Structure of Complexity Classes: Separations, Collapses, and Completeness

Lane A. Hemachandra*
Department of Computer Science
Columbia University
New York, NY 10027, USA

Abstract

During the last few years, unprecedented progress has been made in structural complexity theory; class inclusions and relativized separations were discovered, and hierarchies collapsed. We survey this progress, highlighting the central role of counting techniques. We also present a new result whose proof demonstrates the power of combinatorial arguments: there is a relativized world in which UP has no Turing complete sets.

1 Introduction

Two years ago, the hunting season opened. Quickly, the strong exponential hierarchy fell, followed by the linear-space, logspace, and logspace oracle hierarchies. Soon the LBA problem, a venerable precursor of $P = \neg\neg \neg\neg \neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\neg\n

*Supported by NSF grant CCR-8809174 and a Hewlett-Packard Corporation equipment grant.

complexity theory, and identify the reason for this sudden progress. For many years, NP was viewed as a mysterious black box. During the last few years, we have started to open that box. Counting and combinatorial techniques have been used to explore, and exploit, the strengths and weaknesses of nondeterministic computation.

We survey recent progress on collapsing hierarchies, establishing complexity class containments, and proving relativized separations and non-completeness results.

The second part of this paper—Section 3—presents a result that extends the use of counting arguments to a new area—proving Turing non-completeness. We show that there is a relativized world in which the cryptographic class UP, unique polynomial time, has no Turing complete sets. This extends a long line of research on completeness that has been pursued by Ambos-Spies, Hartmanis, Immerman, and Sipeer [Sip82,HI84,HH88, Amb86]. The goal of this research is to understand the structure of complexity classes by determining which classes have complete sets under which reducibilities.
2 Recent Progress in Structural Complexity Theory

2.1 Collapsing Hierarchies and Class Containments

The most noteworthy recent progress in structural complexity theory is the sudden collapse of complexity hierarchies. The most surprising aspect of these collapses is the simplicity and elegance of the techniques used. The LBA problem, which remained open for twenty-five years, has a four-page resolution.

The key technique in these collapses has been the use of census functions—functions that count. Usually, census functions count the number of elements in prefixes of a set having certain properties. Census functions are not new; Mahaney's proof that NP has no sparse complete set unless P = NP is based on the use of census information [Mah82].

We start by presenting two of the earlier of recent uses of census functions to explore class inclusions. The first shows the relationship between Turing reductions and truth-table reductions, and the second strengthens a long line of "small circuit [KL80]" results.

\(P_{\text{NP}^{\text{NP}}}^{\text{NP}}\) indicates the class of languages accepted by polynomial-time Turing machines that make \(O(\log n)\) calls to an NP oracle. \(P_{\text{TURP}}^{\text{NP}}\) indicates the class of languages that are polynomial-time truth-table reducible to NP [LLS75].

**Theorem 2.1 [Ham87c]**

\[P_{\text{TURP}}^{\text{NP}} = P_{\text{NP}^{\text{NP}}}^{\text{NP}}\]

The proof simply uses binary search to find, in \(O(\log n)\) queries to NP, the number \(m\) of queries of the truth-table reduction that receive the answer "yes," and then uses one further NP query to guess and check which \(m\) queries are answered "yes" and determine if the truth-table system accepts.

**Proof Sketch** \(P_{\text{TURP}}^{\text{NP}} \leq_{\text{NP}^{\text{NP}}} \text{NP}\) means there is a polynomial-time machine that answers "\(z \in S\)" by making queries to SAT [GJ79], such that the queries asked of SAT are independent of the answers received [LLS75]. To prove the \(\subseteq\) part, we have \(P\) perform binary search, using its NP oracle, to find the number of yes answers, and with one final query to the oracle have NP guess which queries receive yes answers and simulate the action of the truth-table reducer. The \(P\) part it trivial—the truth-table reducer asks all queries that might be formed by any of the \(n^{O(1)}\) possible sets of oracle answers of the run of \(P_{\text{TURP}}^{\text{NP}}\).

Detailed investigations of the interleaving of truth-table, oracle query, and boolean hierarchies can be found in [KSW86, AG87, Bie87].

The next example, due to Kadin, strengthens theorems of Karp, Lipton, Long, and Mahaney [KL80, Mah82, Lon82]. The Karp-Lipton "small circuits" theorem (so-called as the hypothesis is equivalent to "NP has small circuits") states:

**Theorem 2.2 [KL80]** If there is an \(S\) such that \(NP \subseteq P^S\), then \(NP^{NP} = PH\), where PH is the polynomial hierarchy.

For the case of simple sparse sets, this was extended by Mahaney.

\(^1\)A set \(A\) is sparse if it has only polynomially many elements of length at most \(n\). That is, \((\exists k)(\forall n)[\text{there are at most } n^k + k \text{ elements of length } \leq n \text{ in } A]\).

**Theorem 2.3** \(S \in NP\) such \(Kadin\) strengthened

**Theorem 2.4** \(S \in NP\) such \(Kadin\) strengthened

**Proof Sketch** number of elements a sparse \(S\) a \(P\) maci oracle to it is easy to calculate query (to go to a certain using Theorem proposed)

The first has functions by. Nonde defined as \(\bigcup_{S \subseteq \Gamma}\) states that implies

**Theorem 2.6** \(P_{\text{NP}} = \bigcup_{S \subseteq \Gamma}\)

This follows \(P_{\text{NP}} = \bigcup_{S \subseteq \Gamma}\) a census at; \(NP^{NP}\) compu answer to q Theorem 2.5 developed the \(P\) machine machine can qu
Theorem 2.3 [Hem86,c]

$\text{P} = \text{E}$, $\text{NP} = \text{UP} = \text{PSPACE} = \text{UP}^*$

The collapse immediately follows from the lemma that if $\text{NP}$ is contained in $\text{P}$, then $\text{P} = \text{NP} = \text{NP}^*$.

Proof: The collapse was first proven by Hemmert [Hem86]. The proof relies on the construction of an $\text{NP}$-complete language that is in $\text{P}$. This construction shows that if $\text{NP} \subseteq \text{P}$, then $\text{NP} = \text{P}$.

Theorem 2.4 [Kad87]

If there is a space-bounded search oracle that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: Let $\mathcal{A}$ be a $k$-query search oracle that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$. Then $\mathcal{A}$ can be simulated by a deterministic Turing machine $M$ that runs in $\text{SPACE}(f(n))$.

Theorem 2.5 [BG88]

Theorem 2.5 shows that $\text{LOG} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.5 relies on the construction of a language that is in $\text{LOG}$ but not in $\text{PSPACE}$. This construction shows that if $\text{LOG} \subseteq \text{PSPACE}$, then $\text{LOG} = \text{PSPACE}$.

Theorem 2.6 [Sze87,Jan93]

For any space-bounded $\text{SPACE}(f(n))$ language $L$, there is a $k$-query search oracle $\mathcal{A}$ such that $L$ can be computed by $\mathcal{A}$ in $\text{SPACE}(f(n))$.

Proof: The proof of Theorem 2.6 relies on the construction of a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.7 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.7 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.8 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.8 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.9 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.9 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.10 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.10 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.11 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.11 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.12 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.12 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.13 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.13 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.14 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.14 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.15 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.15 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.16 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.16 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.17 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.17 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.

Theorem 2.18 [Sze87,Jan93]

If there is a search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$, then $\text{NP} \subseteq \text{PSPACE}$.

Proof: The proof of Theorem 2.18 relies on the construction of a $k$-query search oracle $\mathcal{A}$ that outputs the $k$th bit of the $i$th element in $\text{SPACE}(f(n))$ for any $i$ and $k$.
used to show that parity polynomial time is powerful enough to contain FewP—the subset of NP languages that are accepted by machines that never have many accepting paths.

**Definition 2.8**

1. [PZ83] (Parity Polynomial Time) $\oplus P = \{L |$ there is a nondeterministic polynomial-time Turing machine $N$ such that $x \in L$ if and only if $N(x)$ has an odd number of accepting paths).

2. [All86] FewP = \{L | there is a nondeterministic polynomial-time Turing machine $N$ such that (1) $x \in L$ if and only if $N(x)$ has at least one accepting path and (2) $\exists k \forall x \exists \epsilon \exists \nu (x)[N(x) has at most $|x|^k$ accepting paths]]

3. [CH87] Few is the class of all languages $L$ such that there is a nondeterministic polynomial-time Turing machine $N$, a polynomial-time computable predicate $Q(\cdot, \cdot)$, and a polynomial $q(\cdot)$, such that (1) $x \in L$ if and only if $Q(x, ||N(x)||)$, and (2) $(\forall x)[||N(x)|| \leq q(|x|)]$, where $||N(x)||$ denotes the number of accepting paths of $N(x)$.

**Theorem 2.9 [CH87]** $\oplus P \supseteq$ Few.

**Corollary 2.10 [CH87]** $\oplus P \supseteq$ FewP.

A direct proof of the corollary is immediate.

Given a FewP machine, $N_x$, that on inputs of length $n$ never has more than $n^k + k$ accepting paths, we construct a new machine $N$, a nondeterministic machine with the parity acceptance mechanism, so that $N(x)$ has a path for each path of $N_x(x)$, and a path for each pair of paths of $N_x(x)$, ..., and a path for each $(n^k + k)$-tuple of paths of $N_x(x)$. Each path of $N(x)$ will accept and only if all of the paths that it represents in $N_x(x)$ are accepting paths. Since $\sum_{i \leq n^k + k} (i) = 2^n - 1$ is odd exactly when $j \neq 0$, it follows that FewP $\subseteq \oplus P$. The proof that Few $\subseteq \oplus P$ takes a bit more work, but follows the same line.

### 2.2 Separations

The previous section described a number of collapsing hierarchies and class containment. Hierarchy separations, or even class separations, remain elusive. However, a number of hierarchies have been separated in relativised worlds. These include the boolean hierarchy, the counting hierarchy [CGH+b], and, most importantly, the polynomial hierarchy.

**Definition 2.11**

1. **Boolean Hierarchy** [Wec85] $\Sigma^B_0 = \text{P}$, $\Sigma^B_i = \text{NP}$, $\Sigma^B_i = \{L | \exists L' \in \text{NP}(3L' \in \Sigma^B_i \forall L \in L' - L', i > 1)$.

2. **Polynomial Hierarchy** [Sto77] $\Sigma^P_0 = \text{P}$, $\Sigma^P_i = \text{NP}$, $\Sigma^P_i = \text{NP}^{\Sigma^P_{i-1}}$, for $i > 1$.

**Theorem 2.12 [CGH+a]**

1. There is a relativized world $A$ in which $\Sigma^B_0(A) \neq \Sigma^B_1(A) \neq \Sigma^B_2(A) \neq \cdots$.

2. For each $k$ there is a relativized world $A$ in which $\Sigma^B_{k-1}(A) \neq \cdots \neq \Sigma^B_k(A) = \Sigma^B_{k+1}(A) = \cdots$.

**Theorem 2.13 [Yao85,Ko88]**

1. There is a relativized world $A$ in which $\Sigma_0^P(A) \neq \Sigma_1^P(A) \neq \cdots$.

2. For each $k$ there is a relativized world $A$ in which $\Sigma^P_k(A) \neq \cdots \neq \Sigma^P_{k+1}(A) = \Sigma^P_{k+2}(A) = \cdots$.

**Corollary 2**

Categorical (i.e., every oracle $\text{P}^A = \text{NP}^A$)

**Theorem 2**

Machines $A$
complexity classes—NP, coNP, PSPACE, etc.—have many-one complete sets that help us study them.

Sipser noted, however, that some classes may lack complete sets [Sip82]. His paper sparked much research into which classes have complete languages, and what strengths of completeness results (e.g., many-one or Turing) can be obtained. Of course, if \( P = \text{PSPACE} \), then all classes between \( P \) and \( \text{PSPACE} \) have many-one complete languages. Thus, incompleteness results are typically displayed in relativized worlds [BGS75, Sip82].

Sipser showed that there are relativized worlds in which \( R \) and \( \text{NP} \cap \text{coNP} \) lack many-one complete languages. Hartmanis and Hemachandra showed a relativized world in which \( \text{UP} \)—unique polynomial time (Section 3.1)—lacks many-one complete languages, and noted that if \( \text{UP} \) does have complete languages then \( \text{UP} \) has complete languages with an unusually simple form—the intersection of \( \text{SAT} \) with a set in \( P \) [HH86].

One way of strengthening the above theorems would be to show that these classes lack complete sets even with respect to reducibilities more flexible than many-one reductions, e.g., \( k \)-truth-table, positive truth-table, truth-table, and ultimately Turing reductions [LLS75]. Hartmanis and Immerman, exploiting an insightful characterization of Kowalsky [Kow84], showed that \( \text{NP} \cap \text{coNP} \) has many-one complete languages if and only if it has Turing complete languages [H185]. An elegant generalization of their result by Ambos-Spies shows that for any class \( C \) closed under Turing reductions, \( C \) has Turing complete sets if and only if \( C \) has many-one complete sets [Amb86].

In particular, it follows from the result of Sipser [Sip82] that there is a relativized world \( A \) in which \( \text{NP}^A \cap \text{coNP}^A \) lacks Turing complete sets [HH85, Amb86]. Similarly, since \( \text{P}^{\text{BPP}} = \text{BPP} \) [Zac86], from [HH86]'s proof that \( \text{BPP} \) lacks many-one complete sets in some relativized worlds it follows that it also may lack Turing complete sets.

Theorem 2.20 There is a relativized world \( A \) in which \( \text{BPP}^A \) lacks Turing complete sets.

However, Ambos-Spies's result does not apply to \( \text{UP} \) or any other class not known to be closed under Turing reductions. Furthermore, the technique used to show that \( \text{UP} \) may lack many-one complete languages was an indirect proof via the contradiction of an enumeration condition that characterized the existence of many-one complete languages [HH86]—and does not generalize to the case of Turing completeness.

Section 3 constructs an oracle \( A \) for which \( \text{UP}^A \) contains no Turing complete sets. Our proof exploits the limited combinatorial control of nondeterministic machines to trivialize or corrupt candidates for Turing completeness. This approach extends our theme: the exploitation of the combinatorics of the nondeterministic acceptance mechanism.

It follows immediately from our proof that there is a relativized world \( A \) in which \( \text{UP}^A \) lacks complete languages under all reducibilities more restrictive than Turing reductions.

### 3 Does Com

#### 3.1 Defi

**Definition 3** (Unique Poly) a nondeterministic machine \( N \) such computation path, \( W \) input basical.

\( \text{UP} \) the class of in machine unique NP machine the computer path (i.e., \( P \) say \( L \in \text{UP} \)).

Recently, in both cry theory. In have shown only if \( P \), range is in \( \text{NP} \) in theory that one-w complexity the there exist was recent!

\(^2\text{A function } |z| \text{ (|GS84|) is a } t \text{ polynomial } f^{-1} \text{ (which } E^f \text{ is not).} \)

\(^4\text{Range} \)
3. Does UP have Turing Complete Languages?

3.1 Definitions

Definition 3.1 [Val78]
(Unique Polynomial Time) UP = \{L | there is a nondeterministic polynomial time Turing machine N such that L = L(N), and for all x, the computation of N(x) has at most one accepting path\}. We say that a machine N that for every input has at most one accepting path is categorical.

UP captures the power of uniqueness; UP is the class of problems that have (on some NP machine) unique witnesses. That is, if there is an NP machine N accepting L and for every input x the computation N(x) has at most one accepting path (i.e., N is a categorical machine), then we say L ∈ UP.

Recently, UP has come to play a crucial role in both cryptography and structural complexity theory. In cryptography, Grollmann and Selman have shown that one-way functions exist if and only if P ̸= UP, and one-way functions whose range is in P exist if and only if P ̸= UP ∩ coUP. Thus, we suspect that P ̸= UP because we suspect that one-way functions exist. In structural complexity theory, a conjecture that "P ̸= UP =⇒ there exist non-p-isomorphic NP-complete sets" was recently refuted in a relativized world [HH87].

For background, we first define Turing reductions and completeness in the real (unrelativized) world.

Definition 3.2
1. S₁ ≤P S₂ if S₁ ⊆ P² [GJ79].
2. L is ≤P-complete for UP if L ∈ UP and every set in UP Turing reduces to L (i.e., (∀S ∈ UP)(S ≤P L)).

If we wish to discuss Turing completeness in relativized worlds, we must address the key question: are the Turing reductions allowed access to the oracle? Definitions 3.3.2 and 3.3.3 answer this question "yes" and "no," respectively.

Definition 3.3
1. S₁ ≤P A S₂ if S₁ ⊆ P² A.
2. L is ≤P A-complete for UP A if [L ∈ UP A and (∀S ∈ UP A)(S ≤P A L)].
3. L is ≤P-complete for UP A if [L ∈ UP A and (∀S ∈ UP A)(S ≤P L)].

We suggest that Definition 3.3.2 above is the natural notion of relativized Turing completeness. Adopting it, we prove that there is a relativized world in which UP A has no ≤P A-complete sets. However, for purposes of completeness results, the different notions of relativized Turing reductions stand or fall together.

Lemma 3.4 For any oracle A: [UP A has ≤P A-complete sets if and only if UP A has ≤P A-complete sets].

This is true since if B is ≤P A-complete for UP A, then B ⊗ A is ≤P A-complete for UP A. The analog of Lemma 3.4 for many-one reductions was proven by Sipser [Sip82].
The difference between Definitions 3.3.2 and 3.3.3 is exactly the difference between "full" (3.3.2) and "partial" (3.3.3) relativization discussed in [KMR86] and [Rog87, Section 9.3]. [KMR86] describes how this distinction has had a crucial effect on recent research asking if all NP-complete sets are polynomially isomorphic [Kur83,GJ86,HH87]. However, Lemma 3.4 indicates that in our study of Turing completeness, we need not be concerned with the distinction.

3.2 A Relativized World in Which UP Lacks Turing Complete Sets

This section sketches the construction of an oracle for which $UP^A$ has no Turing complete sets. It follows immediately that $UP^A$ lacks complete sets with respect to reductions more restrictive than $\leq_{\text{T}}^A$, such as truth-table reductions [Lls75], bounded truth-table reductions [Lls75], etc.

Theorem 3.5 There is a recursive oracle $A$ such that $UP^A$ contains no $\leq_{\text{T}}^A$-complete sets.

Corollary 3.6 There is a recursive oracle $A$ such that $UP^A$ contains no:

1. $\leq_{\text{T}}^A$-complete languages.
2. truth-table complete languages.
3. bounded truth-table complete languages.
4. [HH86] $\leq_{\text{m}}^A$ or $\leq_{\text{b}}^A$-complete languages.

Let $(N_i)$ be a standard enumeration of nondeterministic polynomial-time Turing machines and let $(M_i)$ be a standard enumeration of deterministic polynomial-time Turing machines. The idea of the proof is as follows. We wish to show that for no $L \in UP^A$ is $UP^A \subseteq P^L$, which suffices by Lemma 3.4. Each $L$ in $UP^A$ is, by definition, accepted by a categorical machine $(L = L(N_i^A), N_i^A$ categorical). Our goal is to show that for each $i$, either

1. $N_i^A$ is not categorical, or
2. $(\exists L_i)[L_i \in UP^A$ and $L_i \notin P^{L(N_i^A)}]$.

The second condition says that some $UP^A$ language does not Turing reduce to $L(N_i^A)$. That is, every Turing reduction fails on some value. Thus it certainly suffices to show that for all $i$, either

1. $N_i^A$ is not categorical or
2. (a) $(\forall j)(\exists z)[z \in L_i \iff z \in L(M_j^{L(N_i^A)})]$,
   where $L_i = \{ z | f(n) = p_i \} \wedge (\exists y)(|y| = n \wedge y \in A$] and $p_i$ is the $i$th prime, and
   (b) $L_i \in UP^A$.

Note that we have specified $L_i$.

Briefly put, for each $i < j$, we seek to find a way of extending the oracle to make $N_i^A$ noncategorical. Failing this, we argue that we can choose our oracle in such a way as to determine the answers to all oracle queries made by $M_j$, and still have the flexibility to diagonalize against $L_i$. The crucial step is a combinatorial argument that categorical machines which don't trivially accept must reject on an overwhelming number of oracle extensions.

Proof Sketch for Theorem 3.5

We wish to show that there is a relativized world $A$ where $UP^A$ has no Turing complete languages, i.e.,

$$(\forall L \in UP^A)(\exists L' \in UP^A)[L' \notin P^{L'} L].$$

By Lemma 3.4 it

$$(\forall L \in UP^A)(\exists L' \in UP^A).$$

Since each language has at least one categorical language, this says that

$$(\forall i)[L_i \in UP^A]$$

This says:

$$(\forall i)[L_i \in UP^A]$$

Let requirement $R_{i,j}$:

$$(\exists z)[z \in L_i \iff z \in L_j],$$

where $L_i = \{ z | f(n) = p_i \} \wedge (\exists y)(|y| = n \wedge y \in A$] and $p_i$ is the $j$th prime.

Note that we have specified $L_i$.

Let us consider the case of extending the oracle to make $N_i^A$ noncategorical. Failing this, we argue that we can choose our oracle in such a way as to determine the answers to all oracle queries made by $M_j$, and still have the flexibility to diagonalize against $L_i$.

The crucial step is a combinatorial argument that categorical machines which don't trivially accept must reject on an overwhelming number of oracle extensions.

Proof Sketch for Theorem 3.5

We wish to show that there is a relativized world $A$ where $UP^A$ has no Turing complete languages, i.e.,

$$\forall L \in UP^A \exists L' \in UP^A L' \notin P^{L'} L.$$
By Lemma 3.4 it suffices to show that
\[(\forall L \in \text{UP}^A)(\exists L' \in \text{UP}^A)[L' \geq_L L].\]

Since each language in \(\text{UP}^A\) is accepted by at
least one categorical machine, we can equivalently
show that
\[(\forall i)[N^A_i \text{ is noncategorical}] \lor
(\exists L_i \in \text{UP}^A)[L_i \ngtr_L L(N^A_i)].\]

This says that:
\[(\forall i)[N^A_i \text{ is noncategorical}] \lor
(\exists L_i \in \text{UP}^A)[L_i \not\in P(L(N^A_i))].\]

Let requirement \(R_{i+1}\) be
\[R_{i+1} : (3x)[x \in L_i \iff x \in L(M^2_i, A_i)].\]
where
\[L_i = \{1^j \mid (3k \not\in n = p_k) \land (3y)[y = n \land y \in A]\},\]
and \(p_k\) is the \(k\)th prime.

Note that we satisfy (*) if we can satisfy for
all \(i\) the following, which simply (1) specifies the
\(L_i,\) (2) uses the fact that \(P^X = \cup_i L(M^2_i),\) where
\(L_i\) is a standard enumeration of polynomial
machines and without loss of generality \(M_i\) runs in
time \(n^i + 1,\) and (3) notes that differing languages
must differ on some specific element.

1. \(N^A_i\) is noncategorical, OR

2. \((\forall i)[R_{i+1}\text{ is satisfied}] \land (L_i \in \text{UP}^A).\)

Our construction will go by stages. In stage
\(< i, j >\) we will either satisfy \(R_{i+1}\) or know that
\(N^A_i\) is noncategorical.

Initially set \(A_{<i,j>} := \emptyset.\) We'll have \(A =
U_{<i,j>} A_{<i,j>}\)

Stage \(< i, j >:\): If \(N^A_i\) has already been made
noncategorical, skip this stage and set \(A_{<i,j>} :=
A_{<i,j>-1}\). Otherwise, choose a huge integer \(n\) that
is a power of the \(i\)th prime (i.e., so \(3k \not\in n = p_k\))
and is much larger than any previously touched length.

A legal extension of \(A_{<i,j>}\) will be one that does
not touch any string shorter than \(n,\) and that adds
strings to \(A_{<i,j>}\) only at lengths that are powers
of \(p_i.\)

If there is a legal extension \(\hat{A}\) of \(A_{<i,j>-1}\) such
that for some \(y \exists |y| \leq (n^i + j)^i + i\) we have
that \(N^A_i(y)\) is noncategorical,\(^6\) then choose two
accepting paths of \(N^A_i(y)\) and set \(A_{<i,j>}\) to be
\(\hat{A}_{<i,j>-1}\) augmented to agree with \(\hat{A}\) on all strings
queried on those two paths and go to the next stage.

Otherwise, we've failed to make \(N^A_{<i,j>-1}\)
noncategorical, so we must try to satisfy requirement
\(R_{i+1} :\). Our goal is to fix the behavior of
\(M_i^{(k)(A_i)}(1^n)\) and yet have enough flexibility left
to ensure that \(1^n \in L_i\) or \(1^n \not\in L_i\) as we wish.
Thus we can diagonalize to ensure that \(R_{i+1}\) is
satisfied. Furthermore, unless we discover a way
of making \(N^A_i\) noncategorical, we'll put only one
string into \(A\) at each length \(p_i^n\) thus ensuring that
\(L_i \in \text{UP}^A.\)

We simulate the behavior of \(M_i^{(k)(A_i-1)}(1^n),\)
and each time \(M_i\) makes an oracle query, we take
action to control that query.

We use \(T_i\) to indicate the set of strings of length
\(n\) to which the \(i\)th query is oblivious. We'll eventually
show that the \(T_i\) are all so huge that there
must be at some string in \(T_{i+1} \cap T_i \cap \cdots \cap T_{j+1}\).

Action for the first query:

Run \(M_i^{(k)(A_i-1)}(1^n)\) until \(M_i\) asks its first
oracle query, \(q_1.\)
Case 1: \(N^A_{<i-1,j-1} q_1\) accepts. Freeze the
elements along the accepting path, and proceed to
action for the second query. Set \(T_1 = \{x \mid |x| = n\}
\]

\(^6\)We could remove the bound on \(y\)'s size, but
this would make the construction nonrecursive.
and $s$ has not just been frozen).

Case 2: $N^{A,x_i,n-1}(q_1)$ rejects. We argue that there are a huge number of strings of length $n$ that can be added to the oracle which will not change this rejection. Let $T_1 = \{s \mid n = |s|\}$ and the membership of element $s$ in $A$ has not yet been determined and $N^{A,x_i,n-1}(q_1)$ rejects. Let $t_1 = |T_1|$. By Lemma 3.7, $t_1 \geq 2^n - 8(n^2 + j)^i + i^2$.

Action for the 1th query:

At this point, we've already determined the answers to the first $l - 1$ oracle queries. We proceed analogously to the action for the first query (except we respect—and use $k$ of Lemma 3.7 to account for—strings frozen from case 1 of earlier queries, which causes the bounds on $T_i$ to weaken slightly as $i$ grows).

End of query sequence.

Our goal was to fix the responses to the queries while leaving ourselves enough freedom to make $1^* \in L$ or $1^* \notin L$, as we like. We can now do that.

Each $T_i$ is easily of size $\geq 2^n - n^{2i}$. Thus, since each $T_i$ is a subset of the set of length $n$ strings, and since there are at most $n^i + j$, there is some length $n$ string $s$ in $T_1 \cap T_2 \cap \cdots \cap T_{i+1}$. By the definition of the $T_i$'s, adding this string to $\text{Acceptors}_{1,2,\ldots,i-1}$ will have no effect on any of the oracle responses. That is, $M^{N^{A,x_i,n-1}(1^*)}_{2}(A) \geq 1$ accepts if and only if $M^{N^{A,x_i,n-1}(1^*)}_{2}(A) \geq 1$ accepts. If these do accept, choose $i \notin A$. Thus, $1^* \notin L$, but $1^* \notin L(M^{A,x_i,n-1}(1^*))$, so requirement $R_{k,1,2,\ldots,i,1}$ has been satisfied. On the other hand, if $M^{N^{A,x_i,n-1}(1^*)}_{2}(A) \geq 1$ rejects, choose $i \in A$. Thus, $1^* \notin L$, but $1^* \notin L(M^{A,x_i,n-1}(1^*))$, so requirement $R_{k,1,2,\ldots,i,1}$ has been satisfied.

End of Stage $<1, j>$. Note that if we never find a way of making $N^{A,x_i,n-1}(q_1)$ accept, then $L \in \text{UP}^A$ (because the above procedure puts in only one string, $s$, at each length $i$-length to the oracle $L_i$ and $\forall j$ requirement $R_{k,1,2,\ldots,i,1}$ is satisfied). Thus ($\ast \ast \ast$) is satisfied.

On the other hand, if we do find a way of making $N^{A,x_i,n-1}(q_1)$ accept, then ($\ast \ast \ast$) is satisfied.

Thus we have met requirements that are, by the discussion following Corollary 3.1, sufficient to ensure that $\text{UP}^A$ has no $\leq_{\text{P}}^A$-complete languages.

End of Stage $<1, j>$. In the proof, we referred to the following lemma. Loosely, what the lemma says is that if a machine tries to be categorical on all possible oracle queries and it rejects on the empty string, then it must also reject on an overwhelmingly large proportion of strings that add exactly one string.

Lemma 3.7 Let $N^{A,x_i,n-1}(q_1)$ be a nondeterministic Turing machine that runs in $\text{NTIME}[n^i + i]$. Suppose $N^{A,x_i,n-1}(q_1)$ rejects. $A$ has no strings of length $|x|$ and $k$ strings of length $|x|$ have been designated as forbidden from being added to the oracle $A$. Let $\text{Rejectors}_{x} = \{y \mid N^{A,x_i,n-1}(q_1) \geq 1\}$ and $|y| = |x|$ and $y$ is not one of the $k$ forbidden strings. Then either $|\text{Rejectors}_{x}| \geq 2^n - (k + 8(|x|^3 + i^3))$ or there exists a set $S$ such that (1) $|S| \leq 2$, (2) $S$ contains no forbidden string, (3) $(\forall y \in S)|y| = |x|$, and (4) $N^{A,x_i,n-1}(q_1) \geq 1$ is noncategorical (in particular, it has more than one accepting path on input $x$).

Proof Sketch for Lemma 3.7

4 Concl

This paper extends the counting argument to the case where the UP has no Turing-Many related

Let $\text{Acceptors}_{x}$ and $\text{Rejectors}_{x}$, and let $\text{Acceptors}_{x}$ and $\text{Rejectors}_{x}$ be the path (otherwise. For each pair of nonaccepting $\text{Acceptors}_{x}$ and $\text{Rejectors}_{x}$, there is at least one string added from $\text{Acceptors}_{x}$ and $\text{Rejectors}_{x}$).

So the number of $\text{Paths}_{x}$ is $\leq ((n^i + 1))^{|PA| + |PA| + (n^i + 1)^2 + (n^i + 1)^3}$.

4 Concl

This paper extends the counting argument to the case where the UP has no Turing-Many related
4 Conclusion

This paper surveyed recent progress in structural complexity theory, and suggested that counting arguments have played a crucial role in these advances. We proved, using counting techniques, that there is a relativised world in which UP has no Turing complete languages.

Many related open problems remain. How much further can counting techniques be pushed in obtaining complexity hierarchy collapses, class containments, and relativised separations? What techniques might be used to separate complexity classes?

This paper has surveyed just a few of the interesting recent uses of counting. A great variety of applications and discussions of counting in complexity can be found in [All85,CH88,He88b,He88b,He88a,Sch88a,Sim77,Wag88].

References


[HH86] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. In...


[Tar87] G. Tardos. Query complexity, or why is it difficult to separate NP^A \cap \text{coNP}^A \text{ from } P^A \text{ by random oracles}. A. July 1987. Manuscript.


