ldots,\overline{x}_{k-1},\overline{y}).$$

The formula $\psi_{k-1}$ has $k-2$ alternations and length bounded by $m_{k-1} = m_k\log^{j+2} m_k$.

Now suppose $k$ was odd. Now consider

$$\psi_k = \exists\, \overline{x}_1 \forall\, \overline{x}_2 \exists\, \overline{x}_3 \ldots \forall\, \overline{x}_{k-1} \exists\, \overline{x}_k \;\; \phi(\overline{x}_1, \ldots, \overline{x}_k).$$

Assume again that $|\psi_k| = m_k$.

Now consider the formula

$$\tau(\overline{x}_1, \ldots, \overline{x}_{k-1}) = \forall\, \overline{x}_k \;\; \neg\phi(\overline{x}_1, \ldots, \overline{x}_k).$$

The truth of $\tau$ is a $\overline{\mathbf{SAT}}$ question of an input of length bounded by $m_k$.

By the same argument as above, we can reduce $\tau(\overline{x}_1, \ldots, \overline{x}_{k-1})$ to the truth of a formula $\exists\, \overline{y}\; \sigma(\overline{x}_1, \ldots, \overline{x}_{k-1}, \overline{y})$ of size bounded by $m_k \log^{j+2} m_k$.

The truth of the formula $\psi_k$ is equivalent to the truth of

$$\psi_{k-1} = \exists\, \overline{x}_1 \forall\, \overline{x}_2 \exists\, \overline{x}_3 \ldots \forall\, \overline{x}_{k-1} \forall\, \overline{y} \;\; \neg\sigma(\overline{x}_1, \ldots, \overline{x}_{k-1}, \overline{y}).$$

As before, the formula $\psi_{k-1}$ has $k-2$ alternations and length bounded by $m_{k-1} = m_k \log^{j+2} m_k$.

Initially, $m_{s(n)} = n \log^2 n$. By using backward induction we show that $m_1$ is bounded by $n^{1+\gamma}$.

Note that $m_k \leq n^{\log n}$ so $\log m_k \leq \log^2 n$. So we have $m_{k-1} \leq m_k \log^{2j+4} n$ and $m_1 \leq m_k \log^{k(2j+4)} n$. For $k = s(n) = \frac{\log n}{\ell \log\log n}$ we have $m_k = n \log^2 n$ and

$$m_1 \leq n \log^2 n \log^{\frac{(2j+4)\log n}{\ell \log\log n}} n < n^{1+\frac{\gamma}{2}}$$

for $\ell > \frac{4j+8}{\gamma}$.

Thus we have converted an $n \log^2 n$ size $s(n)-1$ alternation formula $\phi_{s(n)}$ to a zero alternation formula $\phi_1$ of size $n^{1+\frac{\gamma}{2}}$. We can then consider $\phi_1$ as a satisfiability question and solve it in

$$\mathbf{NTIME}[n^{1+\frac{\gamma}{2}} \log^{O(1)} n^{1+\frac{\gamma}{2}}] \subseteq \mathbf{NTIME}[n^{1+\gamma}]. \;\square$$

## 4.2 Proof of Lemma 4.2

Lemma 4.2 generalizes the following result of Nepomnjaščiĭ [Nep70].

**Theorem 4.4 (Nepomnjaščiĭ)** *For any $\epsilon > 0$,*

$$\mathbf{NTISP}[n^{O(1)}, n^{1-\epsilon}] \subseteq \bigcup_k \Sigma_k^{\mathbf{LIN}}$$

While credit is usually given to Nepomnjaščiĭ for Theorem 4.4, the result has quite an interesting history. Smullyan [Smu61, p. 147] defined the rudimentary sets. Bennett [Ben62] proves a result on rudimentary sets that implies that every language computed in polynomial time and square root space is rudimentary. Nepomnjaščiĭ [Nep70] extended Bennett's work to Turing machines and showed that every set in $\mathbf{NTISP}[n^{O(1)}, n^{1-\epsilon}]$ is rudimentary. Wrathall [Wra78] later showed that the rudimentary sets are exactly $\Sigma_k^{\mathbf{LIN}}$. Kannan [Kan84] rediscovers Theorem 4.4 and shows that it holds for random-access machines. Kannan [Kan84] and independently Woods [Woo86] generalize this theorem to show constant alternating polynomial time and $n^{1-\epsilon}$ space is contained in $\bigcup_k \Sigma_k^{\mathbf{LIN}}$. Reischuk [Rei90, p. 282] gives an even broader generalization (Theorem 4.3) than our Lemma 4.2.

For completeness, we give a proof of Lemma 4.2 below.

**Begin CHECK**$(\mathbf{ID}_a, \mathbf{ID}_b, t)$
**if** $t \leq q(n)(q(n)+1)$
  **Then** Existentially guess $\mathbf{ID}_1$,...,$\mathbf{ID}_{t-1}$.
    **If** $\mathbf{ID}_a \vdash \mathbf{ID}_1$ and $\mathbf{ID}_1 \vdash \mathbf{ID}_2$ and ... and $\mathbf{ID}_{t-1} \vdash \mathbf{ID}_b$,
      **Then** output **TRUE**
      **Else** output **FALSE**
  **Else** Existentially guess $\mathbf{ID}_1$,...,$\mathbf{ID}_{q(n)}$
    Universally choose $i$ in $\{0, \ldots, q(n)\}$.
    **Let** $m = \lceil \frac{t}{q(n)+1} \rceil$.
    **Case**
      $(i = 0)$ output **CHECK**$(\mathbf{ID}_a, \mathbf{ID}_1, m)$.
      $(0 < i < q(n))$ output **CHECK**$(\mathbf{ID}_i, \mathbf{ID}_{i+1}, m)$.
      $(i = q(n))$ output **CHECK**$(\mathbf{ID}_{q(n)}, \mathbf{ID}_b, t - mq(n))$.
    **END Case**
**END CHECK**

Figure 1: Algorithm for **CHECK**

Our generalization will use techniques developed by Chandra, Kozen and Stockmeyer [CKS81] in their proof that alternating polynomial-time captures **PSPACE**. Their proof itself builds on the techniques used by Savitch [Sav70] in his proof showing **NSPACE**$[s(n)] \subseteq$ **DSPACE**$[s^2(n)]$. Our proof uses techniques similar to Kannan [Kan84] in his proof of Theorem 4.4.

Fix a nondeterministic Turing machine $M$ using time $n^{c\alpha(n)}$ and space $n^{1-\beta}$ and an input $x$. Consider the tableau of some potential accepting computation $M$. Each row $i$ describes the entire configuration of $M$ at time $i$ (except for the input) which has $O(n^{1-\beta})$ bits. There are at most $n^{c\alpha(n)}$ configurations.

The usual divide-and-conquer algorithm of Chandra, Kozen and Stockmeyer [CKS81] would use only time $O(n^{1-\beta})$ but would use $O(\alpha(n)\log n)$ alternations so we need to be more careful.

Instead of just dividing into two pieces like Chandra, Kozen and Stockmeyer, we divide the tableau into an appropriate $n^{\epsilon}$ number of pieces. This allows us to eliminate the extra $\log n$ term.

Let $\mathbf{ID}_0$ represent the initial configuration of $M(x)$. We can assume that after $M$ accepts it erases its tape, moves the head to the left and goes to a special state and stays there forever. Call this final configuration $\mathbf{ID}_f$.

Let $\mathbf{ID}_a \vdash \mathbf{ID}_b$ be true if machine $M$ starting in $\mathbf{ID}_a$ reaches $\mathbf{ID}_b$ in one step. Checking whether $\mathbf{ID}_a \vdash \mathbf{ID}_b$ can be done in deterministic time $O(n^{1-\beta})$.

Define **CHECK**$(\mathbf{ID}_a, \mathbf{ID}_b, t)$ to be **TRUE** if $M$ starting in configuration $\mathbf{ID}_a$ will get to configuration $\mathbf{ID}_b$ in $t$ steps. We have that $M(x)$ accepts if and only if **CHECK**$(\mathbf{ID}_0, \mathbf{ID}_f, n^{c\alpha(n)})$. Fix $q(n) = n^{\epsilon}$.

In Figure 1, we show how to compute **CHECK** recursively on an alternating polynomial-time Turing machine.

Let $\mathbf{ALT}(t)$ be the number of alternations used by $\mathbf{CHECK}(\mathbf{ID}_a, \mathbf{ID}_b, t)$. Each recursive step of **CHECK** uses 2 alternations. We then have the recurrence

$$\mathbf{ALT}(t) = 2 + \mathbf{ALT}(\lceil \frac{t}{q(n)+1} \rceil)$$

For $t > q(n)(q(n)+1)$ we have

$$\lceil \frac{t}{q(n)+1} \rceil \leq \frac{t}{q(n)+1} + 1 \leq \frac{t}{q(n)}.$$

We have $\mathbf{ALT}(q(n)(q(n)+1)) = 1$ so $\mathbf{ALT}(t) = 2\log_{q(n)}(t)$.

For $\mathbf{CHECK}(\mathbf{ID}_0, \mathbf{ID}_f, n^{c\alpha(n)})$ and $q(n) = n^\epsilon$ we have

$$\mathbf{ALT}(n^{c\alpha(n)}) = 2\log_{n^\epsilon} n^{c\alpha(n)} = 2\frac{\log n^{c\alpha(n)}}{\log n^\epsilon} = 2\frac{c\alpha(n)\log n}{\epsilon \log n} = \frac{2c}{\epsilon}\alpha(n)$$

**CHECK** uses at most $O(n^{2\epsilon}n^{1-\beta})$ time for each alternation. This gives us a total time of $O(\alpha(n)n^{1+2\epsilon-\beta})$. Since $\alpha(n) = n^{o(1)}$ if we take $\epsilon < \frac{\beta}{2}$ our algorithm will use only linear time. □

# 5 Circuits and Branching Programs

Theorem 3.1 gives a lower bound for satisfiability on log-time uniform circuits. Harry Buhrman shows how to use Karp-Lipton [KL80] combined with the techniques of Section 4 to improve the bound to log-space uniformity.

**Theorem 5.1 (Buhrman)** **SAT** *cannot be solved by logarithmic-space uniform* $\mathbf{NC}^1$ *circuits of size* $n^{1+o(1)}$.

**Proof Sketch:** We prove Theorem 5.1 in several stages. First we show that if **SAT** has small (nonuniform) circuits then $\mathbf{QBF}_2$ has a quick $\Pi_2$ algorithm. We then use this fact to put $\mathbf{QBF}_{s(n)}$ formulae in $\Sigma_2^p$ for some slow growing $s(n)$. We then show that if **SAT** has log-space uniform $\mathbf{NC}^1$ circuits then $\Sigma_2^p$ is computable in logarithmic space. We can then get a contradiction by diagonalization.

Karp and Lipton prove the following result about the consequences of **NP** having small circuits.

**Theorem 5.2 (Karp-Lipton)** *If* **SAT** *has polynomial-size circuits then* $\Sigma_2^p = \Pi_2^p$.

Analyzing the proof of Karp and Lipton one discovers that the time does not increase by much on random-access machines. By Theorem 2.3 we can convert these machine to multitape Turing machines.

**Corollary 5.3 (Karp-Lipton)** *If* **SAT** *has circuits of size* $n^{1+o(1)}$ *then* $\mathbf{QBF}_2 \in \Pi_2^{n^{1+o(1)}}$.

Similar to the proof of Lemma 4.1 we can then derive that $\mathbf{QBF}_{s(n)}$ is in $\Sigma_2^p$ for a sufficiently slow growing function $s(n)$. Suppose $k$ is even and consider the formula

$$\psi_k = \exists \overline{x}_1 \forall \overline{x}_2 \ldots \forall \overline{x}_{k-2} \exists \overline{x}_{k-1} \forall \overline{x}_k \ \phi(\overline{x}_1, \ldots, \overline{x}_k)$$

where $|\psi_k| = m_k$. Now consider the formula

$$\tau(\overline{x}_1, \ldots, \overline{x}_{k-2}) = \exists\, \overline{x}_{k-1} \forall\, \overline{x}_k \;\; \phi(\overline{x}_1, \ldots, \overline{x}_k)$$

The truth of $\tau$ is a $\mathbf{QBF}_2$ question of an input of length bounded by $m_k$. By Corollary 5.3 we can compute the truth of $\tau$ in $\Pi_2^{n^{1+o(1)}}$. Using a variation of Lemma 2.2, we can reduce this question to the truth of a formula

$$\forall\, \overline{y}_1 \exists\, \overline{y}_2 \sigma(\overline{x}_1, \ldots, \overline{x}_{k-2}, \overline{y}_1, \overline{y}_2)$$

of size bounded by $n^{1+o(1)}$.

The truth of the formula $\psi_k$ is equivalent to the truth of

$$\exists\, \overline{x}_1 \forall\, \overline{x}_2 \ldots \forall\, \overline{x}_{k-2} \forall\, \overline{y}_1 \exists\, \overline{y}_2 \sigma(\overline{x}_1, \ldots, \overline{x}_{k-1}, \overline{y}_1, \overline{y}_2)$$

which is now a formula with $k-2$ alternations.

The case for $k$ odd is handled similarly.

As in Section 4.1 we recurse and get a polynomial-size $\mathbf{QBF}_2$ equivalent to the original $\mathbf{QBF}_{s(n)}$ size formula.

If **SAT** has log-space uniform $\mathbf{NC}^1$ circuits then $\mathbf{NP} = \mathbf{L}$ which implies $\Sigma_2^p = \mathbf{L}$. By Lemma 4.2, we have $\mathbf{L} \subseteq \Sigma_{O(1)}^{\mathbf{LIN}}$. This gives us

$$\Sigma_{s(n)}^{n \log n} \subseteq \Sigma_2^p = \mathbf{L} \subseteq \Sigma_{O(1)}^{\mathbf{LIN}}.$$

We get a contradiction by a simple diagonalization that creates a language in $\Sigma_{s(n)}^{n \log n} - \Sigma_{O(1)}^{\mathbf{LIN}}$. □

Note that no nonlinear lower bound on **SAT** is known for nonuniform $\mathbf{NC}^1$ circuits.

Using a similar proof, we can get lower bounds for uniform branching programs (see [BNS92]).

**Corollary 5.4** **SAT** *cannot be computed on log-space uniform branching programs of size* $n^{1+o(1)}$.

# 6 NP versus NL

Lemma 4.2 gives an immediate separation of nondeterministic logarithmic space and unbounded levels of the polynomial-time hierarchy.

**Theorem 6.1** *For any unbounded function* $s(n)$, **NL** *is strictly contained in* $\Sigma_{s(n)}^p$.

**Proof:** By Lemma 4.2, $\mathbf{NL} \subseteq \Sigma_{O(1)}^{\mathbf{LIN}}$. Theorem 6.1 follows by straightforward diagonalization. □

We know that $\mathbf{P} = \mathbf{NP}$ implies that any constant level of the polynomial-time hierarchy collapses to **P**. However when one examines the proof of this statement one discovers the exponent of the running time of the polynomial-time algorithm for $\Sigma_k^p$ languages increases as a function of $k$.

Suppose one could prove a stronger collapse, one where the exponent of the running time does not depend on the level of the hierarchy. This would allow us to collapse $\Sigma_{s(n)}^p$ to **P** for some nonconstant $s(n)$ and thus separate **NP** from **NL**. However, this direction would require nonrelativizing techniques.

**Theorem 6.2** *For every monotone unbounded function* $s(n)$ *computable in time polynomial in* $n$, *there exists an oracle* $A$ *such that* $\mathbf{P}^A = \mathbf{NP}^A$ *but* $\mathbf{P}^A \neq \Sigma_{s(n)}^{p,A}$.

Theorem 6.2 follows immediately from the following stronger theorem that shows that Lemma 4.1 is nearly tight in relativized worlds.

**Theorem 6.3** *For every monotone unbounded function $s(n)$ computable in time polynomial in $n$ and every constant $\epsilon > 0$, there exists an oracle $A$ such that* $\mathbf{SAT}^A \subseteq \mathbf{DTIME}^A[n^{1+\epsilon}]$ *but* $\mathbf{P}^A \neq \Sigma^{p,A}_{s(n)}$.

This proof builds on techniques used by Ko [Ko89] who presented a relativized world where $\mathbf{P} = \mathbf{NP} \neq \mathbf{PSPACE}$ which in turn built on work by Håstad [Hås89]. In particular we make use of the following bounds for parity.

**Lemma 6.4 (Håstad)** *For sufficiently large $m$ and $d$, no depth $d$ circuit of size $2^{m^{1/2d}}$ can compute parity of $m$ input variables. However, parity can be computed on a size $m2^{m^{1/(d-1)}}$ circuit.*

**Proof of Theorem 6.3:** We will construct $A$ to guarantee that $\mathbf{SAT}^A \in \mathbf{DTIME}^A[n^{1+\epsilon}]$. We do this by guaranteeing that for all formula $\phi$ of length $n$,

$$\phi \in \mathbf{SAT}^A \Leftrightarrow 1^{n^{1+\epsilon}-n-1}0\phi \in A \tag{3}$$

Without loss of generality, assume that $s(n) = O(\log n)$. Let $B_n$ be the lexicographically first $n^{s(n)-1}$ strings of length $n$ that begin with a 0.

To diagonalize we consider the following language:

$$L(A) = \{1^n \mid |A \cap B_n| \text{ is odd.}\}$$

First we argue that for every $A$, $L(A) \in \Sigma^{p,A}_{s(n)}$. By Lemma 6.4, for every $m$ and $d$, the parity of $m$ bits can be computed by a depth $d$ size $m2^{m^{1/(d-1)}}$ circuit of $\wedge$ and $\vee$ gates with negations only on the inputs. The parity of $n^{s(n)-1}$ bits can be computed with a circuit $C$ of size $n^{s(n)-1}2^n$ and depth $s(n)$. Consider the input variables as determining whether each string of $B_n$ is in $A$. The $\Sigma^{p,A}_{s(n)}$ machine can then determine whether $|A \cap B_n|$ is odd by simulating $C$ using $\forall$ states for the $\wedge$ gates and $\exists$ states for the $\vee$ gates.

However, Lemma 6.4 is very tight and we will use this to diagonalize against polynomial-time machines.

Let $M_1, M_2, \ldots$ be an enumeration of polynomial-time oracle Turing machines such that $M_i$ runs in time bounded by $n^i$. For each machine $M_i$ and for some $n$ we will set the strings of length $n$ in $A$ so that $L(M_i^A)$ and $L(A)$ disagree on $1^n$.

Stage $i$:

Fix $n$ such that $s(n) > 4i\ln(i)/\epsilon + 1$ and $n$ is much bigger than any strings sets in previous stages. For $\phi$ of length less than $n$, set $A$ to properly encode Equation (3). Set $A$ to zero for all the other strings of length less than $n$.

Consider the polynomial-time computation of $M_i^A$. We can consider the computation of $M_i^A$ as a circuit $C$ of size $2^{n^i}$ over variables representing whether strings of length at most $n^i$ are in $A$. We wish to convert this circuit to another one that depends only on the variables relating to $B_n$.

For strings of length between $n$ and $n^i$, not in $B_n$ and not used in Equation (3), set them to zero in $A$, i.e., put them all in $\overline{A}$. Suppose $C$ contains the variable $1^{n^{1+\epsilon}-n-1}0\phi$. By construction, $|\phi| \geq n$. We can replace this variable by simulating whether $\phi \in \mathbf{SAT}^A$. We use an $\vee$ of size $2^{|\phi|}$ to guess the possible satisfying assignments and an $\wedge$ of size $|\phi|$ to check the assignment over the

strings of $A$ queried by $\phi$ for that potential satisfying assignment. Note these variables represent strings of length less than $|\phi|$.

Replace all of the strings not queried in $B_n$ this way. This adds two to the depth of the circuit but now every variable representing a string of length $m$ is replaced by one representing a string of length $m^{1/1+\epsilon}$. If we repeat this process $\ln(i)/\epsilon$ times then all variables of Equation (3) represent strings of length less than $n$ so are all previously encoded.

Thus we have a circuit $C'$ of size $2^{n^i}$ and depth $2\log i/\epsilon$ over the $n^{s(n)-1}$ variables representing whether string in $B_n$ are in $A$. By Lemma 6.4, $C'$ cannot compute parity. Fix a setting of the variables such that $C'$ and parity disagree and set $A$ accordingly. This guarantees that $L(M_i^A) \neq L(A)$. □

Theorem 6.1 suggests another attack on the $\mathbf{NP} \neq \mathbf{NL}$ problem. From Theorem 6.1 we have $\Sigma^p_{s(n)} \not\subseteq \mathbf{NSPACE}[\log n]$ for any unbounded $s(n)$. The following lemma says that any improvement of the space bound will yield $\mathbf{NP} \neq \mathbf{NL}$.

**Lemma 6.5** *If* $\mathbf{NP} = \mathbf{NL}$ *then for any* $f(n) = \omega(\log n)$*, there exists an unbounded function* $s(n)$ *such that* $\Sigma^p_{s(n)} \subseteq \mathbf{NSPACE}[f(n)]$.

**Proof:** Assume $\mathbf{NP} = \mathbf{NL}$. We then have $\mathbf{P} = \mathbf{NP}$ so $\mathbf{SAT} \in \mathbf{DTIME}[n^c]$ for some $c$. By techniques similar to the proof of Lemma 4.1 we have $\Sigma^p_{s(n)} \subset \mathbf{DTIME}[n^{d^{s(n)}}]$ for some other constant $d$.

Pick $s(n)$ such that $d^{s(n)} \log n < f(n)$. We then have $\Sigma^p_{s(n)} \subseteq \mathbf{DTIME}[2^{f(n)}]$.

By assumption we also have $\mathbf{P} = \mathbf{NL}$ so $\mathbf{DTIME}[n] \subseteq \mathbf{NSPACE}[\log n]$. By padding we have $\mathbf{DTIME}[2^{f(n)}] \subseteq \mathbf{NSPACE}[f(n)]$. The lemma follows. □

Theorem 6.1 also gives limitations to improving Toda's [Tod91] theorem. Toda showed that any constant level of the polynomial-time hierarchy can be reduced to the complexity class $\mathbf{PP}$. We show that extending his result would yield a nice separation.

**Corollary 6.6** *If for any unbounded monotone function* $s(n)$ *computable in time polynomial in* $n$*,* $\Sigma^p_{s(n)} \subseteq \mathbf{P}^{\mathbf{PP}}$ *then* $\mathbf{PP}$ *strictly contains* $\mathbf{NL}$.

**Proof:** Suppose the assumption is true and $\mathbf{NL} = \mathbf{PP}$. We then have $\mathbf{NL} = \mathbf{P} = \mathbf{PP}$ so

$$\Sigma^p_{s(n)} \subseteq \mathbf{P}^{\mathbf{PP}} \Rightarrow \Sigma^p_{s(n)} \subseteq \mathbf{P}^{\mathbf{P}} \Rightarrow \Sigma^p_{s(n)} \subseteq \mathbf{P} = \mathbf{NL}$$

contradicting Theorem 6.1. □

# 7 Conclusions

Since we expect that satisfiability requires exponential time and linear space the bounds we get are likely far from optimal. Improving the space bound a little bit in Theorem 3.1 would give us $\overline{\mathbf{SAT}} \notin \mathbf{NTIME}[n^{1+o(1)}]$ without any space constraints. Also improving the lower bound for time to $n^{1+\epsilon}$ even with logarithmic space seems quite difficult.

Our results only hold in the uniform setting. In the nonuniform setting we will be unable to diagonalize. Pushing through some of our results to nonuniform classes or a weaker uniformity condition remains open.

We believe that this paper has given hope to the idea that perhaps classes like $\mathbf{NP}$ and $\mathbf{NL}$ can be separated using simple techniques like diagonalization. We should not discard diagonalization

as a non-"natural" proof technique. Rather we should see what diagonalization will do for us in the light of our current understanding of complexity classes.

## Acknowledgments

Much of the research of this paper was motivated by the author's research with Harry Buhrman and Leen Torenvliet on autoreducibility [BFT95]. The author thanks Harry Buhrman for many initial discussions on this paper and for allowing the inclusion of Theorem 5.1 and its proof in this paper.

The author thanks Eric Allender for pointing out the connections to Nepomnjaščiĭ's theorem. Ken Regan was very helpful with simulations of RAMs by Turing machines.

The author also thanks Avi Wigderson, Richard Beigel, Noam Nisan, Sam Buss, Alan Woods, Dieter van Melkebeek, Peter van Emde Boas, Steven Rudich, Manindra Agrawal and Vikraman Arvind for very useful discussions about these results. Leen Torenvliet, Harry Buhrman, Peter van Emde Boas and two very helpful anonymous referees provided important proof readings of the paper.

## References

[Ben62] J. Bennett. *On Spectra*. PhD thesis, Princeton University, 1962.

[BFT95] H. Buhrman, L. Fortnow, and L. Torenvliet. Using autoreducibility to separate complexity classes. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 520–527. IEEE, New York, 1995.

[BNS92] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, October 1992.

[BS90] R. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 14, pages 757–804. North-Holland, 1990.

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

[Cob66] A. Cobham. The recognition problem for the set of perfect squares. In *Conference Record of the Seventh Annual Symposium on Switching and Automata Theory*, pages 78–87. IEEE, New York, 1966.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158. ACM, New York, 1971.

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

[DG84] P. Dúriš and Z. Galil. A time-space tradeoff for language recognition. *Mathematical Systems Theory*, 17(1):3–12, April 1984.

[EP95] J. Edmonds and C. Poon. A nearly optimal time-space lower bound for directed *st*-connectivity on the NNJAG model. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 147–156. ACM, New York, 1995. To appear in SIAM Journal on Computing.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the European Association for Theoretical Computer Science*, 52:229–244, February 1994.

[GJ79] M. Garey and D. Johnson. *Computers and intractability. A Guide to the theory of NP-completeness.* W. H. Freeman and Company, New York, 1979.

[GJ93] J. Goldsmith and D. Joseph. Relativized isomorphisms of NP-complete sets. *Computational Complexity*, 3:186–205, 1993.

[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, 1989.

[GS90] Y. Gurevich and S. Shelah. Nondeterministic linear-time tasks may require substantially nonlinear deterministic time in the case of sublinear work space. *Journal of the ACM*, 37(3):674–687, July 1990.

[Hås89] J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, 1989.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS66] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Mass., 1979.

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

[Kar86] M. Karchmer. Two time-space tradeoffs for element distinctness. *Theoretical Computer Science*, 47(3):237–246, 1986.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on the Theory of Computing*, pages 302–309. ACM, New York, 1980.

[Ko89] K. Ko. Relativized polynomial time hierarchies having exactly $k$ levels. *SIAM Journal on Computing*, 18:392–408, 1989.

[Lev73] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems: in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973. Corrected English translation in [Tra84].

[Nep70] V. Nepomnjaščiĭ. Rudimentary predicates and Turing calculations. *Soviet Mathematics–Doklady*, 11(6):1462–1465, 1970.

[PF79] N. Pippenger and M. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

[PPST83] W. Paul, N. Pippenger, E. Szemerédi, 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, New York, 1983.

[Rei90] R. Reischuk. *Einführung in die Komplexitätstheorie*. Teubner, Stuttgart, Germany, 1990.

[Rob91] J. Robson. An $O(T \log T)$ reduction from RAM computations to satisfiability. *Theoretical Computer Science*, 82:141–149, 1991.

[RR97] A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.

[Sav70] W. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–192, 1970.

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

[Smu61] R. Smullyan. *Theory of Formal Systems*, volume 47 of *Annals of Mathematical Studies*. Princeton University Press, 1961.

[Sv88] C. Slot and P. van Emde Boas. The problem of space invariance for sequential machines. *Information and Computation*, 77(2):93–122, 1988.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tra84] R. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Woo86] A. Woods. Bounded arithmetic formulas and Turing machines of constant alternations. In *Logic Colloquium '84*, volume 120 of *Studies in logic and the foundations of mathematics*, pages 355–377. North-Holland, 1986.

[Wra78] C. Wrathall. Rudimentary predicates and relative computation. *SIAM Journal on Computing*, 7:194–209, 1978.

[Yao94] A. Yao. Near-optimal time-space tradeoff for element distinctness. *SIAM Journal on Computing*, 23(5):966–975, October 1994.