POLYNOMIAL ISOMORPHISMS AND NEAR-TESTABLE SETS

by

Judy Goldsmith

Computer Sciences Technical Report #816

January 1989

POLYNOMIAL ISOMORPHISMS AND NEAR-TESTABLE SETS

Judy Goldsmith

A thesis submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

(Mathematics)

at the

UNIVERSITY OF WISCONSIN–MADISON

1988

ABSTRACT

POLYNOMIAL ISOMORPHISMS AND NEAR-TESTABLE SETS

Judy Goldsmith

Under the supervision of Associate Professor

Deborah Joseph

This thesis investigates two areas in structural complexity, as listed below.

- *The Berman-Hartmanis Conjecture*

The Berman-Hartmanis conjecture states that all $\leq_m^P$-complete sets for NP are polynomially isomorphic. We construct an oracle A such that all $\leq_m^P$-complete sets for NP^A are p^A-isomorphic, and an oracle B such that there are $\leq_{1tt}^P$-complete sets for NP^B that are not p^B-isomorphic.

- *Near-Testable Sets*

Near-testability is a form of self-reducibility based on the lexicographic order. In terms of complexity, the near-testable sets lie somewhere between P and $E \cap PSPACE$. We show that the existence of one-way functions implies that there are near-testable sets that are not in P. In [GJY-87], (2 for 1) p-cheatable sets were suggested as a possible generalization of near-testable sets. Here, we demonstrate that P can be characterized as the intersection of the near-testable sets with the (2 for 1) p-cheatable sets.

I wish to thank my advisor, Debby Joseph, and my coauthors, Paul Young and Lane Hemachandra for their encouragement and support. Without that, this thesis work would not have been completed. And I want to thank all the people who made Madison a wonderful place for me, without whom I might have tried to graduate a long time ago. In particular, the Oak Apple Morris Team, especially Greg Winz; the women of the Women's *Minyan*, and the Rape Crisis Center; all those groups are made up of wonderful people, and have provided space for me to be more than my theorems. My friends, Evelyn Hart, Carolyn Briggs, Arden Rice, Wendy McCanless, Jamie Cowles, and many others have provided the hugs that are only implicit in this thesis. Many people around the CS Department have listened, advised, hugged, and even sometimes proofread. Special thanks are due to Darrah Chavey (who hacked the only figure in this thesis), Anne Condon, Ken Kunen, Bart Miller, Victor Shoup, Meera Sitharam, Igor Steinberg, and Will Winsborough.

There have been many other wonderful people and groups here in Madison. When I arrived here, the members of the math department welcomed me immediately as a colleague and a friend. I hope I will always work in such a comfortable department.

My friends in other places have managed to stay in touch, through email, phone, letters, visits, and telepathy. I am very glad they have. Thank you, Ann Jacobs, Lori Bechtold, Rob Gross.... My family gave up, years ago, asking when I was going to finish up and come home. Well, one out of two. I am grateful to them for the support they have given me over the years. Finally, I owe many thanks to Mordecai Mac Low, who has consistently reminded me of the rewards of ambition, kept me going through the worst of it, and been part of some of the best of it. May there be more!

Finally, there is the future. Many people in the theoretical computer science community have supported and encouraged me. I joined this community because of the research, and because it was a community I wanted to be part of. I feel welcomed by both the computer scientists and the mathematicians at Dartmouth. I look forward to it all.

CONTENTS

CHAPTER 1. INTRODUCTION

In this thesis, we consider two separate areas. The first is the open question of whether all sets that are $\leq_m^P$-complete for NP are polynomially isomorphic. Although we cannot answer this question, we do provide evidence for both sides, and a discussion of various common techniques for this area, along with their limitations. The 'evidence' that we present is in the form of oracles for which all many-one, polynomially complete sets for NP are (or are not) polynomially isomorphic.

The second area we consider is the classification of *near-testable* sets. Near-testability is a self-reducibility property, based on the lexicographic ordering of a set. We consider the effect this structure has on the time and space complexity of a near-testable set. While all sets in P are near-testable, it is not clear whether all near-testable sets are in P. We show that the existence of one-way functions implies that there are near-testable sets that are not in P. This theorem leads to the $\leq_{1,inv}^P$-equivalence of the near-testable sets with the class Parity-P, as presented in [GHJY-87].

The investigation of near-testable sets was motivated by a research program of finding an underlying theory of the effect of internal structure on the complexity of a set. We show that, in certain cases, information about the internal structure gives us full information about the complexity of a set, although in other cases it does not. Our final section will show that P can be characterized by a combination of self-reducible properties. However, we cannot necessarily replace "near-testable" by "Parity-P" in that theorem, indicating that, in spite of the $\leq_{1,li}^P$-equivalence, the two classes have significant differences. We also show that out characterizations cannot be extended, giving some combinations of self-reducible properties that do not characterize P.

CHAPTER 2. THE BERMAN-HARTMANIS CONJECTURE

2.1 Introduction

The field of structural complexity has grown out of the interplay between computational complexity and recursive function theory. Those who study the classes of deterministic and nondeterministic polynomial time definable sets (P and NP), while motivated by real computational questions, have often tried to establish analogies of theorems about recursive and recursively enumerable sets (REC and RE, respectively). For instance, the polynomial time hierarchy, introduced by Meyer and Stockmeyer [MS-72], is modeled on the arithmetic hierarchy. Just as recursion theorists study sets that are complete for RE and other levels of the arithmetic hierarchy, structuralists study sets that are complete for NP and other classes in the polynomial hierarchy.[1]

In [Myh-55], Myhill showed that all of the sets that are $\leq_m$-complete for RE are $\leq_1$-complete for RE, and in fact are all recursively isomorphic. He further characterized the $\leq_m$-complete sets for RE as being the *creative* sets. (A set A is *creative* if it is r.e. and $\overline{A}$ is *productive*, i.e., there is a total recursive function ψ such that, if the i^{th} r.e. set, W_i, is contained in $\overline{A}$, then $\psi(i) \in \overline{A} - W_i$.)

In [Ber-77], Leonard Berman proved a strong analogue of Myhill's Theorem, namely that all $\leq_m^P$-complete sets for E are $\leq_{1,li}^P$-interreducible. This was also proved by Dowd [Dow-78]. The same year, Berman and Hartmanis [BH-77] considered even stronger analogues of Myhill's work. Myhill showed that, because the $\leq_m$-complete sets for RE are creative, they have *padding functions*. That is, for each $\leq_m^P$-complete set C, there is a

[1] The following survey of research on the Berman-Hartmanis conjecture owes a debt to the excellent surveys of the area included in Young's retrospective paper [You-88], and in [KMR-87].

recursive function $p(x,n)$ such that for any x and n, $p(x,n) \in C$ if and only if $x \in C$. This is equivalent to showing that each creative set C is a *cylinder*, that is, $C \cong C \times \mathbf{N}$. The padding functions are used to construct isomorphisms between complete sets; any set that is recursively isomorphic to a cylinder also has padding functions. Berman and Hartmanis showed that SAT and all of the familiar $\leq_m^P$-complete sets for NP have polynomial time padding functions. Furthermore, a set C that is $\leq_m^P$-complete for NP is p-isomorphic to SAT if and only if it has a polynomial time computable padding function. Berman and Hartmanis then conjectured that all $\leq_m^P$-complete sets for NP are p-isomorphic to SAT. This conjecture forms a major open problem in the field today.

We should note here that the Berman-Hartmanis conjecture implies that $P \neq NP$, since $P = NP$ means that all nontrivial sets in P are $\leq_m^P$-complete for NP. However, several researchers have considered the possibility that $P \neq NP$, but the Berman-Hartmanis conjecture is false. In particular, Mahaney and Young [MY-85] showed that, if $P \neq NP$ and not all $\leq_m^P$-complete sets for NP are p-isomorphic, then the collection of p-isomorphism degrees of $\leq_m^P$-complete sets for NP is infinite. Furthermore, if we order these p-isomorphism degrees by $\leq_{1,li}^P$-reductions, then any countable partial order can be embedded into this ordering. In other words, if the Berman-Hartmanis conjecture fails, it fails badly. They also showed that a set A has a polynomial padding function if and only if it is a p-cylinder, i.e., $A \cong^P A \times \mathbf{N}$. This paper extended earlier work by both authors: [You-66], [You-83], and [Mah-81].

There are two major difficulties in extending Myhill's work to *polynomial* isomorphisms. The first is that, in order for two sets to be p-isomorphic, they must have the same density[2] up to polynomial factors. Berman and Hartmanis conjectured that, (given

[2] We say that a set A has $f(n)$ density if the number of strings in A of length $\leq n$ is $\leq f(n)$. For instance, sparse sets have polynomial density.

$P \neq NP$) there are no *sparse* sets that are $\leq_m^P$-complete for NP. Piotr Berman gave a partial solution to this conjecture, by showing that the existence of a $\leq_m^P$-complete *tally* set for NP implies that $P = NP$. Fortune [F-78] extended Berman's result to show that the existence of a sparse $\leq_m^P$-complete set for $coNP$ implies $P = NP$. Finally, Mahaney [Mah-82] showed that the existence of a sparse, $\leq_m^P$-complete set for NP implies $P = NP$.

The second problem in extending Myhill's work is that we do not necessarily know how to *invert* polynomially computable functions quickly, even if they are length-increasing. Berman's theorem shows that, if there are no one-way functions, then all the $\leq_m^P$-complete sets for E are p-isomorphic. Joseph and Young [JY-85], in a careful study of the degree to which Myhill's isomorphism results might carry over to polynomial reducibilities on NP, conjectured that the existence of one-way functions would imply the failure of the Berman-Hartmanis conjecture; Kurtz, Mahaney, and Royer [KMR-86] further conjectured that the Berman-Hartmanis conjecture fails *if and only if* there are one-way functions.[3] However, recent work by Hartmanis and Hemachandra [HHem-87] gives an oracle A such that $P^A \neq UP^A$, yet there are $\leq_m^{P,A}$-complete sets for NP^A that are not p^A-isomorphic to SAT^A.

Research on the Berman-Hartmanis conjecture has taken three principal directions: work on NP, usually concentrating on oracle constructions, or on the possible density of $\leq_m^P$-complete sets for NP; consideration of other complexity classes, such as Berman's theorem for E; and consideration of the p-isomorphism question for sets complete under

---

[3] Grollman and Selman [GS-85] have shown that the existence of one-way functions is equivalent to the separation result $P \neq UP$ (where UP is the subclass, *unique-P,* of NP of sets whose elements have *unique* accepting computations). We will use this form of the hypothesis when discussing oracle results, since it is easier that way to specify that we are discussing the existence of one-way functions *with respect to the oracle.* Note also that $P \neq UP$ implies $P \neq NP$.

weaker reductions. We use these categories to provide a rough framework for the rest of our survey.

Weaker Reductions

Soon after the publication of [BH-77], Hartmanis published [Har-78], in which he applied the same ideas to the study of *Logspace*-complete (or L-complete) sets. He proved that a set is L-isomorphic to SAT if and only if it has a L-computable and L-invertible padding function; that the known L-complete problems for NL (nondeterministic L), CSL (context sensitive languages), P, NP, and $PSPACE$ are all respectively, L-isomorphic, and that no L-complete set for CSL, $PSPACE$, E, or $ESPACE$ can be sparse. He also conjectured that neither NL nor P has sparse, L-complete sets. Kurtz, Mahaney, and Royer [KMR-87] showed that there is a set, C, that is complete for $PSPACE$ under $\leq^L_{2tt}$-reductions, such that any set that is $\leq^L_m$-equivalent to C is L-isomorphic to it. Hartmanis, Immerman and Mahaney [HIM-78] considered p-isomorphisms of sets that are complete under reductions computed by *Logspace* Turing machines with *one-way* read heads (known as *1-L*-reductions) for different classes. Hartmanis and Mahaney [HM-81] then showed that there are no sparse *1-L*-complete sets for NL, partially answering one of Hartmanis's conjectures. In [A-86], Allender showed that all *1-L*-complete sets for $PSPACE$ are p-isomorphic.

Recently, the most commonly studied complete sets are the $\leq^P_{btt}$-complete sets. Several of the constructions in this chapter use $\leq^P_{btt}$-complete sets instead of $\leq^P_m$-complete sets, because this allows us to sidestep conflicts between coding requirements and diagonalization requirements. We will discuss this at greater length later in the chapter. One of the first examples of this use of $\leq^P_{btt}$-completeness is the set constructed by Ko, Long, and Du in [KLD-86]. Their construction answered questions posed by Watanabe in [Wat-86]. In that paper, Watanabe explored Joseph and Young's conjecture that the existence

of one-way functions implies the failure of the Berman-Hartmanis conjecture. Watanabe asked whether the existence of one-way functions would imply that there are sets that are $\leq^P_{1,li}$-equivalent, yet not p-isomorphic. He also asked whether such sets could be $\leq^P_m$-complete for NP. Ko, Long, and Du answered the first question in the affirmative, and gave a partial answer to the second. Namely, they used a one-way function to describe two sets in E that are $\leq^P_{1,li}$-equivalent, not p-isomorphic, and are $\leq^P_{2tt}$-complete for E.

In [KMR-86], Kurtz, Mahaney, and Royer constructed sets A and B that are $\leq^P_{2tt}$-complete for E, such that any set that is $\leq^P_m$-equivalent to A is p-isomorphic to A (in their idiom, "the m-degree of A collapses"), but not every set that is $\leq^P_m$-equivalent to B is p-isomorphic to B. By considering the $\leq^P_m$-degree of B, rather than the smaller $\leq^P_{1,li}$-degree, they were able to do this construction without assuming the existence of a one-way function. Independently, the author and D. Joseph [GJ-86] constructed a time function $T(n)$, such that there are $\leq^P_m$-equivalent, non-p-isomorphic $\leq^P_{2tt}$-complete sets for $DTIME[T(n)]$. Theorem 2.3.1 and the rest of the constructions in §2.3 are based on the construction in [GJ-86]. These constructions are simpler than Kurtz, Mahaney, and Royer's, and they produce non-p-isomorphic sets that are $\leq^P_{1tt}$-complete for E. The constructions in [KMR-86] explicitly consider p-cylinders. In the collapsing construction, they guarantee that every set that is $\leq^P_m$-equivalent to B is also a p-cylinder; in the non-collapsing construction, they diagonalize away from p-cylinders. Our constructions are simpler because they diagonalize directly against p-isomorphisms on finite intervals, rather than infinite "splinters."

In [KMR-87], Kurtz, Mahaney and Royer announced that they have constructed a sparse oracle A for which all $\leq^{P,A}_{2tt}$-complete sets for NP^A are p^A-isomorphic, and another for which they are not. They also constructed a sparse oracle for which all $\leq^{P,A}_m$-complete sets for NP^A are $\leq^P_{1,li}$-equivalent. They speculated that a sparse oracle that resolved the Berman-Hartmanis conjecture in either direction would be strong evidence that that result

held in the unrelativized world. Long [Lon-88] also constructed a sparse oracle for which $E^A \neq NP^A$, a one-way function f^A exists, and there is a $\leq^{P,A}_{2tt}$-complete set for NP^A such that $C^A \equiv^{P,A}_m f^A(C^A)$, but $C^A \not\cong^{P,A} f^A(C^A)$. In Theorem 2.3.1 we show that there is a sparse oracle A such that there are $\leq^{P,A}_{1tt}$-complete sets for NP^A that are $\leq^P_m$-equivalent, yet not p^A-isomorphic. This work was done independently of the others, and once again is a slight simplification of Kurtz, Mahaney, and Royer's.

Other Classes

We have already mentioned some of the results for other classes, in particular for sets that are $\leq^P_{btt}$-complete for E. The study of p-isomorphisms of E-complete sets goes back to Berman's thesis [Ber-77], where he showed that all $\leq^P_m$-complete sets for E are $\leq^P_{1,li}$-equivalent. This was also shown by Dowd [Dow-78], Watanabe [Wat-86], and most recently by Ganesan and Homer [GH-88]. Ganesan and Homer have simplified the proof, and have extended it to $\leq^P_m$-complete sets for NE. (Watanabe [Wat-83] has also extended this to $DSPACE[L(n)]$, for any super-polynomial function $L(n)$.) In addition to these theorems, other constructions also showed that all $\leq^P_m$-complete sets for E are of similar densities. Berman and Hartmanis exhibited a set $A \in E$ that is not sparse, and such that all $\leq^P_m$-reductions from A are one-one almost everywhere, using a construction they attributed to Meyer. Thus, any $\leq^P_m$-complete set for E contains a polynomial image of A, so cannot be sparse. Balcázar and Schöning [BS-85] call such a set *strongly bi-immune*. Ko and Moore [KM-81] showed that the set given in [BH-77] is p-immune (has no infinite, polynomial time computable subsets). They also showed that a $\leq^P_m$-complete set for E cannot be p-immune, although they constructed a p-immune set that is $\leq^P_{tt}$-complete for E.

Instead of increasing the time complexity of the class from P to E, some researchers have considered Σ^P_2, the existential class above NP in the polynomial hierarchy. In [KMR-

87], Kurtz, Mahaney, and Royer considered $\leq_m^P$–degrees of NP-hard sets in Σ_2^P. More recently, Homer and Selman [HS-88] constructed an oracle A, for which $P^A = UP^A \neq NP^A$, and $\Sigma_2^{P,A} = EXP^A$. Since Berman's theorem relativizes, this describes a world in which all $\leq_m^P$-complete sets for $\Sigma_2^{P,A}$ are $\leq_{1,li}^{P,A}$-equivalent, and there are no one-way functions. In other words, the Berman-Hartmanis conjecture holds for $\Sigma_2^{P,A}$. Unfortunately, Homer and Selman have shown that their technique cannot be extended to show that $NP^A = E^A$.

Relativizing the Berman-Hartmanis Conjecture

In contrast to Homer and Selman's construction, Theorem 2.2.3 gives an oracle A such that all $\leq_m^P$-complete sets for NP^A are p^A-isomorphic. In Theorem 2.2.4, we show that the techniques in this construction are not sufficient to guarantee that all of the $\leq_m^{P,A}$-complete sets for NP^A are p^A-isomorphic. There are three principal techniques in the construction in Theorem 2.2.3: creating a dense subset of SAT^A that is strongly bi-immune, forcing all $\leq_m^P$-complete sets for NP^A to be similarly dense, and coding the p^A-isomorphisms into A, and keeping $P^A \neq UP^A$ by diagonalizing. Kurtz [Kur-86] has extended the first technique, to construct an oracle A for which there is a set in NP^A that is strongly bi-immune with respect to $\leq_m^{P,A}$-reductions, forcing all $\leq_m^{P,A}$-complete sets for NP^A to be similarly dense. However, this construction interferes with all P^A-recoverable encodings, so it cannot be combined with the coding techniques of Theorem 2.2.3.

In [JY-85], Joseph and Young defined the *k-creative* sets, and showed that every k-creative set is $\leq_m^P$-complete for NP. Joseph and Young's study of the k-creative sets led to their conjecture that the existence of one-way functions implies the failure of the Berman-Hartmanis conjecture. Joseph and Young's work inspired Homer [Hom-86]'s discussion of *natural* creative sets, in the context of natural Gödel numberings. Others have also considered the structure of creative sets, including Watanabe [Wat-83&86a]. The one-way function conjecture has also stimulated research; we have already mentioned several

constructions that were done in response to this conjecture.

One oracle construction we mentioned in relation to the Joseph-Young conjecture was Hartmanis and Hemachandra's oracle A for which $P^A = UP^A \neq NP^A$, but there is a set C that is $\leq_m^{P,A}$-complete for NP^P, and *not* p^A-isomorphic to SAT^A [HHem-87]. The technique used to construct C is one that goes back to the earliest oracle result that explicitly addressed the Berman-Hartmanis conjecture, by Kurtz [Kur-83]. To guarantee that $C \not\cong^{P,A} SAT^A$, Hartmanis and Hemachandra, like Kurtz, code super-polynomial length intervals of SAT^A into the oracle, so that C consists of the remaining, uncoded intervals. Thus, C cannot be mapped to SAT^A by any $\leq_m^{P,A}$-reduction. Other oracle results have concentrated on the fine structure of the complete sets and the oracle, rather than relying on simple density arguments.

The most recent oracle result is that of Kurtz, Mahaney, and Royer [KMR-88], showing that the Berman-Hartmanis conjecture fails relative to a random oracle. The theorem proved is that, relative to a random oracle, A, there is a so-called "scrambling" function, a very strong form of a one-way function. This function, f^A, has the property that there is no polynomially computable, one-one reduction from $f^A(SAT^A)$ to SAT^A. Therefore, the $\leq_m^{P,A}$ complete set $f^A(SAT^A)$ is not p^A-isomorphic to SAT^A. This is the first instance of a relativization for a full version of either side of the Berman-Hartmanis conjecture.

The isomorphism problem for NP is one of the few major problems in structural complexity for which there are not relativizations for both sides of the question. This fact, plus recent work, including [KMR-88], leads us to believe that Berman and Hartmanis' conjecture is false.

2.2 The Construction of an Oracle A for which All $\leq_m^P$-Complete Sets for NP^A are P-Isomorphic

A necessary condition for two sets to be p-isomorphic is that they have the same densities, up to a polynomial factor. Since we will be comparing sets to SAT^A, we need language to talk about the density of SAT^A. We will show that the following condition describes the density of SAT^A for any A.

Definition 2.2.1. *A set $A \subseteq \{0,1\}^*$ is* thick *if there is a polynomial, $p(n)$, and a one-one reduction, $r(x)$, from $\{0,1\}^*$ to A such that $|r(x)| \leq p(|x|)$.*

In the definition of thickness, we do not require that the reduction r be polynomially *computable*; we do require that it have a polynomial bound on the size of the gaps (i.e., the *number* of contiguous intervals) in its codomain.

There are two approaches that one might take to constructing an oracle for which all NP-complete sets are p-isomorphic: either one might diagonalize away from those sets that are not p-isomorphic to SAT^A, to somehow make those sets incomplete, or one might simply to encode p-isomorphisms between SAT^A and all $\leq_m^P$-complete sets for NP^A into the oracle. We use a combination of these two approaches. We use a diagonalization argument to ensure that sets that do not have the same density as SAT^A, up to a polynomial factor, are not complete. Then, to make a complete set, T^A, p-isomorphic to SAT^A, we code the p-isomorphism $g : T^A \leftrightarrow SAT^A$ into the oracle. Given A, for each x and each g we can compute $g(x)$ with only $2 * |g(x)|$ polynomial size (in $|x|$) queries to A, and similarly, for g^{-1}. Before showing how to construct the oracle A, we need to discuss the density of SAT^A and the way we code relativized Boolean formulas over the alphabet $\{0,1\}^*$.

Lemma 2.2.2. *There is a polynomial encoding of SAT^A into $\{0,1\}^*$ such that SAT^A is both thick and co-thick.*

Proof. When we consider SAT as the set of satisfiable Boolean formulas in conjunctive normal form over the finite alphabet Σ of logical symbols and variables, it is easy to see that the subset of Σ^* consisting of tautologies is thick.[4] Since $SAT \subseteq SAT^A$, SAT^A is also thick over Σ. However, for the purposes of our proof, it is useful to think of SAT^A as a subset of $\{0,1\}^*$ rather than Σ^*; that is, we code relativized Boolean formulas over the standard alphabet Σ into $\{0,1\}^*$. Doing this allows us to identify certain subsets of SAT^A quickly. It also allows us to identify particular relativized Boolean formulas with particular elements of A, also strings over $\{0,1\}^*$.

We could define a mapping from Σ^* to $\{0,1\}^*$ simply by using the change of base formula. This would give us an encoding of SAT^A in $\{0,1\}^*$ of the same density, modulo a constant, as the original. Furthermore, it would furnish us with a deterministic linear-time reduction from SAT^A over Σ^* to SAT^A over $\{0,1\}^*$. However, later details of the proof are cleaner if we use a simple variant of this coding.

We wish to be able to recognize certain types of relativized formulas. Therefore, we code all purely Boolean formulas (those that do not refer to the oracle), as strings beginning with "1." We code all formulas that make reference to the oracle as strings beginning with a "0." Those that consist of a single oracle query will begin with "00," and more complex formulas that involve oracle queries will begin "01." With this coding technique, half the strings in $I(n)$ represent Boolean formulas without oracle references. Of these, a thick subset are satisfiable. Thus, $SAT \subseteq SAT^A$ implies that SAT^A is thick with respect to

---

[4] We define the following function f from the set of Boolean formulas into the set $TAUT$ of tautologies. Let b be a Boolean formula in conjunctive normal form. Then $f(b) = b \vee \neg b$. If b is the conjunction of k clauses of length $\leq s$, then $f(b)$ will have k^2 clauses of length $\leq 2s$, so f is polynomially bounded. If we interpret each element of $\{0,1\}^*$ as a unique Boolean formula, this shows that $TAUT$ is thick. A similar argument shows that $coTAUT \subseteq coSAT$ is also thick. Note that any set that is p^A-isomorphic to SAT^A is also thick.

this coding scheme, for any oracle A. A similar argument shows that $coSAT^A$ is also thick for any A. ∎

We are now ready to prove our first major result.

Theorem 2.2.3. *There is an oracle A such that $P^A \neq UP^A$, and all sets that are $\leq_m^P$-complete for NP^A are p^A-isomorphic. Furthermore, the oracle A is in E.*

Proof. There are three sets of requirements in this construction. In order that $P^A \neq UP^A$, we define the set

$$L^A = \{0^n : \exists z \in A,\ |z| = n \text{ and } z = 01w\},$$

and we require that for each $P_s^A \in P^A$, $P_s^A \neq L^A$. We construct A so that $L^A \in UP^A$. In order that all of the $\leq_m^P$-complete sets for NP^A are p^A-isomorphic, we require that they are all *thick*. Finally, we code p^A-isomorphisms between these sets and SAT^A into the oracle A, so that each isomorphism can be computed in polynomial time, using the oracle as a table. We refer to these three types of requirements as *diagonalization, thickening,* and *coding.*

The construction will proceed in stages. At stage s, we decide A on all strings from length n_s up to n_{s+1}. At stage s, we will consider the sets $\{T_0^A, ...T_s^A\}$ recognized by the first $s+1$ nondeterministic polynomial time oracle Turing machines in our enumeration, and the first $s+1$ polynomially computable functions, $\{f_0, ...f_s\}$. If some f_j is a reduction from SAT^A to T_i^A (where defined on strings of length $\leq n_s$), we say that T_i^A is *active,* and that f_j is active for T_i^A.

In the construction, we begin stage s by meeting one diagonalization requirement ($P_s^A(0^n) \neq L^A(0^n)$ for some n). However, the diagonalization depends on the details of the other parts of the construction, so we begin by describing those.

Thickening

At stage s, suppose T^A is active, so some f_j ($j \leq s$) is a reduction from SAT^A to T^A where it is defined. If f_j were one-one, we would be done, since the one-one image of a thick, co-thick set is itself thick and co-thick. We can force f_j to be one-one on a thick subset of formulas, namely the formulas of the form "$x_0 \in A$," for strings x_0 beginning with 00. Notice that such formulas are easily recognizable, and that exactly one quarter of the formulas of any length are of that form. (In fact, the codes for such formulas also begin 00. This is simple coincidence.) This part of the construction relies on the fact that the reductions f_j *do not consult the oracle.*

For each active T^A, in turn, and each active reduction f from SAT^A to T^A, we guarantee that f is one-one on these formulas, or that f is not a reduction from SAT^A. If there are formulas b_1 = "$x_0 \in A$", and b_2 = "$y_0 \in A$" for some x_0 of length between n_s and n_{s+1}, and y_0 of length less than n_{s+1}, both beginning with 00, and furthermore, $f(b_1) = f(b_2)$, then we spoil f as a reduction from SAT^A, by deciding $A(x_0) = SAT^A(b_1)$ so that $SAT^A(b_1) \neq SAT^A(b_2) = A(y_0)$. Notice that $f(b_1)$ and $f(b_2)$ do not change when we change A. That is necessary for this part of the construction.

At stage s, we may decide $A(x_0)$ for up to $(s+1)^2$ strings beginning 00. Once this is done, we can assume that each active f is one-one on all but $(s+1)^2$ formulas of lengths n_s up to n_{s+1} beginning with 00. Of the remaining strings of these lengths beginning with 00, we include half in A, and restrain half (for each length). Thus, each active set is thick. Notice that any T^A that is $\leq_m^P$-complete for NP^A will have some reduction f that is consistent after some stage s', so T^A will remain thick. Furthermore, the polynomial bound on $|f(x)|$ will give us an approximate bound on the density of both T^A and $co-T^A$.

Coding

The p^A-isomorphisms we define are influenced periodically by the diagonalization part of each stage. However, we use lexical mappings between the active sets and SAT^A as

the default computations. As we observed above, each T^A that is $\leq_m^P$-complete for NP^A is thick and co-thick, so the lexical isomorphism between T^A and SAT^A is polynomially bounded.[5] Suppose T^A is active, and there is no ongoing coding of an isomorphism between T^A and SAT^A. Then we begin by assigning a *key* to the isomorphism, that specifies the isomorphism. We assign a key to each new encoding of an isomorphism at stage s, so that each q assigned is the least $q > s$ such that q has not already been assigned.

Next, we assign a padding function $p(n)$. We choose p so that $p(n)$ is at least the polynomial bound on the run-time of SAT^A, the polynomial bound on the run-time of T^A, and the bound on the lexical isomorphism that we can derive from the bound of the least-numbered active f that is a reduction from SAT^A to T^A.

Each coding string that we insert into the oracle begins with a 1, so as not to interfere with the thickening (or diagonalizing) strings. Each string answers a question of the form, "is the m^{th} bit of $h(x)$ a 1," or, "is there a $m + 1^{st}$ bit of $h(x)$?" Thus, each string is a tuple that specifies the isomorphism key, the direction (to or from SAT^A), x, the type of question, and m. The implementation of this is not important, as long as it is consistent, and as long as each coding string begins with a 1.

Notice that we may drop and add sets from the active list, and we may discover that the padding function we have assigned is not sufficient, if we spoil the corresponding reduction and find a new one. In such cases, we say that the coding is *injured,* and we begin coding the isomorphism from the beginning, with a new key and padding function.[6]

---

[5] The lexical mapping is the simplest isomorphism between two sets, sending the n^{th} element of the first set to the n^{th} element of the second set, and the n^{th} element of the complement of the first set to the n^{th} element of the complement of the second set.

[6] This technique is similar to the one used by Kurtz [K-85] to encode sparse sets into an oracle A, such that $P^A \neq NP^A$. In that construction, the codings are iteratively defined, based on approximations to the census function for the given set. The basis of each iteration is a polynomial large enough that the computation cannot query its

Diagonalizing

At stage s, we guarantee that $P_s^A(0^n) \neq L^A(0^n)$, where P_s^A is the s^{th} P^A machine. Remember that $L^A = \{x : \exists y \ |x| = |y| \text{ and } 01xy \in A\}$. L^A is in UP^A because we include at most one string of the form $01xy$ in A, for each x. We assume, without loss of generality, that $P_s^A(0^n)$ queries fewer than 2^{n-5} strings. Suppose $P_s^A(0^n)$ queries a string w of length $> n$. If w is of the form "$0string$," then we restrain w from A. This may mean that we restrain up to 2^{n-5} thickening strings of some length. However, that still leaves enough strings to adequately thicken SAT^A and coSAT^A. Furthermore, we guarantee that each length is affected by at most one diagonalization. However, if w is of the form "$1string$," and codes part of $g(x)$, for some p^A-isomorphism $g : T^A \leftrightarrow SAT^A$, we need to be more careful.

Case 1. Part of $g(x)$ or $g(y) = x$ has already been encoded. We fix $A(w)$ to be consistent with this encoding.

Case 2. $g : T^A \rightarrow SAT^A$, and none of $g(x)$ has been encoded yet. We fix $g(x)$ to be a relativized Boolean formula $b(\langle x, k\rangle, 0, r)$ specifying the condition that there is some string of length r in A of the form $001\langle x, k\rangle 0y$, for k the key to the isomorphism g, and some $r > n + |k| + |x|$. (Notice that such a string is in the thickening region of the oracle.) Formally,

$$b(\langle x, k\rangle, 0, r) = \neg x_1 \ \& \ \neg x_2 \ \& \ x_3 \ \& \ (\text{the next } |\langle x, k\rangle| \text{ bits represent } \langle x, k\rangle)$$

$$\neg x_{\langle x,k\rangle+1} \ \& \ [x_1 x_2 \cdots x_r \in A].$$

---

own encoding. In Kurtz's proof, if a set is indeed sparse, it eventually becomes fully encoded. Here, if the set T^A is truly $\leq_m^P$-complete, we eventually 1) find a reduction $f : SAT^A \rightarrow T^A$, and 2) a padding function that bounds the run-times of both T^A and g.

For example,

$$b(1101,12) = \neg x_1 \ \& \ \neg x_2 \ \& \ x_3 \overbrace{\ \& \ x_4 \ \& \ x_5 \ \& \ \neg x_6 \ \& \ x_7}^{1101}$$

$$\neg x_8 \ \& \ [x_1 x_2 \cdots x_{12} \in A].$$

We choose r greater than $n + |k| + |x|$, so that $P_s^A(0^n)$ cannot query every string of the form $001xky$ of length r; we also require that $r > p_T(|x|)$, where p_T is a strictly increasing polynomial bound on the run-time of $T_s^{()}$, so that $T_s^A(x)$ cannot query strings of length r. We add the 0 on the end so that subsequent b's in this diagonalization cannot restrain all of the oracle strings that this b discusses–this query generates a unique $\langle x, k\rangle$, and the 0 guarantees that only half the possible extensions can be restrained by another b. We choose the least such r such that none of $g(b(\langle x,k\rangle, 0, r))$ or $g^{-1}(b(\langle x,k\rangle, 0, r))$ has been coded yet.

Since T^A appears to be complete at stage s, there is some $f : SAT^A \to T^A$ in $\{f_0, ..., f_s\}$. Let f be such a reduction with the least number. Then the polynomial bound on $|f(x)|$ gives us an estimate of the density of T^A. Thus, we can guess a polynomial bound, p_g, on the lexical isomorphism $g_l : SAT^A \leftrightarrow T^A$, and on g_l^{-1}. Then for any z with $|z| > p_g(n)$, $|g_l(z)|$ and $|g_l^{-1}(z)|$ are greater than n, so none of $g(z)$ or $g^{-1}(z)$ has been coded yet. Thus, we can always find an $r > p_g(n)$. In fact, we can choose $r \leq p_g(n) + p_T(|x|) + |k| + 2$. Notice that n is polynomially bounded in $|x|$, since $g(x)$ has not yet been coded. This bound is also determined by g_l. Since $|b(\langle x,k\rangle, 0, r)|$ is polynomially bounded in r (remember, $|x| < r$), g remains polynomially bounded, with a new bound depending on the run-time of T^A and the encoding of $g : SAT^A \leftrightarrow T^A$. Notice that, as long as f remains a reduction from SAT^A to T^A, this bound remains fixed.

Case 3. $g : SAT^A \to T^A$, and none of $g(x)$ or $g^{-1}(y) = x$ has been encoded yet. Let f be as before. This time, we consider $b(xy, r)$, for some y with $|y| = p_g(n) + 1$.

Let $p_{SAT}(|z|)$ be a strictly increasing polynomial bound on the run-time of $SAT^{()}(z)$, and $p_f(|z|)$ a strictly increasing polynomial bound on $|f(z)|$. If there is an r such that

$$r \leq |x| + p_g(n) + n + p_{SAT}(|x|) + 2,$$

and a $b(xy, r)$ such that none of $g(f(b(xy, r))$ or $g^{-1}(f(b(xy, r)))$ has been coded, then we choose the least such

$$r > |x| + p_g(n) + n + p_{SAT}(|x|),$$

and the least corresponding y.

If no such $b(xy, r)$ exists, then f maps all such formulas to strings w such that some of $g(w)$ or $g^{-1}(w)$ has been coded. Since there are $< 2^{p_g(n)}$ such w's, and $> 2^{p_g(n)}$ formulas, f is not one-one on these formulas, so we spoil f. If there is another active reduction $f' : SAT^A \rightarrow T^A$, we replace f by the f' of lowest number and proceed as above. If not, we discontinue coding g, restrain the string that $P_s^A(0^n)$ just queried from the oracle, and continue computing $P_s^A(0^n)$.

If f is one-one on the $b(xy, r)$'s, then there is a $b(xy, r)$ that we can use. Thus, r, and consequently g, is still polynomially bounded in $|x|$.

Notice that spoiling f requires that we fix a certain number of thickening strings. However, the only thickening strings of length r affected by the diagonalization are those beginning 001. This leaves all thickening strings of length r beginning 000 for thickening.

Notice that, in *Case 2* or *Case 2*, we may later discover that f is not a reduction from SAT^A to T^A. However, if T^A is $\leq_m^P$-complete for NP^A, then there is *some* reduction $f : T^A \rightarrow SAT^A$, and some stage t, such that after stage t, f is the active reduction from SAT^A to T^A with the lowest number. If we later discover that the function we considered is not a reduction, we say the encoding of g is *injured*, and we begin again, with a new key and padding function. Since this can only happen up to stage t, we know that, eventually, some $g : T^A \leftrightarrow SAT^A$ will be fully encoded in A.

Finally, we claim that this diagonalization does not ruin any p^A-isomorphism g (except for the finite injuries mentioned above). To show this, we must first describe how to compute $g(y)$ for those values of y not directly affected by the diagonalization. We note that $P_s^A(0^n)$ queries fewer than 2^{n-5} strings, so we can simply not count those strings directly affected by the diagonalization when we compute the lexical ordering of the sets.[7] Because so few strings are affected, and both sets are thick, this will add no more than a constant, determined by T^A, to the length of each new $g(x)$. Thus, g remains a p^A-isomorphism between T^A and SAT^A.[8] We guarantee that diagonalizations never overlap, so each $g(x)$ is affected by at most one diagonalization. Therefore, g remains polynomially bounded.

Finally, we add one string of length n, of the form $01string$, to A if $P_s^A(0^n) = 0$.

The Construction

Stage 0. $A = \emptyset$.

Stage s. A is decided up to strings of length n_s. We begin with a diagonalization, with $n = n_s$. We then set n_{s+1} to two more than the length of the longest string affected by the diagonalization.

Next we perform the thickening, and then the coding, one length at a time, for all lengths up to n_{s+1}. We note that the diagonalization affects at most 3/4 of the thickening strings of any given length: $P_s^A(0^n)$ can query no more than 1/4 of them directly, and the

---

[7] This process affects particular isomorphisms, rather than the sets themselves. Although we do not count x or $b(\langle x,k\rangle,0,r)$ (in *Case 2*) when we compute g, we still count $b(\langle x,k\rangle,0,r)$ when we compute other isomorphisms from SAT^A.

[8] Although this section of the proof does not follow their proof exactly, step 6 was inspired by the proof of Theorem 4 in [LY-88].

b's can restrain only those strings beginning 001, i.e., no more than half of the thickening strings of a given length.

Thus, we construct an oracle A for which $P^A \neq UP^A$ and all $\leq_m^P$-complete sets for NP^A are p^A-isomorphic. ■

Theorem 2.2.3 raises several interesting questions. The first is whether we have constructed an oracle counter-example to Joseph and Young's conjecture. Joseph and Young [JY-85] asked whether, if f is a one-way function, $f(SAT) \cong^P SAT$. Notice that, if f^A is a one-way function, $f^A(SAT^A)$ is $\leq_m^{P,A}$-complete for NP^A. However, we have no guarantee that $f^A(SAT^A)$ is $\leq_m^P$-complete for NP^A. In other words, it is possible that $f^A(SAT^A)$ is not p^A-isomorphic to SAT^A, even though all of the $\leq_m^P$-complete sets for NP^A are p^A-isomorphic.

The next question is whether we have considered the right class of complete sets for NP^A. In Theorem 2.2.3, we "collapsed" the $\leq_m^P$-complete sets for NP^A. Clearly, this is not the only definition of many-one complete sets for NP^A. For instance, we could have considered the $\leq_m^{P,A}$-complete sets for NP^A, since that is clearly a stronger definition. It could be argued, however, that the definition we used is the *natural* definition of many-one completeness in an oracle setting, since it is the same definition as in the unrelativized case. The fact that SAT^A is $\leq_m^P$-complete for NP^A lends strength to this argument. Although we often consider *Logspace* reductions on P, or $PSPACE$, for instance, it *could* also be argued that it is unnatural to consider reductions that are less powerful than the Turing Machines we are also discussing.

We next consider p^A-isomorphisms of $\leq_m^{P,A}$-complete sets for NP^A. This class of complete sets *properly* contains the class of $\leq_m^P$-complete sets for many A's. In Theorem 2.2.4, we modify the construction of Theorem 2.2.3 to show that the two classes of NP^A sets can be distinguished with respect to collapsing, i.e., to p^A-isomorphisms.

Theorem 2.2.4. *There is a modification of oracle A such that all $\leq_m^P$-complete sets for NP^A are p^A-isomorphic but* not *all $\leq_m^{P,A}$-complete sets are p^A-isomorphic.*

Proof. One way of ensuring that there are non-p^A-isomorphic complete sets is to have a complete set that is not thick. The construction of Theorem 2.2.3 ensures that all $\leq_m^P$-complete sets for NP^A are p^A-isomorphic. At the same time, we can construct a set G that is $\leq_m^{P,A}$-complete, and that has exponentially large *gaps,* or empty intervals, by adding some extra coding to the oracle. These gaps guarantee that there is no polynomially bounded reduction from G to SAT^A.[9]

G is simple to describe. If $h(0) = 2$, and $h(n+1) = 2^{h(n)}$, then for strings of lengths $h(2n)$ to $h(2n+1)$, G is identical to SAT^A, and empty otherwise. In order that G contain *no* strings of lengths between $h(2n+1)$ and $h(2n+2)$, we code those portions of SAT^A into A. We use the same coding ideas that we used in the construction of Theorem 2.2.3. We reserve the number 2 as the key for this coding; we append $1^{p(n)}$ to each encoding of a string in $I(n)$, where $p(n)$ is a monotonically increasing polynomial bound on the run-time of $SAT^{()}$. The encoding, $e(x)$, is of the form $1^{p(|x|)}\langle 1^2, 0, x, 1\rangle$ whenever $h(2n) \leq |x| < h(2n+1)$ for some n. Since $|e(x)| > p(|x|)$, $SAT^A(x)$ can not consult its own encoding.

We use a similar coding scheme for the intervals of SAT^A and for the isomorphisms because we do not want the different codings to interfere with each other. This way, we know that the same string is never called on to encode both a satisfiable formula, and (part of) an isomorphism.

Finally, we show that the set

$$G = \{x : \exists n\ h(2n) \leq |x| < h(2n+1) \text{ and } SAT^A(x) = 1\}$$

[9] This technique has also been used in [Kur-83 & HHem-87].

is $\leq_m^{P,A}$-complete for NP^A. If $S \in NP^A$ and $f : S \to SAT^A$, we describe the reduction $g^A : S \to G$ as follows:

$$g^A(x) = \begin{cases} f(x) & \text{if } \exists n\ h(2n) \leq |f(x)| < h(2n+1) \\ T & \text{if } \exists n\ h(2n+1) \leq |f(x)| < h((2n+2) \\ & \quad \text{and } e(f(x)) \in A \\ F & \text{otherwise} \end{cases}$$

where T and F are the strings $x_0 \vee \neg x_0$ and $x_0 \wedge \neg x_0$ respectively. (We assume that T is in G.) We observe that

$$x \in S \leftrightarrow f(x) \in SAT^A \leftrightarrow g^A(x) \in G.$$

We have to modify the construction from Theorem 2.2.3 so that the diagonalizations (step 6) only occur in the intervals where SAT^A is not coded into the oracle. Before, the diagonalizations controlled the number of intervals decided at each stage. Now, we extend the construction so that at stage s, we determine all of A from $I(h(2s))$ to $I(h(2s+1))$. Thus, when we decide $L^A(0^n)$ for some n at stage s, all of the strings affected by this diagonalization will have lengths less than $h(2s+1)$. Thus, the encoding of SAT^A will have no impact on the process of diagonalizing. ∎

If we look carefully at the construction in Theorem 2.2.3, we see that there are several parts: "thickening" $\leq_m^P$-complete sets, coding isomorphisms, and keeping $P^A \neq UP^A$. For each potential reduction, f, from SAT^A, either f is one-one almost everywhere on the set M, so we can use it to thicken the corresponding complete set, or we spoil f. The thickening works because, if $T^{A_s}(f(x)) = T^{A_s}(f(y))$, then $T^A(f(x)) = T^A(f(y))$ for any A extending A_s. However, this does not necessarily hold if $f(x)$ depends on the oracle. This property, which we would like to extend, was studied by Berman and Hartmanis [BH-77], and formalized by Balcázar and Schöning [BS-85].

Definition 2.2.5. *([BS-85]) A set B is* strongly bi-immune *if every polynomial reduction from B is one-one almost everywhere. In other words, if f is polynomially computable, and if $f(x) = f(y)$ implies that x is in B if and only if y is in B, then f is one-one on all but a finite number of strings.*

The proof of Theorem 2.2.3 shows that the purely oracle-dependent formulas form a subset of SAT^A that is thick, co-thick and strongly bi-immune with respect to $\leq_m^P$-reductions. Thus, all of the $\leq_m^P$-complete sets for NP^A have the same density, up to polynomial factors. Since this subset of SAT^A is in P^A, it cannot be bi-immune with respect to $\leq_m^{P,A}$-reductions. Unfortunately, we cannot simply alter the construction to allow the reductions we consider to query the oracle, since this would interfere with the coding.[10] However, this aspect of the proof can be isolated, and extended to reductions that are allowed to query the oracle. Kurtz [Kur-86] constructed an oracle C and set $B \in NP^C$ such that B is strongly bi-immune with respect to $\leq_m^{P,C}$-reductions.

It is not clear whether the Berman-Hartmanis conjecture holds with respect to Kurtz's oracle. It is also not clear how to combine this construction with any of the known techniques for deciding the conjecture. Unlike Theorem 2.2.3, there are no intervals on which we can temporarily ignore diagonalization requirements. While we cannot combine Kurtz' technique with coding isomorphisms into C, there are other techniques for guaranteeing that all $\leq_m^{P,C}$-complete sets for NP^C are p^C-isomorphic. Thus, his construction may prove useful, though not in direct combination with Theorem 2.2.3.

---

[10] The problem of simultaneously coding information and satisfying diagonalization requirements is one that has been a stumbling block for extending many results in this area, for instance, [KLD-86, KMR86&87]. In fact, many of the results about complete sets for NP and E use bounded truth-table reductions rather than many-one reductions precisely because of this problem.

Can strongly bi-immune sets help us in an unrelativized setting? [BH-77] showed that there is a strongly bi-immune set in E that is not sparse. However, [Ber-77, Dow-78, W-86, & GH-88] also proved that all $\leq_m^P$-complete sets for E are one-one interreducible. This implies that they are all of the same density, up to polynomial factors, reducing our interest in strongly bi-immune sets for this class. [illegible] bi-immune set cannot be a p-cylinder.[11] [BH-77] observed that any set that is p-isomorphic to SAT is itself a p-cylinder. Thus, we see that the existence of a strongly bi-immune, $\leq_m^P$-complete set for NP would actually contradict the Berman-Hartmanis conjecture.

2.3 The Construction of a Complexity Class Containing Non-Isomorphic $\leq^{\mathbf{P}}_{\mathbf{btt}}$-Complete Sets

In this section we prove a collection of results that show that there are settings in which not all complete sets are $p^{()}$-isomorphic.

Theorem 2.3.1. *There is a sparse oracle C, and sets A and B such that*

i) $P^C \neq NP^C$;

ii) A *and* B *are* $\leq^P_{1tt}$*-complete for* NP^C;

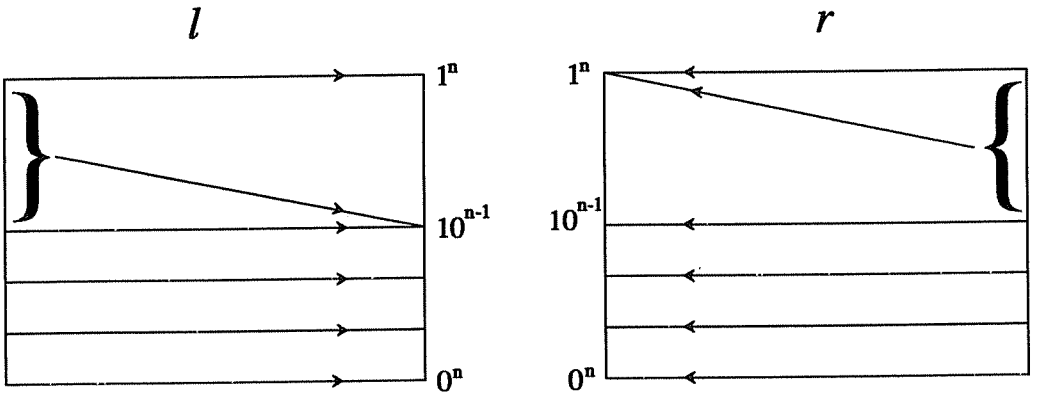
iii) A *and* B *are both thick and co-thick; and*

iv) $A \equiv^P_m B$, *but* $A \not\equiv^{P,C} B$.

Proof. This construction precedes by stages. Each stage consists of two separate diagonalization steps: we diagonalize away from all potential p^C-isomorphisms from A to B, and we guarantee that A is not recognized by any P^C algorithm. As we do each diagonalization step, we maintain $\leq^P_m$-reductions $l : A \to B$ and $r : B \to A$, and we construct a $\leq^P_{1tt}$-reduction $e : SAT^C \to A\ (B)$. As we make decisions about A and B, we also include strings in C, so that both A and B are in NP^C.

We begin with the two many-one, polynomially computable functions, $l(x)$ and $r(x)$; when we have completed the construction, $l(x)$ will be a reduction from A to B, and $r(x)$ will be a reduction from B to A. To do this construction, we need the functions l and r to have several specific properties. To this end, we begin by defining two polynomially computable functions that possess the necessary properties. (After reading our construction the reader will easily be able to find many other functions that would do equally well.)

The reductions:



$$l(x) = \begin{cases} x & \text{if } x = 0w; \\ 1^{|x|} & \text{if } x = 1^{|x|}; \\ 10^{|x|-1} & \text{otherwise.} \end{cases} \qquad r(x) = \begin{cases} x & \text{if } x = 0w; \\ 10^{|x|-1} & \text{if } x = 10^{|x|-1}; \\ 1^{|x|} & \text{otherwise.} \end{cases}$$

With these definitions we are now ready to construct the oracle C, and sets A and B that are $\leq_m^P$-equivalent via the reductions l and r, but not p^C-isomorphic. While doing so, we also ensure that A and B are $\leq_{1tt}^P$-complete for NP^C, and that $P^C \neq NP^C$. The sets C, A and B are constructed in stages. Initially, C, A and B are in $\overline{A}$ ($\overline{B}$). During stage s of the construction we decide which elements in the intervals $I(n_s), I(n_s+1), ..., I(n_{s+1}-1)$ are in C, A and B. Thus, during stage s we do not alter any elements of C, A or B that are in the intervals below $I(n_s)$.

Notice that if l and r are reductions between A and B, there are some constraints placed on how we may add elements to A and B. For instance, any element beginning with 0 that is added to A must also be added to B, and vice versa. These isomorphic intervals of A and B are used to encode SAT^C. Furthermore, each subinterval of strings of the same length beginning with 1 (except 1^n) is a subset of either A or $\overline{A}$, since all of those strings map to the same string in B under the reduction l. Similarly, the subinterval of strings of the same length beginning with 1 (except 10^{n-1}) is a subset of either B or $\overline{B}$. We call these subintervals the *spoiling* intervals, to contrast them with the coding intervals.

- *$\leq^P_{1tt}$-completeness of A and B*

In order to define the encoding $e : SAT^C \rightarrow A$, we first define e_1 and e_2. Let e_1 code SAT^C into the lowest quarter of each interval, and e_2 code SAT^C into the second quarter. In other words, $e_1(x) = 00y$, and $e_2(x) = 01y$, for some y. Furthermore, for both encodings, $|e_i(x)| > q(|x|)$, where $q(|x|)$ is a strictly increasing polynomial bound on the run-time of a nondeterministic algorithm for $SAT^{()}(x)$. This prevents the computation of $SAT^C(x)$ from containing any queries about strings in C that might represent the encoding of $SAT^C(x)$.

For each n, e will be equal to one of e_1 or e_2 in $I(n)$. The unused coding subinterval will be empty. Given a set $T^C \in NP^C$, we can define a $\leq^P_{1tt}$-reduction g from T^C to A as follows. Let f be a $\leq^P_m$-reduction from T to SAT^C.

$$g(x) = \begin{cases} e_1(f(x)) & \text{if } e_1(f(x)) \in A; \\ e_2(f(x)) & \text{otherwise.} \end{cases}$$

If $e_1(f(x)) \in A$, then $f(x) \in SAT^C$, so $x \in T^C$. If $e_1(f(x)) \notin A$, then either $e = e_2$ in that interval, or $x \notin T^C$. In either case, $e_2(f(x)) \in A$ if and only if $x \in T^C$. Thus, A and B are $\leq^P_{1tt}$-complete for NP^C.

- $A, B \in NP^C$

If $x = 1y$, and $x \neq 1^n$, then $x \in A$ if and only if there is some string of length $|x|$ in C, of the form $10z$. If $x = 1^n$, then $x \in A$ if and only if there is some string of length $|x|$ in C of the form $11z$. If $x = 00y$, then $x \in A$ if and only if there is some string of length $|x|$ in C of the form $00z$, and $(e_1 \circ f)^{-1}(x) \in SAT^C$. If $x = 01y$, then $x \in A$ if and only if there is some string of length $|x|$ in C of the form $01z$, and $(e_1 \circ f)^{-1}(x) \in SAT^C$. To decide whether $x \in B$, simply decide whether $r(x) \in A$.

- *Sparseness*

The oracle C contains at most three strings of any given length. For each n, C contains a string of length n of the form

- $00z$, if $e = e_1$ on $I(n)$; or
- $01z$, if $e = e_2$ on $I(n)$;
- $10z$, if the spoiling interval of $I(n)$ is contained in A; and
- $11z$, if the spoiling interval of $I(n)$ is contained in B.

Thus C is sparse.

We include spoiling intervals in A and B for one of two reasons: to spoil a potential p^C-isomorphism between A and B, or to guarantee that A and B are not in P^C.

- *Initialization and specification*

We begin by fixing an enumeration of polynomial time functions. Let $p_i(|x|)$ be a strictly increasing polynomial bound on the run-time of $f_i^{()}(x)$. We require that our enumeration satisfy $p_i(n) < 2^{n-3}$ for all $n > i$.

Next, we fix the enumeration of polynomial time oracle Turing machines, $M_i^{()}$. Let $q_i(|x|)$ be a strictly increasing polynomial bound on the run-time of $M_i^{()}(x)$. We require that $q_i(n) < 2^{n-3}$ for all $n > i$.

- *The construction*

Stage 0. $C = A = B = \emptyset$.

Stage s. As we mentioned at the beginning of the proof, stage s consists of two steps, which we refer to as the diagonalization step and the spoiling step. The diagonalization steps guarantee that each deterministic polynomial time oracle Turing machine T_i^C does not recognize A, so $P^C \neq NP^C$, and the spoiling steps guarantee that A and B are not p^C-isomorphic. At the end of each step, we code SAT^C into A and B so that they will be $\leq_{1tt}^P$-complete for NP^C. We begin with the spoiling step.

- $A \not\cong^{P,C} B$

Let $f_1^{()}, f_2^{()}, f_3^{()}, \ldots$ be an enumeration of polynomially computable functions, as specified above. We need to ensure that any pair, $\langle f_i^C, f_j^C \rangle$, of functions that are mutual inverses are not reductions between A and B. To do this we consider one pair $\langle f_i^C, f_j^C \rangle$ at each stage.

At stage s, let $\langle f_i^C, f_j^C \rangle$ be the first pair of functions that have not been considered at an earlier stage. We begin by verifying that the pair behave as mutual inverses on all elements on intervals $I(1)$ up to $I(n_s)$, and that they are reductions between A_s and B_s. If either of these conditions fail, then f_i^C and f_j^C cannot form an isomorphism between A and B, so we are done with the spoiling step of stage s. Otherwise, we must add elements to A and B to spoil the possibility that f_i^C and f_j^C form a p^C-isomorphism.

If f_i^C is one-one and we compute f_i^C on the upper half of $I(n_s)$, then there must be strings where f_i^C and l do not agree. If we let x be the least element in that interval such that $f_i^C(x) \neq 10^{n_s-1}$, and we ensure that $A(x) \neq B(f_i^C(x))$, then we will have guaranteed that $f_i^C : A \not\to B$. If $f_i^C(x)$ queries C about some z of length $\geq n_s$, we immediately *restrain* z from C. Because of the enumeration of $f_i^{()}$'s we have chosen, this cannot restrain all of the strings that could represent y, for any y of length $\geq n$.

To ensure that $A(x) \neq B(f_i^C(x))$, we do the following.

Case 1: $f_i^C(x)$ is in an interval less than $I(n_s)$. Then $B(f_i(x))$ was decided at an earlier stage, so we simply add x to A if and only if $f_i^C(x)$ is not in B.

Case 2: $f_i^C(x)$ is in a spoiling interval greater than or equal to $I(n_s)$. Then neither x nor $f_i^C(x)$ have been added to A or B. Therefore, we simply add x to A and $f_i^C(x)$ to $\overline{B}$.

Case 3: $f_i^C(x)$ is in a *coding* interval greater than or equal to $I(n_s)$. Then we fix e on that interval so that $f_i^C(x)$ is not in the range of e. This implies that $f_i^C(x) \notin B$, so we add x to A.

Note that in *Case 2* or *Case 3*, $f_i^C(x)$ may be in an interval that is greater than $I(n_s)$. Furthermore, $f_i^C(x)$ may have queried a string z in an interval greater than $I(n_s)$. Let $n_s + m$ be the length of the longest string affected by the computation. Now, during the remainder of this step of stage s, we decide which elements in the intervals $I(n_s), ..., I(n_s + m)$ are in A and B, and which elements are in C. Our first task is to add to A, B, $\overline{A}$ and $\overline{B}$, elements whose membership is tied to the decisions we made about x and $f_i^C(x)$. For the other intervals, the even-length spoiling intervals are included in A and $\overline{B}$, and the odd-length intervals vice-versa. Next we choose the least representative in C that has not been restrained for each spoiling interval we have added to A or B, and for each interval of e.

We observe that, once C, A, and B are constructed, $A \not\cong^{P,C} B$. Suppose this were not the case. Let f^C be a p^C-isomorphism from A to B. Let $s_1, s_2, s_3, ...$ be an infinite sequence such that at each stage s_k, the functions f_i^C and f_j^C considered are equal to f^C and $(f^C)^{-1}$. Then at each stage s_k, for each k, there is a corresponding x such that $A(x) \neq B(f^C(x))$. This contradicts our assumption that f^C was a p^C-isomorphism from A to B. Thus $A \not\cong^{P,C} B$.

- $P^C \neq NP^C$

This step in the construction is again modeled on the construction in [BGS-75] of an oracle to separate P and NP. It is similar to the spoiling step. At stage s, suppose that the spoiling step of the stage has decided membership in A, B, and C for all strings up to $I(n_s + m + 1)$. Let x be a string in the spoiling interval of $I(n_s + m + 1)$ of A. Let $M_s^{()}$ be the s^{th} deterministic polynomial time oracle Turing machine in our enumeration. We run $M_s^C(x)$, restraining any z of length $> n_s + m$ that is queried. We set $A(x) = 1 - M_s^C(x)$. This guarantees that M_s^C does not recognize A. Since each set recognized by a deterministic polynomial time oracle program is recognized by infinitely many programs

in our enumeration, this construction guarantees that any such program with oracle C is wrong about A infinitely often. Furthermore, since $A \leq_m^P B$, this also guarantees that any such program with oracle C is wrong about B infinitely often. Thus A and $B \in NP^C - P^C$.

To finish stage s, we decide membership in C, A and B of all strings up to $I(n_{s+1})$, where n_{s+1} is one more than the longest string affected by this step. Then we let $e = e_1$ on those intervals, and we decide the remaining spoiling intervals on the basis of parity, as before. Finally, we choose the least representative that has not been restrained for each subinterval, and include it in C, if a representative is required. Notice that e is a one-one, polynomially bounded reduction from SAT^C to A and B, so each is thick. ∎

Observation 2.3.2. *In the construction above, the oracle C is in E.*

Observation 2.3.3. *In the construction above, $P^C \neq UP^C$.*

The set consisting of the spoiling subintervals of A is in $UP^C - P^C$.

Because we have encorporated Baker, Gill and Solovay's construction into the construction of Theorem 2.3.1, the question arises of whether we can show that this result holds with probability one, in the style of [BG-81]. However, Bennet and Gill's Lemma 1 does not apply here, since the set A violates Condition 4 of their Lemma 1: $SAT^C \subset A$, and SAT^C is a p-cylinder, so each bit of C decides infinitely many bits of SAT^C, and hence of A. Kozen and Machtey [KM-80] give an alternative characterization of the oracle properties that hold with probability one. They show that a diagonalization language L is not in a class M if and only if for each $M_i \in$ M, the set of oracles A such that $L^A = M_i^A$ is nowhere dense. In the construction for Theorem 2.3.1, we have the requirement that there is a witness for e, for each interval. In other words, we set aside some polynomially recognizable subset E of C, such that, for each n, $e = e_1$ on $I(n)$ if there is a witness in some specified subset of E, and $e = e_2$ on $I(n)$ if there is a witness in some other specified subset

of E; there must be a witness for at least one of these. This necessity makes the set of acceptable extensions of C_s nowhere dense. Thus, we cannot prove that there is an oracle C extending C_s that properly defines A and B, such that $L^C \neq M_i^C$. However, many of the constructions discussed in [BG-81], such as the Baker, Gill and Solovay construction, are *compatible* with this one. Our next theorem shows that this combined construction is robust, in the sense that it can be successfully combined with many other constructions. It is essential to note that the construction does not require the *addition* of *specific* elements to the oracle, although it does require that certain elements be *restrained* from the oracle.

The two properties that make the construction robust are the possibility of satisfying each requirement of the construction at whatever stage it is introduced, and the notion of a *restraint set.* It is possible to combine more than one such construction if the combined restraint sets do not interfere with any so-called "positive" requirements of including elements in the oracle. At least, they should not restrain *every* element.

Definition 2.3.4. *We say that an oracle construction* can accept a bounded restraint set *if there is some super-polynomial function $f(n) < 2^n$, such that it is possible to complete the construction, even if for each n, up to $f(n)$ strings of length n have been previously restrained from the oracle.*

It is sufficient in the following theorem that when we are combining two constructions, each one restrains no more than 1/4 of the strings of a given length.

The following definition adds more structure than we actually used in combining the two constructions above. We could have encoded A and B on the odd strings, and the Baker, Gill and Solovay set on the even strings. We separate them in the definition in order to see more clearly how the two constructions interact.

Definition 2.3.5. *We say that an oracle construction* can tolerate delays *if the oracle can be divided into two infinite, thick, polynomially recognizable sets, E and O, so that*

1) the construction can be carried out in E,

2) the construction can accept a bounded restraint set within E, and

3) the construction only affects O in a bounded restraint set.

The delays come from the other construction. By specifying that both constructions can accept delays, we guarantee that a requirement from one construction will not *injure* one from the other construction. This is important, since it would be difficult to confine injuries induced by another, independent construction. A typical forcing argument (in the sense of Torenvliet and Van Emde Boas [TV-86]), such as Kurtz's construction of a strongly bi-immune set, *cannot* tolerate delays, whereas a straightforward diagonalization argument can.

Corollary 2.3.6. *Any oracle construction that can tolerate delays can be combined with the oracle construction of Theorem 2.3.1.*

To prove this Corollary, we need only go through the proof of Theorem 2.3.1, replacing the Baker, Gill and Solovay construction by something else, or *combining* it with a new construction. Notice that there is a lot of flexibility in the old construction. A spoiling requirement can be satisfied whenever it is introduced. Therefore, we can delay the old construction, if that will simplify the new one. In fact, we can even allow the new construction to insert elements into some of the even intervals, as long as the intervals are syntactically correct for the old construction and there are infinitely many intervals left for the old construction. In other words, the new construction may define arbitrary intervals of E (and thus of A and B), so long as those intervals are in the form of the intervals determined by the construction in Theorem 2.3.1, and at each stage, the old construction can determine *some* intervals of E. It is likely that the new A and B will be different from the old A and B, but the crucial properties will still hold, that $A \not\cong^{P,C} B$, and that A and B are $\leq^P_{1tt}$-complete for NP^C. The one property that may be lost in the combination is

the sparseness of the oracle. This is evident, for instance, in the proof sketch for Corollary 2.3.8.

Corollary 2.3.7. *There is a modification of the oracle C in Theorem 2.3.1, such that every infinite set in NP^C (or $NP^C \cup coNP^C$) has an infinite P^C subset.*

Corollary 2.3.8. *There is a modification of the oracle C in Theorem 2.3.1, such that $NP^C = coNP^C$.*

Sketch of proof. We simply "hide" $coSAT^C$ in C so that it can be found by an NP^C algorithm but not by a P^C algorithm, and so that each x is represented by a string sufficiently long that $coSAT^C(x)$ cannot query its own representative. ∎

Corollary 2.3.9. *There is a modification of the oracle C in Theorem 2.3.1, such that $NP^C \neq coNP^C$.*

Sketch of proof. To show that $NP^C \neq coNP^C$, we construct an infinite set $S \in NP^C$ such that S intersects every infinite set in NP^C. Thus, $S \notin coNP^C$. The construction is the usual diagonalization with finite injury, except that we are diagonalizing over NP^C sets, T_i^C, rather than P^C sets. However, to find $x \in T_i^C$, we need only find *one* accepting computation of $T_i^C(x)$, so we only need to restrain one set of oracle queries of polynomial size. Therefore, the restraint set remains bounded. ∎

Corollary 2.3.10. *There is a modification of the oracle C in Theorem 2.3.1, such that $NE^C \neq E^C$.*

In [AFH-86], Ambos-Spies, Fleischhack, and Huwig attempted to unify many of the common diagonalization arguments over P. They described a class of diagonalization proofs, and showed that there are sets that have *all* of the properties specified by these diagonalizations. Most of the sets constructed in their paper are tally sets, and we will use this convention here.

Definition 2.3.11. *([AFH-86]) A property $\mathcal{Q}$ is enforced by* a p-standard diagonalization *if there is a sequence $\{C_e\}$ of sets in P such that for any tally set T, if for every $e \in \mathbf{N}$,*

$$(\exists^{\infty} s)\ (\exists i) \leq 1\ [T \upharpoonright s * \langle i \rangle \in C_e]$$

$$\Rightarrow\ (\exists s)\ [T \upharpoonright s \in C_e]$$

then T has property $\mathcal{Q}$. In other words, if T meets every C_e that gives it infinitely many chances to do so, then T has property $\mathcal{Q}$.

(Here, "$T \upharpoonright s$" means the binary string b of length s, where the k^{th} bit of b is 1 if and only if $0^k \in T$.)

Theorem 2.3.12. *Any property that is enforceable by a p-standard diagonalization is compatible with the construction in Theorem 2.3.1.*

Sketch of proof. We assume that the set constructed by the p-standard diagonalization depends on the second half of each interval of C: $T^A = \{0^k :\ \exists y\ |y| = k-1 \text{ and } 1y \in A\}$. We construct A, and T^A, by stages. At the beginning of stage s, we attempt to satisfy some active requirement, where a requirement R_e says that $T^A \upharpoonright s \in C_e^A$, and R_e is active if $e \leq s$ and it has not yet been met. If there is some e such that $T^A \upharpoonright n_s * \langle i \rangle \in C_e^A$ for $i = 0$ or 1, we fix A so that this holds for the least such e and i, restraining any string of length $\geq n_s$ that $C_e^A(T^A \upharpoonright n_s * \langle i \rangle)$ queries. This can easily be shown to be a bounded restraint set. We then proceed with the construction from Theorem 2.3.1 on the rest of the interval. At each stage, we define A on exactly one interval, given a bounded restraint set from previous intervals. This is a finite injury construction, since each time we satisfy one requirement of the new construction, we may ignore the possibility of satisfying higher numbered requirements on strings in that interval. Fortunately, we only care about those requirements that give us infinite opportunity to satisfy them. ∎

As corollaries to Theorem 2.3.12, we have that all of the constructions listed in [AFH-86] are compatible with the construction in Theorem 2.3.1.

Corollary 2.3.13. *There is a modification of the oracle C in Theorem 2.3.1 such that there is a p^C-bi-immune set in NP^C.*

Corollary 2.3.14. *There is a modification of the oracle C in Theorem 2.3.1 such that there is a set $D \in NP^C$ that is not auto-p^C-reducible. In other words, if $f^C : D \to D$ is a $\leq_m^{P,C}$-reduction, then f^C is the identity almost everywhere.*

We will give the definitions of p-selective and NT sets in §3.2. The relevant fact is that, in each case, there is a polynomial algorithm that relates the membership of two elements in such a set. Thus, a p-standard diagonalization will avoid both properties.

Corollary 2.3.15. *There is a modification of the oracle C in Theorem 2.3.1 such that there is a set in NP^C that is not p^C-selective.*

Corollary 2.3.16. *There is a modification of the oracle C in Theorem 2.3.1 such that there is a set in NP^C that is not in NT^C.*

There are a number of oracle constructions that are not compatible with this one. Several such constructions involve coding information into polynomially recognizable subsets of the oracle, instead of hiding it, as we have done in the constructions above. The constructions that are incompatible with the construction in Theorem 2.3.1 tend to be compatible with the construction in Theorem 2.2.4 that separated the $\leq_m^{P,D}$-complete sets. (That construction involved coding exponentially long intervals of SAT^D into the oracle D.) In fact, one example of this other sort of compatibility is Theorem 2.2.4. Another example comes from [HHem-87], in which $P^D = UP^D$. Since most diagonalization constructions of oracles take one of these two forms (sparse "hidings" versus polynomial codings), it seems

that separating some class of NP^C-complete sets does not limit our other diagonalization options.

In [KLD-86], Ko, Long and Du constructed sets that are one-one equivalent, not p-isomorphic, and $\leq^P_{btt}$-complete for E, using a one-way function. The proof of Theorem 2.3.1 yields a simpler proof that there are many-one equivalent $\leq^P_{btt}$-complete sets for E that are not p-isomorphic. In their construction, each diagonalization step affects infinitely many bits of A and B; in our construction, each diagonalization directly affects only two subintervals.

The sets we construct are not necessarily equivalent under *one-one* p-reductions, but our proof does *not* require the existence of a one-way function. Kurtz, Mahaney and Royer have a proof of the same result in [KMR-86] that relies on the existence of p-immune sets in $TIME[2^{cn}]$. While their proof is not necessarily difficult, ours is more direct, and gives more intuition about the sets constructed.

Theorem 2.3.17. *There are sets A and $B \in E$ such that*

i) A and B are $\leq^P_{1tt}$-complete for E, and

ii) $A \equiv^P_m B$, but $A \not\equiv^P B$.

Sketch of proof. The construction is a simpler version of the construction in Theorem 2.3.1. Again, we start with the reductions $l : A \to B$ and $r : B \to A$, and construct A and B accordingly. We do not have an oracle, so there are no restraint sets. We take care not to consider any f_i until x is large enough that the run-time of $f_i(x)$ is bounded by $2^{|x|}$, so the construction stays in exponential time. Finally, instead of letting e map SAT into A, we choose some set K that is $\leq^P_m$-complete for E to take the place of SAT. ∎

Notice that we did not mention thickness in Theorem 2.3.11. By Berman's theorem [Ber-77], we know that all $\leq^P_m$-complete sets are thick and co-thick, so the reduction of E to A and B is sufficient to guarantee that both are thick and co-thick.

It is natural to ask at this point whether we could have constructed A and B so that they were $\leq_m^P$-complete. Unfortunately, we do not know how to do this. The difficulty here is that if A and B are to be $\leq_m^P$-complete, then the above construction must encode a copy of A into B and vice versa. However, all of the standard encoding methods result in reductions that are for all practical purposes one-one. This is in conflict with the part of the construction that ensured that A and B were not p-isomorphic by canceling one-one reductions between A and B. As we have mentioned before, this difficulty is not unique to our constructions.

A final consequence of the proof techniques of Theorem 2.3.1 is that there are $\leq_m^P$-degrees that do not consist of a single isomorphism type.

Corollary 2.3.18. *There is an $\leq_m^P$-degree that contains non-p-isomorphic sets.*

CHAPTER 3. NEAR-TESTABLE SETS

3.1 Introduction

The near-testable sets were introduced in [GJY-87] as part of a study of the effect of internal structure on the complexity of a set. A set A is near-testable if there is a polynomially computable function, t, that recognizes when its input, x, is on the boundary between A and $\overline{A}$. In other words, for each string x, $t(x) = 1$ if and only if exactly one of x and $x-1$ (the immediate predecessor of x) is in A. The internal structures considered in [GJY-87] are ordering structures induced by the self-reducible properties. The ordering structure on near-testable sets is simply the lexicographic ordering.

Any set in P is near-testable. The natural question to ask about near-testable sets is whether all near-testable sets are in P. I show (Theorem 3.3.1) that there are near-testable sets that are not in P if there are one-way functions.[12] In [Hem-87], Hemachandra extended this result to show that there are near-testable sets that are not in P if and only if there are sets in Parity-$P - P$.[13] These results were further extended in [GHJY-87] where it is shown that the collection of near-testable sets is one-one equivalent to the class Parity-P via p-invertible functions.

In [GJY-87], a subclass of the p-cheatable sets, the *(2 for 1)* p-cheatable sets, was described as a potential extension of the near-testable sets. That paper asked about the

[12] Recall that f is a one-way function if it is polynomially computable, polynomially honest, and one-one function from strings to strings that has no polynomially computable inverse.

[13] A set A is in Parity-P if there is a nondeterministic Turing machine M such that $x \in A$ if and only if $M(x)$ has an *odd number* of accepting computations. Parity-P was introduced in [PZ-82].

relationship between these two properties. I show (Theorem 3.4.1) that sets that are both (2 for 1) p-cheatable and near-testable are polynomially recognizable.[14] This theorem was extended in [GJY-88c] to show that *any* p-cheatable set that is near-testable is in P.

In [GJY-88c], we also considered other combinations of self-reducible properties. For instance, we showed that Turing self-reducible, p-cheatable sets are in P.[15] I have included several results of this type in §3.4 of this chapter. Although the near-testable sets are closely related to Parity-P, sufficient conditions for the existence of p-cheatable sets in Parity-$P - P$ are given. In combination with Theorem 3.4.1, this is reason to believe that (if Parity-$P \neq P$) Parity-P is not equal to the near-testable sets. I also give sufficient conditions for the existence of p-selective, near-testable sets that are not in P, and a construction of a p-cheatable, p-selective set that is not in P. Thus, not all combinations of self-reducibility properties yield characterizations of P.

[14] This was proved independently by Beigel, [Bei-87b].

[15] This was shown independently in [ABG-88].

3.2 Definitions

The sets we consider are either subsets of the natural numbers, or sets of strings over the alphabet Σ. Most frequently, we will use $\Sigma = \{0,1\}$. With any alphabet Σ, there is a natural ordering of Σ^*, namely the lexicographic order. Thus, all strings of length n precede all strings of length $n+1$, and the strings of length n are ordered by the "dictionary" order. This induces an isomorphism between Σ^* and the natural numbers. We use "$x-1$" to mean the immediate predecessor of x in this order.

Definition 3.2.1. *A set A is* near-testable *($A \in NT$) if there is a polynomially computable function that given x decides whether exactly one of x and $x-1$ is in A. That is, the function*

$$t(x) = \chi_A(x) + \chi_A(x-1) \ (mod\ 2)$$

is polynomially computable.

Thus, with respect to the lexicographic ordering on Σ^*, we can fully relate the membership questions for x and $x-1$. Near-testability is a special case of Balcázar's *word decreasing query self-reducibility* [Bal-87], an independently defined concept.

Observation 3.2.2. *([Bal-87], [GJY-87])*

(i) If $A \in P$, then $A \in NT$;

(ii) if $A \in NT$, then $A \in E \cap PSPACE$;

(iii) there is an $A \in E - NT$.

A useful concept in the study of near-testable sets is the *boundary* of a set. Given a set A, the boundary of A will contain the first element of each contiguous sequence of elements of A and of $\overline{A}$ (except 0). Thus, we have

$$boundary(A) = \{x : \text{exactly one of } x \text{ and } x-1 \in A\}.$$

We can picture a set A for which $0 \in \overline{A}$ as follows:

The thinner bars represent $\overline{A}$, the thicker bars represent A, and the thin vertical lines are elements of $boundary(A)$. Notice that $boundary(A) = boundary(\overline{A})$.

Observation 3.2.3. *$A \in NT$ if and only if $boundary(A) \in P$.*

Definition 3.2.4. *([BGGO-86, et al.]) A is (n for k) p-cheatable if there is a polynomial time oracle Turing machine M^A that on input $x_1 ... x_n$, outputs $\chi_A(x_1), ... \chi_A(x_n)$, and makes only k queries to the oracle. We say that A is p-cheatable if A is (2^k for k) p-cheatable for some k.*

Beigel actually defines a set to be p-cheatable if there is *some* oracle B and polynomial time oracle Turing machine $M^{()}()$ such that M^B on input $x_1 ... x_{2^k}$ outputs $\chi_A(x_1), ... \chi_A(x_{2^k})$, and makes only k queries to the oracle.

Observation 3.2.5. *([Be-87a], [GJY-87a]) A set A is (2 for 1) p-cheatable if and only if there is a function associated with A, of the form*

$$g(x,y) = \begin{cases} \chi_A(x) + \chi_A(y) \pmod 2, & \text{or} \\ \chi_A(x), & \text{or} \\ \chi_A(y). & \end{cases}$$

Proof. If there is such a function g, then it is easy to see that there is a *(2 for 1)* algorithm that makes use of g.

So let us assume that we have a *(2 for 1)* algorithm $M^A(x,y)$. We construct g by *simulating* M^A, without actually querying A. Since $M^A(x,y)$ makes only one query to the oracle, we consider both possible outcomes of the calculation. Each outcome contains $\chi_A(x)$ and $\chi_A(y)$. If the two outcomes agree on one of these, then we have absolute information about membership of that string in A. Otherwise, we learn that $x \in A$ if and only if $y \in / \notin A$. ∎

The above observation can be generalized, to show that for any (2^k for k) p-cheatable set there is an associated function that, on input $(x_1, ...x_{2^k})$, either outputs absolute information about membership in A of one string, or relates $\chi_A(x_i) \oplus \chi_A(x_j)$ $(i < j \leq 2^k)$ to the membership information for the other inputs. (See [GJY-87], or [GJY-88a, Theorem 1, Lemma 1].) Unlike the near-testable sets, p-cheatable sets may have high time complexity.

Theorem 3.2.6. *([BGO-87])*

i. There are p-cheatable sets of arbitrarily high time complexity, and

ii. all p-cheatable sets are decidable.

There is one other type of self-reducible set that we will consider in §3.4. This is the *p-selective* sets, which were introduced by Selman in [Sel-79]. Selman's definition is the polynomial time analog of the *semi-recursive* sets introduced by Jockusch [Joc-68].

Definition 3.2.7. *A is* p-selective *if there is a polynomial computable function $s(x, y)$ such that*

i) $s(x, y) \in \{x, y\}$, and

ii) if $x \in A$ or $y \in A$, then $s(x, y) \in A$.

Selman and Ko have both given structural characterizations of the p-selective sets. We present the most general characterization, from [Ko-83]. First, we need some background definitions.

Definition 3.2.8. *([Ko-83]) A* preorder *R on Σ^* is* partially polynomial-time computable *if there is a polynomial-time computable function f such that*

(a) $f(x, y) = f(y, x) = x$, if xRy but not yRx,

(b) $f(x, y) = f(y, x) \in \{x, y\}$, if xRy and yRx, and

(c) $f(x, y) = \#$, if neither xRy nor yRx, where $\#$ is a special symbol not in Σ.

Given a preorder R, we define an equivalence S, where xSy if and only if xRy and

yRx. We can then define the relation R' on Σ^*/S in the usual manner. We call S and R' the equivalence relation and the partial order *induced* by R, respectively.

Theorem 3.2.9. *([Ko-83]) A set $A \subseteq \Sigma^*$ is p-selective if and only if there is a partially polynomial time computable preorder R in Σ^* such that if S and R' are the induced equivalence relation and partial ordering as defined, then*

a. *R' is a linear ordering, and*

b. *A is the union of an initial segment of $(\Sigma^*/S, R')$.*

Thus, we see that A is an initial segment of the $<_{R'}$ ordering. This shows that p-selectivity induces an ordering structure on sets, just as 2-conjunctive truth-table self-reducibility induces an ordering on SAT.

3.3 One-Way Functions and NT

To construct a near-testable set that is not in P, we use an idea from the study of NP. If $A \in NP$, then it may be difficult to know whether there is a witness to "$x \in A$", although it is easy to test whether y is such a witness. Thus, $\{\langle x, y\rangle : \ y \text{ witnesses } x \in A\}$ is in P, but finding each element of this set may be hard without exhaustive search. We use a one-way function to define a set with this and other useful properties.

Theorem 3.3.1. *If there is a one-way function, f, then there is a set $A \in NT - P$.*

Proof. We define A implicitly by using the function f to define *boundary*(A). In particular, we let *boundary*$(A) = \{\langle z, y\rangle : \ f(y) = z\}$, where we define the pairing function $\langle z, y\rangle$ to have certain specific properties. We define A by *boundary*(A), and $0 \notin A$. Clearly, *boundary*$(A) \in P$. We claim that A is not.

First, we define the pairing function $\langle z, y\rangle$. Let $q(n)$ be a strictly increasing polynomial bound for both f and f^{-1}. We are only interested in pairs (z, y) where y might be $f^{-1}(z)$. This is the case only when $|y| \leq q(|z|)$. We call a pair (z, y) *relevant* if $|y| \leq q(|z|)$. Let $\langle z, y\rangle = zy10^{q(|z|)-|y|}$ for relevant pairs (z, y). Notice that this divides the strings of length $n + 1 + q(n)$ into subintervals corresponding to each of the strings of length n.

Suppose $A \in P$. Then, to determine if $f(z)$ exists, we need only determine whether there is a boundary element of the form $\langle z, y\rangle$ for some y. We note that, by the definition of $\langle z, y\rangle$, the string $z0^{q(|z|)+1}$ is not in *boundary*(A), and is between any boundary elements of the form $\langle z - 1, y\rangle$ and those of the form $\langle z, y\rangle$. Notice also that for each z, there is at most one element of *boundary*(A) of the form $\langle z, y\rangle$. Thus, we can deduce that there is such an element (i.e., that $f^{-1}(z)$ exists) if exactly one of $(z-1)1^{q(|z-1|)+1}$ and $z0^{q(|z|)+1}$ is in A.

Furthermore, using the same reasoning as above, we can use binary search on the subinterval corresponding to z, to *find* $f^{-1}(z)$, if it does exist. This requires checking

if polynomially many strings are in A; if A were in P, we could polynomially invert f. Therefore, $A \in NT - P$. ∎

To prove the next result it is useful to introduce the notion of the parity of a set. Notice that the preceding proof shows the close connection between a set, the boundary of the set, and the *parity* of the boundary.

Definition 3.3.2. *Let $<$ be the lexicographic ordering on $\{0,1\}^*$. For any set B we define the parity of B as follows:*

$$parity_B(x) = \begin{cases} 0 & \text{if } |\{y \leq x : y \in B\}| \text{ is even;} \\ 1 & \text{otherwise.} \end{cases}$$

Then, identifying a set with its characteristic function, it is easy to see for any set A that

$$0 \notin A \Rightarrow \ A = parity_{boundary(A)}, \quad \text{and}$$

$$0 \in A \Rightarrow \ \overline{A} = parity_{boundary(A)}.$$

Thus, $A \in P$ if and only if the parity of the boundary of A is in P.

Theorem 3.3.3. *If f in the previous construction is both one-way and onto i.e. if $f^{-1}(z)$ is uniquely defined for each z, then $A \in (NP \cap coNP) - P$.*[16]

Proof. We observe that, if f in the previous proof is onto, then there are an even number of boundary elements of length $n + 1 + q(n)$ for each n, namely 2^n of them. Thus

$$parity_{boundary(A)}(1^{n+1+q(n)}) = 0,$$

[16] Grollman and Selman show in Theorem 11 of [GS-84] that the existence of such a function is equivalent to $P \neq UP \cap coUP$. Our theorem is a variation of a result from Brassard, Fortune, and Hopcroft [1978], which is quoted as Exercise 13.24 in Hopcroft and Ullman [HU-79], which asks the reader to prove that $\{\langle x, y\rangle : \ f^{-1}(x) > y\}$, f a one-way function, is in $(NP \cap coNP) - P$.

and for $|z| \neq n + 1 + q(n)$ for any n,

$$parity_{boundary(A)}(z) = 0,$$

i.e., $z \notin A$.

Suppose we are given a string of the form wz, where $|w| = 1 + q(|z|)$. To decide whether $zw \in A$, we need to know

$$parity_{boundary(A)}(zw).$$

This can be broken down into two questions. The first is, "how many strings (mod 2) of length $|z|$ precede z?" The second is, "is $f^{-1}(z) < w$?" The answer to the first question is simply the parity of z. To answer the second, we guess $y = f^{-1}(z)$, and verify. This nondeterministic polynomial time procedure provides a witness for $z \in A$ *or* for $z \notin A$. Thus, $A \in NP \cap coNP$. By Theorem 3.3.1, we know that $A \notin P$. ■

In the proof of Theorem 3.3.1, we used the fact that there was either one or no boundary string in each "z-interval." We could have used any set B, such that B is defined by *an odd number* of strings in each interval. We use Parity-P to describe such sets.

Papadimitriou and Zachos refer to Parity-P as "a more moderate version" of Valiant's class $\#P$, the class of functions, $f(x)$, that count the number of accepting paths produced by a nondeterministic polynomial time Turing machine on input x [Val-79]. Like $\#P$, Parity-P is in $PSPACE$, and is thought not to be contained in the polynomial time hierarchy. Clearly, if we could compute $\#P$, then we could compute Parity-P. However, there is no obvious reason that the ability to distinguish between an odd and even number of accepting paths would help in computing the total number of accepting paths.

After seeing the previous proof, Hemachandra pointed out that the near-testable sets are in Parity-P, and that all sets in Parity-P can be uniformly $\leq_{tt}^{P}$-reduced to near-testable sets. Notice that the class UP is a subclass of Parity-P. Thus, the following theorem,

proved jointly by Goldsmith, Hemachandra, Joseph, and Young, is an extension of Theorem 3.3.1, and of Hemachandra's result.

Theorem 3.3.4. *([GHJY-87]) Every set in Parity-P is $\leq_1^P$-reducible to a near-testable set via a p-invertible reduction.*

Proof. This proof is a variation on the proof of Theorem 3.3.1. Let $S \in$ Parity-P. Let M be the parity acceptor for S, and let $q(|x|)$ be a strictly increasing polynomial bound on the run-time of $M(x)$. We can encode a computation, c, of $M(x)$ as a binary string of length $q(|x|)$, that describes a path through the nondeterministic computation tree, padded by the appropriate number of zeroes. We define $T \in NT$ implicitly, by $0 \notin T$, and

$$\begin{aligned} \mathit{boundary}(T) = \{\langle x, i, c\rangle : \ & c \text{ codes an accepting computation of } M(x), \\ & \text{and } i \in \{0,1\}.\} \end{aligned}$$

Here, we use $\langle x, i, c\rangle$ to abbreviate $x1i1c$.

We include i so that each accepting computation is encoded *twice.* Since $0 \notin A$, this means that $x0^{q(|x|)+3} \notin T$ and $x1^{q(|x|)+3} \notin T$ for each x. However, the "midpoint" between these two strings, $x110^{q(|x|)+1}$, is in T if and only if there are an *odd number* of accepting computations of $M(x)$. If we let $f(x) = x110^{q(|x|)+1}$, then f is a $\leq_1^P$-reduction from S to T, which is easily invertible. ∎

Corollary 3.3.5. *([GHJY-87]) $NT = P$ if and only if Parity-$P \neq P$.*

3.4 Characterizing P Using Self-Reducibility

In [GJY-87], we asked whether the (2 for 1) p-cheatable sets were an extension of the near-testable sets. We show that this is not the case in Theorem 3.4.1. Beigel proved this theorem independently in [Bei-87b], and announced it in [ABG-87].

Theorem 3.4.1. *All sets that are both (2 for 1) p-cheatable and near-testable are in P.*

Proof. Let A be near-testable and (2 for 1) p-cheatable. Let t and g be the corresponding functions. To decide whether x is in A, we do the following. We assume that we know if 0 is in A. We maintain two variables, a and b. At each stage, we know whether $a \in A$, and we know $\chi_A(b) \oplus \chi_A(x)$. To begin, we set $a = 0$ and $b = x$. If $a < b - 1$, then let c be the midpoint between a and b, and compute the *(2 for 1)* function $g(c, b)$. If $g(c, b)$ outputs $\chi_A(b)$, we are done. If it outputs $\chi_A(c)$, we let $a := c$, and if it outputs $\chi_A(c) \oplus \chi_A(b)$, we compute $\chi_A(c) \oplus \chi_A(x)$, and let $b := c$.

When $a = b - 1$, we run the near-testing function $t(b) = \chi_A(b) \oplus \chi_A(a)$, and use the information about $\chi_A(a)$ to deduce $\chi_A(b)$, and thus $\chi_A(x)$.

We apply $g(c, b)$ at most $log(x) + 1$ times, and $t(b)$ once, where $b, c \leq x$. Thus, this algorithm is polynomial in $|x|$, i.e., $A \in P$. ∎

This result can be extended to show that all p-cheatable sets that are near-testable are in P.

Theorem 3.4.2. *([GJY-87]) All sets that are both p-cheatable and near-testable are in P. In other words, P consists of those sets that are both p-cheatable and near-testable.*

The only examples we have of non-trivial p-cheatable sets are sets with such sparse boundaries that, given two elements of such a set that are separated by a boundary element, then in time polynomial in the length of the larger one, we can recapitulate the construction of the set, up to the smaller string. (These sets are either very sparse, or else have very

long sequences of elements whose membership depends on a few elements.) For sets in $E - P$, we formalize this as follows.

Definition 3.4.3. *S is* log*-sparse *if, whenever $x, y \in S$, $x < y$ implies that $2^{|x|} \leq |y|$.*

If we let $f(0) = 2$, and $f(n+1) = 2^{f(n)}$, then the tally set $\{1^{f(n)}\}$ is log*-sparse.

We can construct a set $S \in E - P$ by diagonalization. If $S \subseteq \{1^{f(n)}\}$, then S is (2 for 1) p-cheatable as well: for $n > m$, in time polynomial in $f(n)$, we can simulate the m^{th} diagonalization, and decide whether $1^{f(m)} \in S$. Thus, the *(2 for 1)* algorithm for S need only query S about $1^{f(n)}$. Similarly, S is p-selective.

Observation 3.4.4. *([GJY-88]) If there is a set in Parity-$P - P$ that is contained in a log*-sparse set in P, then there is a set in Parity-P that is (2 for 1) p-cheatable, yet is not in P.*

Although we have shown that the collection of near-testable sets is one-one interreducible with the class Parity-P, this is one indication that they are not the same. The following two observations provide further indication that they are different.

Observation 3.4.5. *There are sparse sets in Parity-$P - P$ if and only if there are sets in Parity-$E - E$.*

The proof of Observation 3.4.5 follows the proof of the analogous theorem for NP in [HIS-85]. The only difference is that, in order to prove that Parity-$E = E$ implies that there are no sparse sets in Parity-$P - P$, we must take care to do the multiple Parity-time calculations *sequentially,* rather than in parallel. Thus, the multiple calculation will remain in Parity-time.

Observation 3.4.6. *If $S \in NT$ and S is sparse, then $S \in P$.*

Lastly, we consider the relationship between the near-testable sets and the p-selective sets. Again, we consider log*-sparse sets. This time, we want a set $S \in P$ such that there

is no polynomial algorithm that, on input 0^n, can *list* all of the elements of S of length n. Allender and Rubinstein [AR-86] proved that the existence of such sets is related to the existence of certain kinds of one-way functions.

Definition 3.4.7. *A sparse set S is* P-printable *if there is a polynomial time algorithm that, on input n, will generate all of the elements of S of length $\leq n$.*

Theorem 3.4.8. *If there is a log*-sparse set in P that is not P-printable, then there is a near-testable set that is p-selective, yet is not in P. Furthermore, by Theorem 3.4.2, this set cannot be p-cheatable.*

Proof. Let I_n be the interval of strings from length $f(n)$ up to length $f(n+1)$. Let $\hat{S} \in P$ be a log*-sparse set that is not P-printable. By the definition of log*-sparsity, we know that $|\hat{S} \cap I_n| \leq 1$ for each n.

Let $S_0 = \hat{S} \cap (\cup_n I_{3n})$, $S_1 = \hat{S} \cap (\cup_n I_{3n+1})$, and $S_2 = \hat{S} \cap (\cup_n I_{3n+2})$. Then $\hat{S} \cap S_i \in P$ for each i, and for at least one i, $\hat{S} \cap S_i$ is not P-printable. Without loss of generality, we will assume that $i = 0$, and set $S = \hat{S} \cap S_0$.

We define the set A as follows:

$$A = \{x : x = 0^{f(3n)}\} \cup \{x \in I_{3n} \text{ and } \exists y \in I_{3n} \cap S,\ y > x\}$$

$$\cup \{x : x \in I_{3n} \text{ or } I_{3n+1} \text{ and } I_{3n} \cap S = \emptyset.\}$$

In other words, if there is an element $y \in I_{3n} \cap S$, then it serves as a "breakpoint," dividing that interval into a left subinterval in A, and a right subinterval in $\overline{A}$. If there is no such element, then both I_{3n} and $I_{3n+1} \subset A$.

The boundary of A consists of

$$\{0^{f(3n)}\} \cup S \cup \{0^{f(3n)} + 1 : 0^{f(3n)} \in S\}$$

$$\cup \{0^{f(3n+2)} : I_{3n} \cap S = \emptyset.\}$$

By the definition of the I_k's, if $y = 0^{f(3n+2)}$, then in time polynomial in $|y|$, we can test whether there is an element of S in I_{3n}. Thus, $boundary(A) \in P$, so $A \in NT$.

The following function is a p-selection function for A:

$$s(x,y) = \begin{cases} x & \text{if } y \in I_{3n+2} \text{ for some } n, \text{ or} \\ & \text{if } x \in I_{3n} \cup I_{3n+1} \text{ and } y \in I_{3n} \cup I_{3n+1}, \\ & \quad \text{and } x < y, \text{ or} \\ & x \in I_{3n} \cup I_{3n+1},\ y \in I_{3m} \cup I_{3m+1},\ m > n, \\ & \quad \text{and } x \in A; \\ y & \text{otherwise.} \end{cases}$$

Note that, in the third case, we can test $x \in A$ in time polynomial in $|y|$.

Finally, suppose $A \in P$. Then the following algorithm P-prints S: for each k, if $f(3n) < k < f(3n+1)$ for some n, and if $0^k \in A$ and $0^{k+1} \notin A$, then there is an element of length k in S. If $f(3n) = k$, then there is an element of S of length k if and only if $0^{k+1} \notin A$. Otherwise, if there is an element of A of length k, we can use binary search to find it. Notice that this can be done in time polynomial in k, contradicting our assumption that S was not P-printable. Therefore, $A \notin P$. ∎

Observation 3.4.9. *There are sets that are p-cheatable, p-selective, and not in P.*[17]

Proof. Any subset of $\{1^{f(n)}\}$ that is in $EXP - P$ will be both p-cheatable and p-selective. The p-selector function is defined as follows, for $x < y$:

$$s(x,y) = \begin{cases} x & \text{if } y \neq 1^{f(n)} \text{ for any } n; \\ & \text{if } x = 1^{f(n)} \text{ and } y = 1^{f(m)} \text{ and } x \in A; \\ y & \text{otherwise.} \end{cases}$$ ∎

[17] [ABG-88] also observes that the properties of being p-cheatable and p-selective are independent of each other.

APPENDIX 1. NOTATION AND TERMINOLOGY

We use the following standard conventions: the letters i, j, k are indices; P_i, R_i, S_i, T_i and M_i are characteristic programs in some canonical enumeration of Turing machines; and f, g, h are functions from N to N or from $\{0,1\}^*$ to $\{0,1\}^*$, which are usually polynomially computable.

P and NP are the classes of polynomial and nondeterministic polynomial time decidable sets; a set $A \in P$ if there is some polynomial time program T_i such that $x \in A$ if and only if $T_i(x) = 1$. We frequently abuse notation and use T_i both as a program and as the set for which the program decides membership.

Thus,

$$P = \bigcup_c DTIME[n^c + c];$$

$$NP = \bigcup_c NTIME[n^c + c];$$

$$E = \bigcup_c DTIME[2^{cn}];$$

$$NE = \bigcup_c NTIME[2^{cn}];$$

$$EXP = \bigcup_c DTIME[2^{n^c}];$$

$$NEXP = \bigcup_c NTIME[2^{n^c}];$$

$$PSPACE = \bigcup_c SPACE[n^c + c];$$

$$Logspace = \bigcup_{c>0} DSPACE[logcn].$$

The class UP is the subclass of NP such that $A \in UP$ if and only if there is a nondeterministic polynomial time Turing machine M that recognizes A, such that $M(x)$ has at most one accepting computation for each input x.

In [PZ-83], Papadimitriou and Zachos defined the class *Parity-P*. A is in Parity-P if there is a nondeterministic polynomial time Turing machine M such that $x \in A$ if and only if there are an odd number of accepting computations of $M(x)$. This class has some similarities with the class *EP* defined by Goldschlager and Parberry [GP86]. *EP* is the class of sets computable by nondeterministic Turing machines extended by an *exclusive-or* operator. By adding an exclusive-or "gate" at the root of the computation tree of a nondeterministic Turing machine, the machine can determine the parity of the number of accepting computations. Since Goldschlager and Parberry did not require their machines to use the exclusive-or operator, *EP* contains both *NP* and Parity-P.

We say that a set A is *Turing self-reducible* if there is a polynomial time oracle Turing machine, $M^{()}()$, such that $M^A(x) = 1$ if and only if $x \in A$, and $M^A(x)$ only queries strings that are *shorter* than x.

In [Bal-87], Balcazar introduced the notion of *word-decreasing query or wdq-) self-reducible.* We say that a set A is wdq self-reducible if there is a polynomial time oracle Turing machine, $M^{()}()$, such that $M^A(x) = 1$ if and only if $x \in A$, and $M^A(x)$ only queries strings that *lexically precede* x.

In Chapter 1, we assume a uniform enumeration of polynomial time Turing machines with oracles (both deterministic and nondeterministic). In particular, we assume that, given a specification for $T_i^{()}$, we can determine a polynomial bound, $p_i(|x|)$, on the run-time of $T_i^{()}(x)$, independent of the oracle. This can be done by specifying that each $T_i^{()}$ is equipped with a polynomial clock that stops the calculation after a certain number of steps.

We discuss several types of polynomial reductions and equivalences. We say that $A \leq_m^P B$ if there is some many-one, polynomial time computable f such that $x \in A$ if and only if $f(x) \in B$. If there is a reduction $f : A \to B$ that is one-one and length-increasing, then we say $A \leq_{1,li}^P$. If there is polynomial one-one, onto, *polynomially invertible* reduction

from A to B, then A and B are *p-isomorphic* ($A \cong^P B$).

We also consider *bounded truth-table* reductions. We say $A \leq^P_{ktt} B$ if there is a polynomial time function f such that, on input x, f computes $x_1, ...x_k$ and Boolean function α, such that $x \in A$ if and only if $\alpha([x_1 \in B], ..., [x_k \in B]) = 1$. Notice that many-one reductions are a special case of the truth-table reductions.

For a given reduction, r, we say that A and B are polynomial r-equivalent ($A \equiv^P_r B$) if $A \leq^P_r B$ and $B \leq^P_r A$. The collection of all sets B such that $A \equiv^P_r B$ is called the *r-degree* of A.

We say that f is a *one-way function* if f is polynomially computable, polynomially honest (i.e. there is a polynomial $r(n)$ such that $|f^{-1}(x)| \leq r(|x|)$ for all x such that $f^{-1}(x)$ exists), one-one, and f^{-1} is *not* polynomially computable. We say that f is p-invertible if f is one-one and f^{-1} is polynomially computable.

A set A is *sparse* if there is a polynomial $p(n)$ such that, up to length n, there are no more than $p(n)$ elements in A. If $A \subseteq \{0\}^*$, (or $\{1\}^*$) then A is a *tally* set.

Finally, in our constructions we will consider the interval $I(n)$ of all binary strings of length n.

APPENDIX 2. RELATIVIZING COOK'S THEOREM

When studying properties of complexity classes, it is often useful to study the sets that are complete for those classes. Unfortunately, there are classes $\mathcal{C}$ and oracles A and B such that $\mathcal{C}^A$ has complete sets, and $\mathcal{C}^B$ does not. (See [HHem-86], for instance.) However, it is part of the folklore of the field that NP^A always has $\leq_m^P$-complete sets, and, what is more, that there is a particular set, SAT^A, that is always complete for NP^A. Schöning proved this in [Sch-81], although he did not give the full details of the proof. Therefore, we are including a more detailed version, that is useful for proofs in Chapter 2.

Our definition of SAT^A must include oracle queries. We formalize this with clauses of the form $(x_0x_2\overline{x}_3 \in A)$, or $(\overline{x}_3x_3\overline{x}_2x_2x_2 \notin A)$, where we interpret a "truth assignment" as a mapping of the variables to $\{0,1\}$. Thus, a string of symbols can be read as a binary string.[18]

Thus, SAT^A is the set in NP^A of "relativized Boolean" satisfiable formulas, i.e., formulas of the form

$$(x_0 \vee x_1 \vee x_2)\&(\overline{x}_1 \vee x_3)\&(x_0x_2\overline{x}_3 \in A)\& \text{ etc.}$$

A relativized Boolean formula is satisfiable *with respect to* A if there is a mapping of the variables that makes that formula a true sentence. In other words, the predicate $SAT^{(\ )}(\ )$ takes as arguments a formula $b(x_0,...x_n)$, and an oracle A. Therefore SAT^A is dependent on A, as well as on the satisfiability of the Boolean part of the expression.

Our proof relies heavily on the proof of Cook's Theorem given in [MY-78]. The proof in the unrelativized case requires that we show that any Turing machine is equivalent to a machine that has only one tape, always moves the tape head at each instruction execution,

[18] We assume that an oracle A is a subset of $\{0,1\}^*$.

and the tape is one-way infinite (so the tape has a first square, but no last square). It then shows that given any NP machine, we can uniformly encode a computation of the machine on a given input as a Boolean formula, which is satisfiable if and only if the machine accepts that particular input. Furthermore, this reduction is polynomial in the length of the input. The proof we present is a modification of this approach. We begin by standardizing our definition of an oracle Turing machine.

An oracle Turing machine instruction $\langle i, e, a_j, a_m, D, p\rangle$ specifies the current state, the tape in question (work or query), the symbol read, the symbol written, direction moved (all on the specified tape), and next state.

Definition A.2.1. *A* restricted nondeterministic Turing machine with oracle *has the following properties.*

1) The machine has two tapes, one for the input and workspace and one for the oracle.

2) The machine has states $0, ..., q, q+1, q+2, h(=q+3)$. *0 is the initial state;* q *is the oracle query state, the only state in which the machine queries the oracle;* $q+1$ *and* $q+2$ *are the only states that can immediately follow* q*, indicating yes and no answers, respectively, to the oracle query; and* h *is the unique halting state, so that* M *accepts string* x *on a given computation if and only if during that computation, the machine reaches state* h*, and whenever it enters state* h*, it halts.*

3) If $\langle i\ e\ a_j\ a_m\ D\ p\rangle$ *and* $\langle i\ e'\ a_j\ a_n\ D'\ p'\rangle$ *are two instructions in the machine, then* $e = e'$, $a_m = a_n$, *and* $D = D'$.

Proposition A.2.2. *Any nondeterministic Turing machine with oracle,* $M^{()}()$, *is polynomially equivalent to a restricted nondeterministic Turing machine with oracle.*

In other words, there is a restricted machine $M'^{()}$ and a polynomial $p(|x|)$ so that for any oracle A, and input x, $M^A(x)$ halts within $p(q(|x|))$ steps, where q is a polynomial bound on the run-time of the original machine.

Sketch of Proof. We assume that the original machine has only one oracle tape, and that it records replies to oracle queries in its states, instead of on tape. If there are more oracle tapes, we simulate the machine by making the extra oracle tapes into work-tapes, and copying the contents of the appropriate work-tape onto the oracle tape before querying the oracle. We then approximate $M'^{()}()$ by a one work-tape machine. We merge the read-write tapes (if there are more than one) into one tape by interleaving them (for two tapes, this corresponds to putting one on even squares, the other on the odds.) If the original machine has k read-write heads on its work-tape(s), we increase the tape alphabet, so that we can replace symbol a by a symbol a_i, specifying that the i^{th} head is reading symbol a. Then we replace each step in the original computation with a sweep of the tape (in $\mathcal{O}(q(|x|))$ time) in the new machine. For further details of machine manipulation, see [LP-81, 198-204, 324-6]. At worst, these transformations increase the run-time of $M'^{()}(x)$ to $\mathcal{O}(q(|x|)^2)$.

Next, by a similar procedure, we force this tape to be a one-way infinite tape (if it was originally two-way), essentially by folding it in half and intermeshing again.

The second condition implies that the following transitions never occur:

- $\langle q, e, a_j, a_m, D, p\rangle$,
- $\langle h, e, a_j, a_m, D, p\rangle$,
- $\langle i, e, a_j, a_m, D, q+1\rangle$ and
- $\langle i, e, a_j, a_m, D, q+2\rangle$.

A machine that satisfies this definition has its nondeterminism limited to the choice of the next state, and not the write instruction, the direction the tape head moves, or the choice of tapes. This last is easily implemented: each state i for which there are transitions for both tapes generates i_{work} and i_{query}; each of the transitions to i generates two transitions, that the computation chooses between nondeterministically. The rest can also be implemented by creating intermediary states where an unacceptable instruction

previously existed. Again, this may multiply the number of states by a small constant. The details are available in [MY-78, pp234-5]. ∎

Theorem A.2.3. *([Sch-81]) For any A, SAT^A is $\leq_m^P$-complete for NP^A.*

Proof. There are two parts to any proof that a given set is complete for a specified class. First, we must show that the set is in the class; second, we must show that any set in the class reduces to that set. In this case, the first part is easy.

- $SAT^A \in NP^A$

There is a polynomial time, nondeterministic Turing machine that on input $b(x_1...x_n)$ "guesses" an assignment for $x_1...x_n$, and checks if the resulting sentence is true. This checking can be done in linear time, with fewer than $|b(x_1...x_n)|$ queries to the oracle.

For the second part, we must show that any computation of a restricted Turing machine with oracle can be described by a relativized Boolean formula that is satisfiable if and only if that Turing machine accepts the given input.

As in Machtey and Young's proof for the unrelativized version, our goal is to produce, for machine M and input x, a relativized Boolean formula, which is satisfiable for the given oracle if and only if there is a halting computation of M on input x with that oracle. The construction is uniform for all M. We begin by describing Machtey and Young's proof.

- *The unrelativized version*

We construct a Boolean formula, based on the description of machine M, and input x, that is satisfiable if and only if there is an accepting computation of $M(x)$. The variables we use represent bits of information about the computation of $M(x)$. We describe the formula one conjunct at a time.

Given machine M, with polynomial bound $p(|x|)$, $h+1$ states, and alphabet symbols $b, a_1, ..., a_K$, b a blank, we set $m = p(|x|)$ for the given input x. We observe that we need

only consider times $t = 1, ..., m$, and m tape squares and $k+1$ symbols. We first construct an $m^2 * (k+1)$ matrix of Boolean variables giving each possible symbol for each square at any possible time:

$SYMB(t,i,j)$ is true if and only if, at time t, the i^{th} square contains the j^{th} symbol.

We construct two more matrices of variables of sizes m^2 and $m*(h+1)$, respectively, to specify the position of the head, and to specify the state of the machine at time t:

$HEAD(t,i)$ is true if and only if, at time t, the head is at square i, and

$STATE(t,s)$ is true if and only if, at time t, the machine is in state s.

In order to describe a valid computation, for each t and square i, exactly one $SYMB(t,i,j)$ must be true; similarly for each t, exactly one of $HEAD(t,i)$, and one of $STATE(t,s)$ must be true. We use the following formula to state this.

1) $JUSTONE(x_1, ...x_r)$ is true if and only if exactly one of $x_1, ...x_r$ is true.[19]

We need a total of $(k+1)+m+(h+1)$ of these.

Next we have formulas that express legal computations, based on the machine table: given time t, the head position, the state, and the symbol, we specify the valid write commands, move commands, and next states.

We say that, for each time t, square i, and symbol j,

2) $\neg SYMB(t,i,j) \vee HEAD(t,i) \vee SYMB(t+1,i,j)$. In other words, either the symbol on square i at time t remains there at time $t+1$, or the head is at that square at time t to change it.

We have one conjunction,

3) $\bigwedge_{0 \leq t \leq m} STATE(t,h)$,

which is true if and only if the computation halts by the m^{th} step.

---

[19] $JUSTONE(x_1, ...x_r)$ is shorthand for the CNF formula $(\neg x_1 \vee x_2 \vee ... \vee x_n)$ & $(x_1 \vee \neg x_2 \vee ... \vee x_n)$ & ... & $(x_1 \vee x_2 \vee ... \vee \neg x_n)$.

We have an formula that describes the initial condition of the tape, specifying the input, and that the rest of the m squares are blank. The first half of this formula depends on the input. The second half is

4) $\bigwedge_{|x|<i\leq m} SYMB(0,i,b)$.

Finally we have the transition table formulas:

5) $(HEAD(t,i)\ \&\ STATE(t,s)\ \&\ SYMB(t,i,j)$

$\rightarrow SYMB(t+1,i,a_q))$[20]

$\&\ (HEAD(t,i)\ \&\ STATE(t,s)\ \&\ SYMB(t,i,j)$

$\rightarrow STATE(t+1,r))$

$\&\ (HEAD(t,i)\ \&\ STATE(t,s)\ \&\ SYMB(t,i,j)$

$\rightarrow HEAD(t+1,i+(-1)^n))$.

The conjunction of the Boolean formulas described by 1–5 is satisfiable if and only if there is a valid halting computation of M on the given input, which accepts that input. It is a formula of length $\mathcal{O}(m^2)$, which is certainly polynomial in $|x|$.

- *The relativized version*

We have already described the legal transitions, and the matrices of possible states, symbols, and head positions on the work-tape. In addition, we have a set of variables to describe the oracle tape. We define the variables $OHEAD(t,i)$ and $OSYMB(t,i,j)$ in the obvious manner. Here again, t ranges from 1 to m, i from 0 to m. However, j may be one of the three symbols used on the oracle tape: 0, 1, and b.

For each t and i we have

---

[20] This could be written in conjunctive normal form as $\neg HEAD(t,i)\vee\neg STATE(t,s)\vee \neg SYMB(t,i,j)\vee SYMB(t_1,i,a_n)$.

6) $JUSTONE(OHEAD(t,1),...,OHEAD(t,m))$ &

$JUSTONE(OSYMB(t,i,1),OSYMB(t,i,0),OSYMB(t,i,b))$.

At time 0, and after each oracle call, the oracle tape is blank:

7) $\bigwedge_{1\leq i\leq m} OSYMB(0,i,b)$, and for each t,

$\bigwedge_{1\leq i\leq m}(\neg STATE(t,q) \vee OSYMB(t+1,i,b))$.

For each t, i, and j, we have

8) $\neg OSYMB(t,i,j) \vee OHEAD(t,i) \vee OSYMB(t+1,i,j)$. In other words, either the symbol on square i of the oracle tape at time t remains there at time $t+1$, or the oracle head is at that square at time t to change it.

We have formulas describing those transitions that involve the oracle tape without oracle queries.

9) $(OHEAD(t,i)$ & $OSTATE(t,s)$ & $OSYMB(t,i,j)$

$\rightarrow OSYMB(t+1,i,a_n))$

& $(OHEAD(t,i)$ & $OSTATE(t,s)$ & $OSYMB(t,i,j)$

$\rightarrow OSTATE(t+1,r))$

& $(OHEAD(t,i)$ & $OSTATE(t,s)$ & $OSYMB(t,i,j)$

$\rightarrow OHEAD(t+1,i+(-1)^n))$.

Finally, we describe the oracle queries. At time t, if M is in state q, we want to ask whether the string of nonblank characters on the oracle tape form a string in the oracle. We assume that the nonblank characters are 0's and 1's, and that they are on squares $q,...,n$ (for $n \leq m$), and that square $n+1$ contains a blank if $n < m$. If we know that the i^{th} square at time t contains a nonblank symbol, then $OSYMB(t,i,1) = 1$ (true) if the symbol is a 1, and $OSYMB(t,i,1) = 0$ if the symbol is a 0.

10) $(STATE(t,q) \wedge \neg OSYMB(t,1,b) \wedge OSYMB(t,2,b) \vee$

$\rightarrow [OSYMB(t,1,1) \in A])$

& $(STATE(t,q) \wedge \neg OSYMB(t,2,b) \wedge OSYMB(t,3,b) \vee$

$$\rightarrow [OSYMB(t,1,1)OSYMB(t,2,1) \in A]$$

$$\& \ (STATE(t,q)\neg \vee OSYMB(t,m,b)\wedge$$

$$\rightarrow [OSYMB(t,1,1)\cdots OSYMB(t,m,1) \in A]).$$

The length of the conjunction of the formulas described in 1–10 is still of order m^2. In fact, it is only twice as long as the previous one, and is satisfiable if and only if there is an accepting computation of $M^A(x)$. Therefore, any NP^A set may be mapped into the set of satisfiable relativized Boolean formulas, and its complement mapped to the complement of SAT^A, making $SAT^A \leq_m^P$-complete for NP^A. ∎

BIBLIOGRAPHY

[AR-86] E. Allender, "Isomorphisms and *1-L* reductions," *Journal of Computer and System Sciences,* to appear. (First published in *Structure in Complexity Conference,* Springer Verlag Lecture Notes in Computer Science **223** (1986), 12-22.)

[AR-86] E. Allender, and R. Rubinstein, "P-printable Sets", to appear in *SIAM Journal of Computing* (Preliminary version appeared as "The complexity of sparse sets in P," *Structure in Complexity Conference,* Springer Verlag Lecture Notes in Computer Science **223** (1986), 1-11.)

[AFH-86] K. Ambos-Spies, H. Fleischhack, and H. Huwig, "Diagonalizations over polynomial time computable sets," *Preprint* (1986), 1-54. (Also accepted by *Theoretical Computer Science.*)

[ABG-87] A. Amir, R. Beigel, and W. Gasarch, "Polynomial terse sets II," *Manuscript* (1987).

[ABG-88] A. Amir, R. Beigel, and W. Gasarch, "Cheatable, p-terse, and p-superterse sets," *University of Maryland Technical Report TR-2090* (1988), 1-23.

[BGS-75] T. Baker, J. Gill and R. Solovay, "Relativizations of the $P = NP$ question," *SIAM Journal of Computing* **4** (1975), 431-442.

[Bal-87] J. Balcázar, "Self-reducibility," *Symposium on the Theoretical Aspects of Computer Science,* Springer Verlag Lecture Notes in Computer Science **247** (1987), 136-147.

[BS-85] J. Balcázar, and U. Schöning, "Bi-immune sets for complexity classes," *Mathematical Systems Theory* **18** (1985), 1-10.

[Bei-87a] R. Beigel, "Bi-immunity and separation results for cheatable sets," *Preprint* (June 87), 1-15.

[Bei-87b] R. Beigel, "A note on some open problems of Goldsmith, Joseph, and Young," *Working draft* (October 1987), 1-4.

[BGGO-86] R. Beigel, W. Gasarch, J. Gill and J. Owings, "Verbose, terse sets and

superterse sets," *Manuscript* (1986).

[BGGO-87] R. Beigel, W. Gasarch, J. Gill and J. Owings, "Terse, superterse and verbose sets," *University of Maryland Technical Report TR-1806* (March 1987), 1-25.

[BGH-87] R. Beigel, W. Gasarch, and L. Hay, "Bounded query classes and the difference hierarchy," *University of Maryland Technical Report TR-1847,*

[BGO-87] R. Beigel, W. Gasarch, and J. Owings, "Terse sets and verbose sets," *Recursive Function Theory: Newsletter* **36** (Feb. 1987), 13-14. (1987), 1-26.

[BG-81] C. Bennet and J. Gill, "Relative to a random oracle A, $P^A \neq NP^A$ with probability one," *SIAM Journal of Computing* **10** (1981), 96-113.

[Ber-77] L. Berman, *Polynomial reducibilities and complete sets,* Ph.D. Thesis, Cornell University, 1977.

[BH-77] L. Berman and J. Hartmanis, "On isomorphisms and density of NP and other complete sets," *SIAM Journal of Computing* **6** (1977), 305-322.

[Ber-78] P. Berman, "Relationship between density and deterministic complexity of NP-complete languages," *Symposium on the Mathematical Foundations of Computer Science,* Springer Verlag Lecture Notes in Computer Science **62** (1978), 63-71.

[Dow-78] M. Dowd, "Isomorphism of complete sets," *Unpublished manuscript,* (1978).

[For-79] S. Fortune, "A note on sparse complete sets," *SIAM Journal of Computing* **8** (1979), 431-433.

[GH-88] K. Ganesan and S. Homer, "Complete problems and strong polynomial reducibilities," *Boston University Technical Report #88-001* (1988), 1-13.

[GHJY-87] J. Goldsmith, L. Hemachandra, D. Joseph, and P. Young, "Near-testable sets," *University of Washington Technical Report #87-11-06* (1987), 1-18.

[GJ-86] J. Goldsmith and D. Joseph, "Three results on the polynomial isomorphisms of complete sets," *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science* (1986), 390-397.

[**GJY-87**] J. Goldsmith, D. Joseph, and P. Young, "Self-reducible, p-selective, near-testable, and p-cheatable sets: the effect of internal structure on the complexity of a set," *Proceedings of the Second Annual Structure in Complexity Conference,* IEEE Computer Society (1987), 50-59. (Also appeared as *University of Washington Technical Report #87-06-02.* Later revised as *University of Wisconsin Technical Report #743* (1988), 1-22.)

[**GJY-88**] J. Goldsmith, D. Joseph, and P. Young, "Using self-reducibilities to characterize polynomial time," *University of Wisconsin Technical Report #749* (1988), 1-20.

[**GS-84**] J. Grollman and A. Selman, "Complexity measures for public key cryptosystems," *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science* (1984), 495-503.

[**GP-86**] L. M. Goldschlager and I. Parberry, "On the construction of parallel computers from various bases of Boolean functions," *Theoretical Computer Science* **43** (1986), 43-58.

[**Har-78**] J. Hartmanis, "On logtape isomorphisms of complete sets," *Theoretical Computer Science* **7** (1978), 75-89.

[**Har-83**] J. Hartmanis, "Generalized Kolmogorov complexity and the structure of feasible computations," *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science* (1983), 439-445.

[**HH-86**] J. Hartmanis and L. Hemachandra, "Complexity classes without machines: on complete languages for UP," *Cornell Technical Report TR-86-746* (April 1986), 1-17.

[**HHem-87**] J. Hartmanis and L. Hemachandra, "One-way functions, robustness, and the non-isomorphism of NP-complete sets," *Proceedings of the Second Annual Structure in Complexity Conference,* IEEE Computer Society (June 1987), 160-174.

[**HIM-78**] J. Hartmanis, N. Immerman, and S. Mahaney, "One-way log-tape reductions," *Proceedings of the 19th Annual IEEE Symposium on the Foundations of Computer Science* (1978), 65-72.

[**Hem-87**] L. Hemachandra "$P^A \neq NT^A$ With Probability 1," *Unpublished manuscript*

(June 1987), 1-11.

[Hom-86] S. Homer "On simple and creative sets in NP," *Theoretical Computer Science* **47** (1986), 169-180.

[HS-88] S. Homer and A. Selman, "Oracles for structural properties: the isomorphism problem and public-key cryptography," *Extended abstract* (1988), 1-9.

[HU-79] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Language, and Computation* (1979), Addison-Wesley, Reading, MA

[Joc-68] C. Jockusch, Jr., "Semirecursive sets and positive reducibility," *Transactions of the AMS* **131** (1968), 420-436.

[JY-85] D. Joseph and P. Young, "Some remarks on witness functions for non-polynomial and non-complete sets in NP," *Theoretical Computer Science* **39** (1985), 225–237.

[Ko-87] K. Ko, "On helping by robust oracle machines," *Proceedings of the Second Annual Structure in Complexity Conference,* IEEE Computer Society (June 1987), 182-190.

[KLD-86] K. Ko, T. Long and D. Du, "A note on one-way functions and polynomial-time isomorphisms," *Theoretical Computer Science* **39** (1986), 225-237. (First reported in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (1986), 295-303.)

[KM-81] K. Ko, and D. Moore, "Completeness, approximation and density," *SIAM Journal of Computing* **10** (1981), 787-796.

[KM-80] D. Kozen and M. Machtey, "On relative diagonals," *IBM Technical Report #8184* (1980), 1-11.

[Kur-83] S. Kurtz, "A relativized failure of the Berman-Hartmanis conjecture," *Unpublished manuscript* (1983).

[Kur-85] S. Kurtz, "Sparse sets in NP-P: relativizations," *SIAM Journal Computing* **14** (1985), 113–119.

[Kur-86] S. Kurtz, *Personal communication* (Nov. 1986).

[**KMR-86**] S. Kurtz, S. Mahaney, and J. Royer, "Collapsing degrees (extended abstract)," *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science* (1986), 380-389.

[**KMR-87**] S. Kurtz, S. Mahaney, and J. Royer, "Progress on collapsing degrees (extended abstract)," *Proceedings of the Second Annual Structure in Complexity Conference,* IEEE Computer Society (June 1987), 126-131.

[**KMR-88**] S. Kurtz, S. Mahaney, and J. Royer, "The isomorphism conjecture fails relative to a random oracle," *University of Chicago Technical Report #88-11* (Aug. 1988), 1-23.

[**LP-81**] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation,* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[**Lon-88**] T. Long, "One-way functions, isomorphisms, and complete sets," abstract, *Abstracts of the American Mathematics Society* **9** (1988), 125.

[**LY-88**] L. Longpre and P. Young, 'Cook is faster than Karp: a study of reducibilities in *NP*," *Proceedings of the Third Annual Structure in Complexity Conference,* IEEE Computer Society (June 1988), 293-302.

[**MY-78**] M. Machtey and P. Young, *An Introduction to the General Theory of Algorithms,* North-Holland, NY, 1978.

[**Mah-81**] S. Mahaney, "On the number of p-isomorphism classes of NP-complete sets," *Proceedings of the 22nd IEEE Symposium on the Foundations of Computer Science* (1981), 271-278.

[**Mah-82**] S. Mahaney, "Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis," *Journal of Computer Systems Science* **25** (1982), 130-143.

[**MY-85**] S. Mahaney and P. Young, "Reductions among polynomial isomorphism types," *Theoretical Computer Science* **39** (1985), 157-165.

[**MP-79**] A. Meyer and M. Paterson, "With what frequency are apparently intractable problems difficult?" *MIT/LCS/TM-126* (1979).

[**MS-72**] A. Meyer and L. Stockmeyer, "The equivalence of regular expressions with squaring requires exponential space," *Proceedings of the 13th ACM Symposium on Switching and Automata Theory* (1972), 125-129.

[**Myh-55**] J. Myhill, "Creative sets," *Z. Math. Logik Grundlagen Math* (1955), 97-108.

[**PZ-82**] C. H. Papadimitriou, and S. T. Zachos, "Two remarks on the power of counting," *Proceedings of the 6th Annual GI Conference on Theoretical Computer Science,* Springer Verlag Lecture Notes in Computer Science **145** (1983), 269-275.

[**Sch-81**] U. Schöning, "A note on complete sets for the polynomial-time hierarchy," *ACM SIGACT News* **13** (1981), 30-34.

[**Sel-79**] A. Selman, "P-selective sets, tally languages, and the behavior of polynomial reducibilities on NP," *Math Systems Theory* **13** (1979), 55-65.

[**TV-87**] L. Torenvliet, and P. van Emde Boas, "Diagonalisation methods in a polynomial setting," *Structure in Complexity Conference,* Springer Verlag Lecture Notes in Computer Science **223** (1986), 330-346.

[**Val-79**] L. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal of Computing* **8** (1979), 410-421.

[**Wat-83**] O. Watanabe, "On pseudo-p-isomorphism of k-creative sets," *Department of Information Sciences, Tokyo Institute of Technology Technical Report #C-57* (1983), 1-8.

[**Wat-86a**] O. Watanabe, "Hardness and Creativity," *Department of Information Sciences, Tokyo Institute of Technology Memo* (2/26/86), 1-4.

[**Wat-86**] O. Watanabe, "On one-one p-equivalence relations," *Theoretical Computer Science* **38** (1986), 157-165.

[**You-66**] P. Young, "Linear orderings under one-one reducibility," *Journal of Symbolic Logic* **31** (1966), 70-85.

[**You-83**] P. Young, "Some structural properties of polynomial reducibilities and sets in NP," *Proceedings of the 15^{th} Annual ACM Symposium on the Theory of Computing* (1983),

392-401.

[You-88] P. Young, "Juris Hartmanis: fundamental contributions to isomorphism problems," *University of Washington Technical Report # 88-06-02* (1988), 1-19. (First appeared in the *Proceedings of the Third Annual Structure in Complexity Conference,* IEEE Computer Society (June 1988), 138-154.