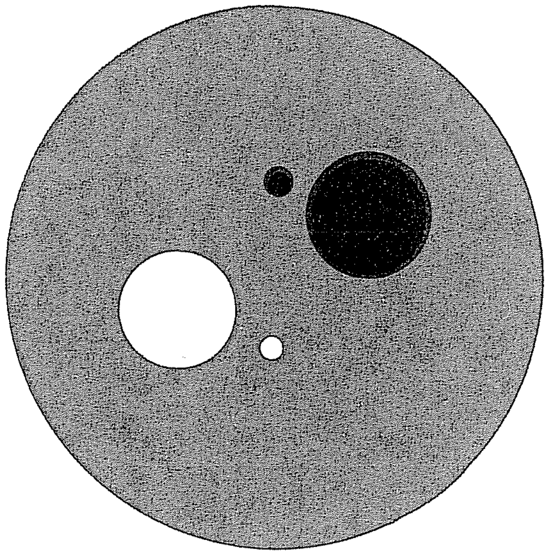


NEAR-TESTABLE SETS

by

Judy Goldsmith
Lane Hemachandra
Deborah Joseph
Paul Young

Computer Sciences Technical Report #797

October 1988

NEAR-TESTABLE SETS

JUDY GOLDSMITH
Dartmouth College

LANE HEMACHANDRA
Columbia University

DEBORAH JOSEPH
University of Wisconsin - Madison

PAUL YOUNG
University of Washington

Authors' addresses:

J. Goldsmith:	Mathematics Department, Dartmouth College, Hanover, New Hampshire 03755.
L. Hemachandra:	Computer Science Department, Columbia University, New York, NY 10027.
D. Joseph:	Computer Sciences and Mathematics Departments, University of Wisconsin 1210 West Dayton St., Madison, WI 53706.
P. Young:	Computer Science Department FR-35, University of Washington, Seattle, WA 98195. (1988-89: Brittingham Visiting Professor Computer Sciences Department, University of Wisconsin 1210 West Dayton St., Madison, WI 53706.)

NEAR-TESTABLE SETS*

JUDY GOLDSMITH
Dartmouth College

LANE HEMACHANDRA
Columbia University

DEBORAH JOSEPH
University of Wisconsin - Madison

PAUL YOUNG
University of Washington

Abstract. In this paper we introduce a new property of sets which we call *near-testability.* A set S is *near-testable* ($S \in NT$) if the membership relation for all immediate neighbors is polynomially computable; i.e., if the function $t(x) = \chi_S(x) + \chi_S(x-1)\ (mod\ 2)$ is polynomially computable. The near-testable sets form a subclass of the class, $\oplus P$ (*parity-P*), introduced by Papadimitriou and Zachos. $\oplus P$ has a complete set $\oplus SAT$ which has recently been shown by Valiant and Vazirani to be hard for NP under randomized polynomial time reductions. We prove that there is a uniform polynomial one-one reduction which takes every set in $\oplus P$ to a near-testable set, and we show that the image of $\oplus SAT$ under this reduction (which we call $NTSAT$) is polynomially isomorphic to $\oplus SAT$. As corollaries we have that $NTSAT$ is complete for both NT and for $\oplus P$, that $NTSAT$ is hard for NP under randomized polynomial time reductions, and that the existence of one-way functions implies the existence of sets that are near-testable but not polynomially decidable. We then ask whether near-testability is preserved under p-isomorphisms. This leads to a generalization, NT^* of NT similar to those introduced by Meyer and Paterson and by Ko for self-reducible sets. With this more general definition, NT^* is shown to be closed under polynomial time isomorphisms while remaining a subclass of $\oplus P$. We conjecture that it is a proper subclass. In fact we show that, relative to a random oracle, the containments $P \subseteq NT \subseteq NT^* \subseteq \oplus P$ are proper with probability one. We also show that, relative to a random oracle, with probability one NT and NT^* are incomparable with both NP and with $coNP$. Finally, we consider the effects that the distribution and density of elements have on the complexity of near-testable sets.

Contents.

Categories and Subject Descriptors: F.1.3 [Theory of Computation]: complexity classes
General Terms: Structural complexity theory
Additional Key Words and Phrases: near-testable, one-way functions, *parity-P*, self-reducible

* This work was supported in part by National Science Foundation grants DCR-8402375 and DCR-8520597, and by a Brittingham Visiting Professorship at the University of Wisconsin-Madison.

Many of the results reported here have also been reported, sometimes in abbreviated form, in [GJY87a] and in [He87]. This technical report also appears as University of Washington Computer Science Department Technical Report 87-11-06.

1. INTRODUCTION—BASIC FACTS ABOUT NEAR-TESTABLE SETS.

Structural complexity theory is often concerned with the interrelationship between sets in a complexity class, (e.g., *Is set S complete for class C?*), and inclusion relationships between classes, (e.g., *Does $P = NP$?*). However, the internal properties of sets (e.g., *Do all NP-complete sets have infinite polynomially decidable subsets?)* are also of interest. We are concerned here not just with the internal structure of sets, but more specifically with the *ordering structures* that exist *within* a set and with the effect that these orderings have on the time and space complexity of the set.

The study of sets and their associated orderings is not new. For example, the classes of *Turing self-reducible* sets, *p-selective* sets, and *p-cheatable* sets have each been widely studied, and the sets in each class have an internal partial ordering which is imposed by the nature of the "self-reducibility" the set exhibits. Turing self-reducible sets have an internal well-founded ordering that ties the membership question for any element of the domain to the membership question for polynomially many "shorter" elements, and the p-selective sets impose on their domain a polynomially testable reflexive and transitive preorder.

This paper is part of a general investigation of how internal orderings affect the time and space complexities of sets. In [GJY87a] we surveyed results concerning various classes of "self-reducible" sets, outlining in a systematic way how ordering structures affect the time and space complexities of these sets. In [GJY87b&c] we reported more fully on our own investigations of the time and space complexity of p-selective, p-cheatable, self-reducible, and word-decreasing query self-reducible sets.

In this paper, following up the preliminary report in [GJY87a], we introduce the class of *near-testable* sets, a class of sets which have a particularly simple internal ordering structure. Like many other self-reducible sets, near-testable sets lie somewhere between P and *Pspace*. We originally became interested in these sets because they are a particularly easily defined extension of P for which it seems difficult to prove that the extension is proper.

Sets are near-testable if, in polynomial time, for any element, x, it is possible to fully relate the membership question for x to the membership question for x's immediate predecessor in the lexicographic ordering.[1] Because of this very simple structure, we hope that it will be possible to analyze near-testable sets in ways that are not possible for more complicated classes.[2]

This intuitive definition of near-testability is easily formalized as follows:

[1] Here, and throughout this paper, we use any standard polynomial one-one encoding between *all* strings over some alphabet and the natural numbers so that we may speak interchangeably about strings and numbers, with no distinctions between the two. (See, e.g., the coding of strings to numbers given in the paragraph on pages 21 and 22 of [MY-78].) In these codings, 0 stands both for the number 0 and for the empty string.

[2] In [Ba87], Balcázar introduced the *word decreasing query self-reducible* sets. Although our original interest in near-testable sets was independent of Balcázar's work, in hindsight his definitions provide a nice motivation for studying near-testable sets. Balcázar said that a set S is *polynomial time word decreasing query (wdq) self-reducible* if there is a polynomial time deterministic oracle Turing machine M such that M^S decides membership in S and for each input x, all queries to S lexicographically precede x. Near-testable sets are easier to analyze than Balcázar's sets, since for near-testable sets the membership question for any given element can be reduced

Definition 1.1. *A set S is* ***near-testable*** *if there is a polynomially computable function that, given $x > 0$, decides whether exactly one of x and $x - 1$ is in S. That is, the function*

$$t(x) = \chi_S(x) + \chi_S(x-1) \pmod 2$$

is polynomially computable. We denote the class of near-testable sets by NT.

Thus, with respect to our ordering on Σ^*, we can fully relate the membership questions for x and $x - 1$, where $x - 1$ denotes time the lexicographic, or equivalently the numeric, predecessor of x. One easily sees that if S is near-testable, then $\overline{S}$ is also, and that all near-testable sets are decidable in linear exponential time[3] and in polynomial space. In addition, it is easily seen that if S is near-testable and if either S or $\overline{S}$ is polynomially sparse, then S is in P. On the other hand, a straightforward slow diagonalization allows one to construct sets that are exponentially (or even just superpolynomially) decidable but not near-testable. We summarize these observations as follows.

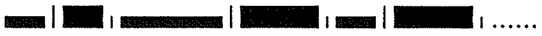
Observation 1.2.
i) $P \subseteq NT = coNT \subseteq Exp \cap Pspace$.
ii) *If $S \in NT$ and if either S or $\overline{S}$ is polynomially sparse, then $S \in P$.*
iii) *There is a set in Exp that is not near-testable.*

An important concept in the study of near-testable sets is the ***boundary*** *of a set. Given a set S, the boundary of S will be defined by*

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

Thus the boundary of a set S consists of the first element of every contiguous sequence of elements of S and of $\overline{S}$ with the exception of the first contiguous sequence.

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



S (exclusive of points in its boundary) is represented by the thicker horizontal lines, $\overline{S}$ (exclusive of points in its boundary) by the thinner horizontal lines, and *boundary*(S) by lines of the form "|" and "ı". Notice that *boundary*(S) = *boundary*$(\overline{S})$.

Observation 1.3. *S is near-testable if and only if boundary$(S) \in P$.*

2. RELATING NEAR-TESTABILITY AND PARITY TESTING.

In [PZ82] Papadimitriou and Zachos introduced a class of sets *parity-P*, denoted by $\oplus P$, which has recently proved useful in the study of randomized polynomial time reducibilities. We will see that the near-testable sets form a subclass of $\oplus P$, and perhaps surprisingly, every set in $\oplus P$ is one-one polynomial time reducible, $\leq_1^P$, to a near-testable set.

in polynomial time to its *immediately* preceding neighbor.

[3] Throughout this paper we will use *Exp* to denote the class of sets recognizable in deterministic time $2^{O(n)}$ and *EXP* to denote the class of sets recognizable in time $2^{n^{O(1)}}$.

Definition 2.1. *A set S is in parity-P if there is a nondeterministic polynomial time Turing machine M such that $x \in S$ if and only if the number of accepting paths in M's computation tree for input x is odd.*[4]

Papadimitriou and Zachos refer to $\oplus P$ as "a more moderate version" of Valiant's class *number-P*, ($\#P$), which is the class of functions, $f(x)$, that count the number of accepting paths produced by a nondeterministic polynomial time Turing machine on input x. Like $\#P$, $\oplus P$ is in *Pspace* and is thought not to be contained in the polynomial time hierarchy. Clearly, if we could compute $\#P$, then $\oplus P$ would be no more difficult. However, there seems to be no obvious way that distinguishing between an odd and even number of accepting paths should help in computing the total number of accepting paths.

We next observe that the near-testable sets form a subclass of $\oplus P$.

Theorem 2.2. *$NT \subseteq \oplus P$.*

Proof. Suppose that S is *near-testable.* Let us assume for the moment that $0 \in \overline{S}$. We will describe a nondeterministic *parity machine*, M, that recognizes S. On input x, M will guess a string lexicographically less than or equal to x, trying to guess an element of *boundary(S)* and verify its guess. Note that $x \in S$ *if and only if* the number of strings in *boundary(S)* that are less than or equal to x is odd *if and only if* there are an odd number of accepting computation paths for $M(x)$. If $0 \in S$, then we simply add one vacuous accepting path to each of M's calculations. ∎

The preceding proof shows the close connection between a set, the boundary of the set, and the *parity* of the boundary. In fact, for any set B if we let

$$parity_B(z) = \begin{cases} 1 & \text{if } |\{w \leq z : w \in B\}| \text{ is odd} \\ 0 & \text{otherwise,} \end{cases}$$

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

$$(*) \qquad 0 \in \overline{S} \Rightarrow S = parity_{boundary(S)} \qquad \text{and} \qquad 0 \in S \Rightarrow \overline{S} = parity_{boundary(S)}.$$

Although it seems unlikely that all sets in $\oplus P$ are near-testable, we next show that these two classes are nevertheless very closely related.

Theorem 2.3. *Every set in $\oplus P$ is $\leq_1^P$-reducible to a near-testable set via a polynomially invertible reduction which has a polynomially decidable range.*[5]

Proof. Let $T \in \oplus P$ and let M be a nondeterministic parity machine that recognizes T. We will give a *uniform* construction which, given any nondeterministic machine M running in polynomial time, p, constructs a near-testable set S such that, if T is the set accepted by M when M is viewed as a parity machine, then $T \leq_1^P S$.

[4] 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 $\oplus P$.

[5] Notice that Theorems 2.2 and 2.3 do *not* guarantee that every *set* in $\oplus P$ is even polynomially many-one equivalent to a near-testable set. We conjecture that such a strong equivalence between $\oplus P$ and *NT* does not hold.

Without loss of generality we may make a number of assumptions about the polynomial bound p and the length of M's computations[6] on inputs of length n:

i) that p is strictly monotonically increasing;
ii) that the function $n + p(n)$ is invertible in polynomial time;
iii) that all computations for an input of length n have length exactly $p(n)$; and
iv) that no computation y is in 0^* or in 1^*.

To construct the set S we first construct a polynomially decidable set B. S is then completely defined by specifying that $0 \in \overline{S}$ and that *boundary*$(S) = B$. The basic idea is to let

$$B = \{\langle x, y\rangle : y \text{ is an accepting computation of } M(x)\},$$

for $\langle\ ,\ \rangle$ a well-behaved pairing function. This will yield only that T is polynomially truth-table reducible to S, so after seeing how the basic idea works we modify the definition of B to obtain $T \leq_1^P S$.

First we describe a suitable pairing function. We will only be concerned about pairing elements (x, y) where y could possibly be a computation of M on input x. Since this is the case only when $|y| = p(|x|)$, we will call such a pair (x, y) *relevant.*

We need a pairing function that: (i) sorts the relevant pairs by their first element, so all $\langle x, y\rangle$'s come before all $\langle x+1, z\rangle$'s; (ii) for relevant pairs, $\langle x, y\rangle$, is computable in time polynomial in the length of the first element; and (iii) has a decoding function for relevant pairs that is polynomially computable in the length of the string coding the pair. It is easily seen that these three conditions are met if we simply take $\langle x, y\rangle$ to be the *concatenation* of the strings x and y.

Using this pairing function, B is obviously in P, and if we define S by requiring that $0 \in \overline{S}$ and *boundary*$(S) = B$, then by Observation 1.3, S is near-testable.

To see that T is truth-table reducible to S, notice first that, by our final assumption about encodings of computation paths, no string of the form $x0^{p(|x|)}$ or $x1^{p(|x|)}$ is in B. Using this and the fact that T is the *parity* set for M's computations, we see that for all x

$$x \in T \iff parity_B(x0^{p(|x|)}) \neq parity_B(x1^{p(|x|)}).$$

But B is simply the boundary of S, so by observation $(*)$, which preceded the statement of this theorem, we have that for $x \neq 0$

$$x \in T \iff S(x0^{p(|x|)}) \neq S(x1^{p(|x|)}),$$

which is a very simple polynomial time two truth-table reduction of T to S.

But we want the stronger result that $T \leq_1^P S$. To obtain this, we modify S by changing the definition of B. Notice that if we simply *doubled* the number of accepting paths then on every computation tree we would *always* have that $parity_B(x0^{p(|x|)}) = parity_B(x1^{p(|x|)}) = 0$. This would make the membership test we have given for T always come out false, but notice that if we could then always find a relevant pair $\langle x, y\rangle$ such that *exactly half* of x's computations always preceded $\langle x, y\rangle$, then we would have that $x \in T$ if and only if $parity_B(\langle x, y\rangle) = 1$.

[6] Here a *computation* is not the sequence of instantaneous descriptions which encode the complete computation path but rather, a computation is the sequence of 0's and 1's that tell us which of the nondeterministic branch points are taken as a nondeterministic Turing machine traverses a particular computation.

This requirement is easily met. We modify S by changing the definition of the boundary B to

$$(**) \qquad B = \{\langle xc, y\rangle : c \in \{0,1\} \text{ and y is an accepting computation of } M(x)\}.$$

This definition effectively duplicates the computation paths, once for $c = 0$ and once for $c = 1$ so that we now always have $parity_B(x00^{p(|x|)}) = parity_B(x11^{p(|x|)}) = 0$. Furthermore half of B's "accepting strings" now lie below $x10^{p(|x|)}$ and half lie above, so we have

$$x \in T \iff parity_B(x10^{p(|x|)}) = 1 \iff x10^{p(|x|)} \in S.$$

This gives the desired polynomial one-to-one reduction of T to S. The set S is again near-testable because B is its boundary and B is in P.

This reduction is polynomially invertible and has a polynomially decidable range because of our assumptions about p. ∎

Corollary 2.4. *$\oplus P \neq P$ if and only if $NT \neq P$.*

Notice that for any one-one, polynomially computable, polynomially honest function f, the set *Range*(f) is in *parity-P*. In fact *Range*(f) is in the subclass of *parity-P* called *unique-P*, which is the class of sets accepted by nondeterministic machines that always have at most one accepting computation. It has been shown that *unique-P* is equal to P if and only if every one-one polynomially computable and polynomially honest function has a polynomially computable inverse, ([GS84]). Polynomially computable functions of this form which do *not* have polynomially computable inverses are called *one-way* functions. Thus we have the following corollary.

Corollary 2.5. *If one-way functions exist, then $NT \neq P$.*

3. COMPLETE PROBLEMS FOR NEAR-TESTABLE SETS.

The key to completeness for Valiant's class $\#P$ is the concept of a solution-preserving reduction. A reduction is said to be *parsimonious* if it preserves not only membership in a set, but also the *number of witnesses* that a given element is in the set. If we define $\#P$ to be the function obtained by regarding Boolean formulas as inputs and obtaining outputs by counting the number of satisfying assignments to the Boolean formulas, then it is easily seen that $\#SAT$ is complete for $\#P$ provided the reductions used in the proof of Cook's Theorem are parsimonious reductions of sets in NP to SAT. If the proof is carefully done, then these reductions are in fact easily seen to be parsimonious, one-to-one and polynomially invertible.[7] Valiant, ([Va79]), has shown that a number of interesting sets, including $\#SAT$, are complete for $\#P$.

Obviously, there is a corresponding definition of $\oplus SAT$, which is the set of Boolean formulas that have an odd number of satisfying assignments. Furthermore, because the reductions used in the proof of Cook's Theorem are parsimonious, the proof which shows that $\#SAT$ is $\leq_1^P$-complete for $\#SAT$ also shows that $\oplus SAT$ is $\leq_1^P$-complete for $\oplus P$.

[7] See e.g., the proof of Cook's theorem in [MY78; Theorem 7.3.9] and also the comments in Exercise 7.3.28 of [MY78].

Since $NT \subseteq \oplus P$, the near-testable set to which $\oplus SAT$ can be reduced using Theorem 2.3 must be $\leq_1^P$-complete not only for the near-testable sets, but also for $\oplus P$. We call this set $NTSAT$.[8] Intuitively, $NTSAT$ is just (an encoding of) the parity set obtained by a careful doubling of the satisfying assignments for Boolean expressions.

Valiant and Vazirani, ([VV86]), have recently shown that $\oplus SAT$ is of interest in the study of randomized reductions since it is NP-hard under randomized reductions.

Definition 3.1. *S is reducible to T by a randomized polynomial time reduction r if there is a polynomial time probabilistic Turing machine computing r, and polynomial $p(n)$ such that for all x, if $x \notin S$ then $r(x) \notin T$, and if $x \in S$ then $Probability[r(x) \in T] \geq p(|x|)^{-1}$.*

Theorem 3.2. *The set $NTSAT$ is*
i) polynomially isomorphic to $\oplus SAT$,
ii) $\leq_1^P$-complete both for NT and for $\oplus P$,
iii) NP-hard under randomized reductions,[9] and
iv) truth-table self-reducible via a simple truth-table of size two.

Proof. (i) By definition, $\oplus SAT \leq_1^P NTSAT$ under the one-one, polynomially *invertible* function f of Theorem 2.3. The set $\oplus SAT$ is a *polynomial cylinder* because it is easily seen to have a one-to-one polynomially computable, polynomially invertible, *padding function*, $pad(x, y)$ such that for all x and y

$$x \in \oplus SAT \iff pad(x, y) \in \oplus SAT.$$

(Simply "pad" Boolean expressions by adding useless variables. If done properly, this won't change the parity of the satisfying assignments for SAT.) Also, $\oplus SAT$ is $\leq_1^P$-complete for $\oplus P$, so $NTSAT \leq_1^P \oplus SAT$. These interreducibilities between $NTSAT$ and $\oplus SAT$ easily imply that the padding function for $\oplus SAT$ induces a corresponding padding function for $NTSAT$, so both are cylinders. It follows (see e.g., Lemma 2.4 of [MY85]) that $\oplus SAT$ and $NTSAT$ are polynomially isomorphic.

(ii) This follows immediately from (i) since $NT \subseteq \oplus P$, and hence $NTSAT \in \oplus P$.

(iii) First observe from the definition of randomized polynomial time reductions that if S is reducible to B via a randomized polynomial time reduction and if $B \leq_1^P C$, then S is reducible to C via a randomized polynomial time reduction. But $\oplus SAT$ is hard for NP under randomized polynomial time reductions, and $\oplus SAT \leq_1^P NTSAT$.

(iv) This will follow from a careful look at the proof of Theorem 2.3.

[8] Technically, $NTSAT$ depends both on the choice of the parity machine, M, which recognizes $\oplus SAT$ and on the polynomial, p, used as an explicit bound on the run time of M. For now, any choice of M and p satisfying Conditions (i) - (iv) in the proof of Theorem 2.3 will do. Later, when we use $NTSAT$ in Theorem 3.2, we will require that the machine M be chosen in a very straightforward and natural way.

[9] Richard Biegel has independently constructed a set which is $\leq_{2-tt}^P$-complete for NT and NP-hard under randomized reductions, (personal communication).

To this end, consider the set $NTSAT$ derived from $\oplus SAT$ using the construction in the proof of Theorem 2.3. Recall that $\oplus SAT$ came from a standard encoding of SAT, and that the machine M which recognized SAT was required not to have any accepting choice sequences in 0^* or in 1^*. Clearly, without loss of generality, we can assume first that the formulas in $\oplus SAT$ are all in conjunctive normal form, and we can further assume by appending appropriate useless conjuncts, (e.g., "$[\neg x_i]$ & $[x_k]$" for new variables x_i and x_k), that the formulas used in defining $\oplus SAT$ have no satisfying assignments in 0^* or in 1^*.

Consider now the standard nondeterministic machine, M, which accepts SAT, and consider formulas with variables $x_1, x_2, \ldots, x_n$. The machine M runs in time which is linear in the length of the *formula*, and in the proof we assumed that the computation, i.e, the list of choices of branch points, always had length equal to the computational time of M. But in fact, if the machine M for SAT is chosen in its most obvious way, then the ***relevant*** branch points for the nondeterministic machine always correspond exactly to the choice of assignments for the variables $x_1, x_2, \ldots, x_n$. Thus for this simple case, in the proof of Theorem 2.3, we can assume that the computations, i.e, the sequence of choices at the branch points, always have length exactly n.

Returning to the proof of part (iv) of the theorem, first consider the boundary set B defined in equation $(**)$ in the proof of Theorem 2.3. For any x there are an *even* number of relevant pairs $\langle xc, y\rangle$ that *are* in the boundary set B. Thus, since we began by putting $0 \in \overline{NTSAT}$, and since B is the boundary for $NTSAT$, we must have that for all x, $\langle x0, 0^{p(|x|)}\rangle \notin NTSAT$. Thus for the *first* computation sequence for x, membership in $NTSAT$ is directly testable.

Now let $F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n)$ be any Boolean formula in conjunctive normal form as described above. For any such formula, we made the *possible* computation choices of the standard nondeterministic Turing machine which recognizes SAT in polynomial time and the *parity* machine which recognizes $\oplus SAT$ correspond *exactly* to the 2^n zero-one valued Boolean vectors of the form $b_1, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots, b_n$, which represent *possible* satisfying assignments for $F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n)$. Thus in the proof of Theorem 2.3 equation $(*)$, we can write the inputs and computation choices which correspond to *relevant* pairs for the set B in the form

$$F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), c, b_1, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots, b_n$$

where the b_i's and the bit c are all just Boolean values. Now for this Boolean formula, F, let

$$F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), c, b_1, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots, b_n$$

correspond to *any* relevant pair.

If $c = 0$ and all $b_j = 0$, then we already know that

$$F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), c, b_1, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots, b_n \quad \notin NTSAT.$$

Assume next that $b_1, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots, b_n \notin 0^*$, and that j is the first position where the bit $b_j = 1$. In this case,

$$F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), c, 0_1, \ldots, 0_{j-1}, 1_j, b_{j+1}, \ldots, b_n \quad \in NTSAT \quad \textit{if and only if}$$

there are an odd number of Boolean assignments less than or equal to $c, 0_1, \ldots, 0_{j-1}, 1_j, b_{j+1}, \ldots, b_n$ which represent successful computations on $F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n)$.

But the assignments less than or equal to $c, 0_1, \ldots, 0_{j-1}, 1_j, b_{j+1}, \ldots, b_n$ are exactly those less than or equal to

$$c, 0_1, \ldots, 0_{j-1}, b_j, b_{j+1}, \ldots, b_n$$

in which the bit b_j is fixed at 1 together with those less than or equal to

$$c, 0_1, \ldots, 0_{j-1}, b_j, 1_{j+1}, \ldots, 1_n$$

in which the bit b_j is fixed at 0.

Therefore, we are able to see that there are an odd number of Boolean assignments less than or equal to $c, 0_1, \ldots, 0_{j-1}, 1_j, b_{j+1}, \ldots, b_n$ which are successful computations on $F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n)$ *if and only if* the parity of the assignments less than or equal to $c, 0_1, \ldots, 0_{j-1}, b_{j+1}, \ldots, b_n$ which are successful computations on $F(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n)$ is *un*equal to the parity of the assignments less than or equal to $c, 0_1, \ldots, 0_{j-1}, 1_{j+1}, \ldots, 1_n$ which are successful computations on $F(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n)$. I.e.,

$$(***) \quad \begin{aligned} &[\![F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), c, 0_1, \ldots, 0_{j-1}, 1_j, b_{j+1}, \ldots, b_n \in NTSAT]\!] \iff \\ &[\![F(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n), c, 0_1, \ldots, 0_{j-1}, 1_{j+1}, \ldots, 1_n \notin NTSAT \Leftrightarrow \\ &\quad F(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n), c, 0_1, \ldots, 0_{j-1}, b_{j+1}, \ldots, b_n \in NTSAT]\!], \end{aligned}$$

which is a standard example of a two-truth-table self-reduction of *NTSAT* to shorter elements of *NTSAT*.

Finally, we consider the case where $c = 1$ and all $b_j = 0$. In this case, we know by our assumptions on the computations that $F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), 1, 0_1, \ldots, 0_{j-1}, 0_j, 0_{j+1}, \ldots, 0_n \notin B$. But this implies that

$$\begin{aligned} &F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), 1, 0_1, \ldots, 0_{j-1}, 0_j, 0_{j+1}, \ldots, 0_n \in NTSAT \iff \\ &F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), 0, 1_1, \ldots, 1_{j-1}, 1_j, 1_{j+1}, \ldots, 1_n \in NTSAT. \end{aligned}$$

This is not a self reduction since both formulas have the same length, but for the latter formula $b_j \neq 0$, so the earlier results may be used to self-reduce

$$F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), 1, 0_1, \ldots, 0_{j-1}, 0_j, 0_{j+1}, \ldots, 0_n$$

by instead using (***) to self-reduce

$$F(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n), 0, 1_1, \ldots, 1_{j-1}, 1_j, 1_{j+1}, \ldots, 1_n. \blacksquare$$

4. POLYNOMIAL ISOMORPHISMS AND GENERALIZATIONS OF NT.

In this section we address the question, "To what extent is the property of being near-testable preserved under polynomial time isomorphisms?" Given the close connection between near-testable sets and their boundaries, one might superficially expect that an isomorphism between the boundaries of two near-testable sets would induce an isomorphism of the corresponding sets, or conversely that an isomorphism between two near-testable sets should induce an isomorphism of the corresponding boundaries. But such expectations fail.

Observation 4.1. *For near-testable sets S and T*

i) boundary(S) $\cong^P$ boundary(T) does not imply that $S \cong^P T$, and

ii) $S \cong^P T$ does not imply that boundary(S) $\cong^P$ boundary(T).

Proof. (i) If we take a very sparse polynomially decidable set, say one whose elements satisfy $x < y$ implies $|x| < 2^{|y|}$, then this single set serves as a boundary for *two* near-testable sets, S and $\overline{S}$. Clearly S and $\overline{S}$ cannot be polynomially isomorphic even though they have the *same* boundary. Thus we cannot expect polynomial isomorphisms of boundaries to induce polynomial isomorphisms of near-testable sets, and no generalization of the notion of near-testability can change this.

(ii) If we ask the converse question, whether a polynomial isomorphism of near-testable sets should imply an isomorphism of their boundaries, the answer is again "no," but in this case the answer is less definitive. For example, let $S = \{0x : x \in \{0,1\}^*\}$ and $T = \{x0 : x \in \{0,1\}^*\}$. In this case S and T are trivially near-testable since they are in P. Note that $S \cong^P T$. But the boundary of T is all of $\{0,1\}^*$, and the boundary of S contains only two elements of each length, so it is polynomially sparse. Therefore *boundary*(S) $\not\cong^P$ *boundary*(T). ∎

Note that in our second example, the boundaries failed to be polynomially isomorphic because of the rigidity of the underlying ordering structure which we insisted on using for our notion of "nearness." If we had been allowed to use *reverse* lexicographic ordering to measure "nearness" in the set T, then the boundaries of S and T in the second example would have been polynomially isomorphic.

One property that distinguishes near-testable, wdq self-reducible sets, and Turing self-reducible sets is that their definition is based on the standard lexicographic ordering of Σ^*. For this reason near-testability and other notions of self-reducibility are quite sensitive to the encodings of the set with respect to this fixed order structure. For example these properities of sets are not necessarily closed under polynomially computable isomorphisms (or even length preserving permutations) of Σ^*. Thus, because of our reliance on the canonical lexicographic ordering, the property of near-testability as we have defined it is probably not preserved under polynomial isomorphisms.

Observation 4.2. *If $NT \neq P$, then near-testability is not preserved by polynomial isomorphisms.*

Proof. Since $NTSAT$ is complete for NT, if $NT \neq P$ then $NTSAT \notin P$. As observed in the proof of Theorem 3.2, $NTSAT$ has a polynomially computable padding function, *pad*. Let $T = \{x0 : x \in NTSAT\}$. It is easy to see that T and $NTSAT$ are polynomially many-one equivalent and that the padding function *pad* for $NTSAT$ can be used to induce a corresponding padding function on T. Thus both $NTSAT$ and T are ***polynomial cylinders***, and, as in the proof of Theorem 3.2, it follows from Lemma 2.4 of [MY85] that $NTSAT$ and T are polynomially isomorphic. Thus $T \notin P$. We now show that T is not near-testable. Since every other string is *not in* T, we see that $x \in T$ if and only if $x \in$ *boundary*(T) $\cap \{\Sigma^* 0\}$. Therefore *boundary*(T) $\in P \Rightarrow T \in P$. Thus *boundary*($T$) $\notin P$ so T is *not* near-testable. ∎

Our last two observations suggest that, if we want the property of being near-testable to be more broadly applicable and to be closed under polynomial isomorphisms, then we should relax our requirement that "nearness" be measured only in terms of the standard lexicographic ordering of Σ^*. Note that similar problems arise for properties like Turing self-reducibility when the definition of the property is strictly tied to the lexicographic ordering of Σ^*. The situation can be remedied by allowing more general orderings of Σ^*; for example Meyer and Paterson ([MP-79]) and Ko ([Ko-83]) have given generalized definitions for underlying ordering structures to use in defining Turing self-reducibility. We will use a similar approach to define a broader class, NT^*, that has many of the properties of NT, yet is more robust.

Basically, we would like to say that a set is in NT^* if there is *some suitable polynomial ordering* relative to which the set is near-testable. To formalize this definition we must describe the properties of a suitable ordering.

Our orderings will be *tree orderings*, or more precisely *forests*, where the minimal elements are the roots of the trees. An element x is *less than* an element y, $x \prec y$, if $x \neq y$ and x and y lie on the same branch of a tree with x closer to the root than y. The roots of the trees must form a polynomially decidable set, which we will call *ROOTS*. We will say that such orderings are *polynomially computable* if $\prec$ is a polynomially testable relation and if, for $x \notin ROOTS$, the function $pred(x) =$ *the unique element* y *such that* $y \prec x$ *and for all* z, $z \prec x \Rightarrow z \preceq y$, is polynomially computable.

In addition, to preserve some of the time and space complexity properties of NT we will require that the ordering be *exponentially well-founded.* For this we require that for some polynomial p, $\max\{|z| : z \prec x\} \leq p(|x|)$. This will imply that $|\{y : y \prec x\}| \leq 2^{p(|x|)}$, and thus that for any x, the path from x back to its root behaves reasonably like the sequence $x,\ x-1,\ \ldots,\ 1$.

Definition 4.3. *A set S is in NT^* if there exists a polynomially computable, exponentially well-founded ordering, $\prec_S$, such that*

i) $ROOTS \cap S$ is decidable in polynomial time, and

ii) for $x \notin ROOTS$, $t(x) = \chi_S(x) + \chi_S(pred_S(x)) \pmod 2$, is a polynomially computable function.

For a set S in NT^* we define $boundary(S) = \{x :$ exactly one of x and $pred_S(x) \in S\}$.

Theorem 4.4.

i) $NT \subseteq NT^* = coNT^* \subseteq \oplus P$.

ii) $NT^* \neq P \Leftrightarrow NT \neq P \Leftrightarrow NT^* \neq NT$.

Proof. (i) Obviously any set in NT is in NT^* since the lexicographic ordering satisfies the properties of Definition 4.3. NT^* is closed under complements since the tree defining a set $S \in NT^*$ can be switched to define $\overline{S}$ simply by noting that if $ROOTS \cap S$ is decidable in polynomial time, so is $ROOTS \cap \overline{S}$. Sets in NT^* are in $\oplus P$ for basically the same reason that sets in NT are in $\oplus P$: we can design a nondeterministic machine that on input x, *guesses* elements z along the branch of a tree from a root to x, checks that $z \preceq_S x$, calculates $y = pred(z)$ if z is not a root, and checks (using the polynomially computable function t) that y is a boundary element along this path or, if z is a root, checks that $z \in S$. Thus the computation tree for this machine on input x has accepting paths for each element z along the branch from x to its root for which $z \in S \iff pred(z) \notin S$, plus one additional accepting path if the root is in S. Thus $x \in S$ if and only if the number of accepting paths is *odd.*

(ii) If $NT^* \neq P$, then $\oplus P \neq P$, so by Corollary 2.4 $NT \neq P$. If $NT \neq P$, the image, T, of $NTSAT$ in Observation 4.2 is an example of a set in $NT^* - NT$. If $NT^* \neq NT$ then trivially $NT^* \neq P$. ∎

From the above result we have that $P \subseteq NT^* \subseteq Pspace$, as well as the following corollary.

Corollary 4.5. *Every set in $\oplus P$ is $\leq_1^P$-reducible to a set in NT^*.*

One motive for extending the definition of NT was to find a superset of NT that was closed under polynomial isomorphisms. The next theorem shows that NT^* accomplishes this goal.

Theorem 4.6. *NT^* is closed under polynomial isomorphisms.*

Proof. Suppose that $S \in NT^*$ and g is a polynomial isomorphism between S and a set T, $g : T \to S$. We will denote by $\prec_S$, $pred_S$, $ROOTS_S$, and t_S the tree ordering, predecessor function, set of *roots* and near-testability function for S. We need to define similar relations, sets and functions for T. We do so as follows:

i) $x \prec_T y$ *if and only if* $g(x) \prec_S g(y)$,

ii) $pred_T(x) = g^{-1} \circ pred_S \circ g(x)$,

iii) $ROOTS_T = \{x : g(x) \in ROOTS_S\}$, and

iv) $t_T(x) = t_S(g(x))$.

Clearly, the ordering $\prec_T$ is polynomially computable, $ROOTS_T$ is a polynomially decidable set of roots for $\prec_T$ and t_T is a near-testability relation for T. Also clearly, $ROOTS_T \cap T$ is polynomially decidable. To see that $\prec_T$ is exponentially well-founded, let p be the polynomial which witnesses that $\prec_S$ is exponentially well-founded and let p_g be a polynomial which bounds the stretching and shrinking done by g. Then

$$\max\{|z| : z \prec_T x\} \;\leq\; p_g(\max\{|z| : g(z) \prec_S g(x)\}) \;\leq\; p_g(p(p_g(|x|))).$$

In addition note that

$$|\{y : y \prec_T x\}| \;=\; |\{y : y \prec g(x)\}| \;\leq\; 2^{p(|g(x)|)} \;\leq\; 2^{p(p_g(|x|))}. \blacksquare$$

In the calculations above, we never used the fact that the inverse, $g^{-1}(x)$, was unique. All we needed to know was that, for each x, the set $g^{-1}(x)$ could be found in polynomial time and that its size was polynomially related to the length of x. Therefore, if g is not one-to-one but is *polynomially many-to-one* and completely polynomially invertible in the sense just described,[10] then for each x we can $\prec_T$ order the elements of $g^{-1}(x)$ by simply using the natural lexicographic ordering. This yields the following observation.

Observation 4.7. *NT^* is closed under polynomial many-one reductions that are onto Σ^* and have completely polynomially computable inverses.*

It is also worth observing that the isomorphisms of Theorem 4.6 preserve the type of tree structures of the underlying orderings. Thus, Theorem 3.2, (i), tells us that the standard complete set, $\oplus SAT$, for $\oplus P$ is polynomially isomorphic to $NTSAT$ we have

Corollary 4.8. *The standard complete set for $\oplus P$, $\oplus SAT$, is in NT^* with an underlying polynomial ordering of type $(N, <)$.*

In the proof of Observation 4.1, we observed that even for generalized definitions we should not expect polynomial isomorphisms of boundaries of near-testable sets to imply the existence of polynomial isomorphisms between the sets. The converse question for NT, whether polynomial isomorphisms of sets in NT

[10] Functions with this property were called ***strongly invertible*** by Allender and Rubinstein ([AR86]). Such functions will be discussed in more detail in Section 6.

imply the existence of polynomial isomorphisms between the boundaries was answered negatively in Observation 4.1 (ii), but only because of the restriction to the use of the canonical ordering $(N, <)$ in the definition of NT. For NT^*, the situation is different. In Theorem 4.6 we saw that if we have a set in NT^* based on the exponentially well-founded tree ordering $\prec_S$, then *any* isomorphism g provides an isomorphism of the *induced* ordering $\prec_T$ (as described in the proof of Theorem 4.6), and it is obvious that the boundaries under $\prec_S$ and $\prec_T$ are polynomially isomorphic. We state this result as

Corollary 4.9. *Given any two polynomially isomorphic sets S and T in NT^*, the boundary of S is isomorphic to that boundary of T which is taken with respect to the tree ordering induced by the isomorphism.*

Corollary 4.8 raises the possibility that $\oplus P = NT^*$. We conjecture that the containment $NT^* \subseteq \oplus P$ is proper. Except for the obvious hypothesis that there exist sparse sets in $\oplus P - P$, we do not have interesting (and nontrivial) conditions which imply that the this is true, although in Section 5 we will see that with respect to a random oracle the containment is proper with probability one.

If the containment is not proper, then $NT^* = \{S : S \leq_m^P NTSAT\}$, and this would say that this generalized notion of near-testability is not just an interesting ordering property that some sets possess, but that it defines a natural complexity class.

To prove the oracle results in Section 5, the following variation of Observation 1.4, part (ii), will be useful.

Observation 4.10. *If $S \in NT^*$ and if either S or $\overline{S}$ is polynomially sparse, then $S \in P$.*

Proof. The proof is the same as for NT. Assume that S is sparse, and let p and q be polynomials such that of the elements of length less than or equal to n, at most $q(n) \in S$, and of the elements preceding x in the ordering $\prec_S$ none is larger than $p(|x|)$. Then for any x, consider the $2 * q(p(|x|)) + 1$ elements in the chain of elements that immediately precede x in the ordering $\prec_S$. These elements may be split into two subsets by enumerating them using the predecessor function and splitting them every time the near-testability relationship tells us that we have crossed a border between S and $\overline{S}$. One of these subsets will be contained in S and the other in $\overline{S}$. Because S is sparse, the larger of these subsets is in $\overline{S}$. This gives a polynomial test for membership in S. Obviously if $\overline{S}$ is sparse, then the same test works, except that the larger of the two groups of elements in the chain of predecessors is in S. ∎

5. RELATIVIZING NT.

Bennett and Gill, ([BG81]), began the study of results that hold for almost every oracle. Although there are examples of results that hold for almost every oracle yet are false in an unrelativized setting, ([Ku82]), probability one results are sometimes considered evidence that a property holds in the "real" world.

In this section we combine relativized versions of our results above with the results of Bennett and Gill to show that, with probability one, NT^A contains computationally difficult sets. We also show that, with probability one, both NT^A and NT^{*A} are incomparable with NP^A and with $coNP^A$.

Theorem 5.1. *Relative to a random oracle A, $P^A \subsetneq NT^A \subsetneq NT^{*A} \subsetneq \oplus P^A$, with probability one.*

Proof. Bennett and Gill, ([BG81], Theorem 3), use the language

$$ODD^A = \{x : \textit{an odd number of strings of length } |x| \textit{ are in } A\}$$

to show that $PP^A \subsetneq PSPACE^A$ with probability one. Since ODD^A is obviously in $\oplus P^A$, their proof shows that ODD^A separates P^A from $\oplus P^A$ with probability one. If we let $T^A =_{def} 0^* \cap ODD^A$, then $ODD^A \equiv_m^P T^A$, so we now have that T^A separates P from $\oplus P^A$ with probability one. But T^A is sparse, and since Observation 4.10 clearly relativizes to an oracle computation, $T^A \in NT^{*A} \iff T^A \in P^A$. Thus $NT^{*A} \subsetneq \oplus P^A$ with probability one.

Now the proof of Theorem 2.3 and hence Corollary 2.4 also relativizes, as does Theorem 4.4. Corollary 2.4 guarantees that any oracle which separates P from $\oplus P$ also separates P from NT. And then Theorem 4.4, part (ii) guarantees that any oracle which separates P from NT also separates NT from NT^*. This establishes for any oracle A for which ODD^A separates P^A and $\oplus P^A$,

$$P^A \underset{\neq}{\subset} NT^A \underset{\neq}{\subset} NT^{*A} \underset{\neq}{\subset} \oplus P^A.$$

Since the collection of oracles A for which ODD^A separates P and $\oplus P$ has measure one, this establishes the theorem. ▮

These results can also be used to separate NT^A from PP^A and from $NP^A \cup coNP^A$. (PP is the class defined of languages defined by Gill that are recognized by nondeterministic Turing machines with more than half of their paths *accepting*.) In fact, the preceding proof directly shows

Corollary 5.2. *If A is a randomly chosen oracle, then $\oplus P^A - PP^A \neq \emptyset$ with probability one.*

Corollary 5.3. *If A is a randomly chosen oracle, then*
i) $NT^A - PP^A \neq \emptyset$ with probability one, and
ii) $NT^A - (NP^A \cup coNP^A) \neq \emptyset$ with probability one.

Proof. (i) Suppose that A is an oracle relative to which $\oplus P^A - PP^A \neq \emptyset$, and let $L^A \in \oplus P^A - PP^A$. First, notice that if $L^A \in \oplus P^A$ and if f is the reduction described in the proof of Theorem 2.3, then $S^A = f(L^A) \in NT^A$. (This follows directly from the fact that *boundary*$(S^A) \in P^A$.) Second, notice that if $L^A \in PP^A$ and $S^A \leq_m^P L^A$, then S^A is in PP^A. Putting these two observations together, (i) is a direct consequence of Corollary 5.2.

(ii) Notice that for every oracle A, $NP^A \cup coNP^A \subseteq PP^A$. Therefore (i) implies that $NT^A - (NP^A \cup coNP^A) \neq \emptyset$ with probability one. ▮

The same techniques can be used to show that relative to a random oracle, A, with probability one there are sets in NP^A and in $coNP^A$ which are not in NT^{*A}.

Theorem 5.4. *Relative to a random oracle A,*

$$(NP^A - NT^{*A}) \neq \emptyset \quad \text{and} \quad (coNP^A - NT^{*A}) \neq \emptyset$$

with probability one.

Proof. Given a set A, define a function $\xi(x)$ = the string of $0's$ and $1's$ of length $|x|$ such that the k^{th} bit is 1 if and only if $x10^{k-1} \in A$. Bennett and Gill show that, with probability one, the language $RANGE3^A =_{def} \{x : \exists y[\xi(y) = xxx]\}$ is both *P-immune* and *coP-immune;* i.e., neither $RANGE3^A$ nor $\overline{RANGE3^A}$ contains an infinite polynomially decidable set. Since $RANGE3^A$ is obviously in NP^A, this shows that, with probability one, $RANGE3^A$ separates NP^A from P^A. ([BG-81]; Theorem 6.)

But since $RANGE3^A$ is both P-immune and coP-immune, 0^* must intersect both $RANGE3^A$ and $\overline{RANGE3^A}$ infinitely often. Thus, as pointed out by Bennett and Gill, $T^A =_{def} RANGE3^A \cap 0^*$ must be a polynomially sparse set which is in $NP^A - P^A$ with probability one. Since it is polynomially sparse, just as in the proof of Theorem 5.1, T^A must therefore be in $NP^A - NT^{*A}$ with probability one.

Since NT^{*A} is closed under complements, we may separate $coNP^A$ from NT^{*A} with probability one by using the complement of T^A. ∎

6. SOME ADDITIONAL FACTS ABOUT NEAR-TESTABLE SETS.

In this final section we present additional results concerning near-testable sets. In Section 1 we noted that any polynomially sparse set that is near-testable is polynomially decidable. Here we return to this theme, first discussing the effects that the distribution and density of elements have on the complexity of a near-testable set. Next, having observed in Corollary 2.4 that the existence of one-way functions implies that there are near-testable sets that are not polynomially decidable, we prove a partial converse to this result. And finally, we briefly relate P-selective and near-testable sets.

We begin this section with the observation that because *near-testability* is sensitive to the distribution of elements in a set, and to density, certain sets are very unlikely to be near-testable. For instance, since the primes (with the exception of 2) are distributed only throughout the *odd* integers, an odd number greater than 3 is prime if and only if it is in $boundary(\{primes\})$. Thus,

Observation 6.1. *If* $\{q : q \text{ is prime}\}$ *is near-testable, then primality testing can be done in polynomial time.*

Since the set of primes is known to be in ZPP, this tells us that not all sets in ZPP are near-testable unless primality testing is in P. It should be observed that we can polynomially encode any set S into the odd numbers, and therefore the encoded set S' is near-testable if and only if $S \in P$. This again points out the how sensitive NT (although not NT^*) is to the underlying order structure.

One can generalize the above example by replacing the set of odd numbers by any polynomially recognizable set that is sufficiently dense.

Definition 6.2. *Let* $\prec$ *be any polynomial computable, exponentially well-founded ordering. We say that a set* D *is uniformly dense with respect to* $\prec$ *if there is a polynomial* $p(n)$ *such that for any string* x*, there is an element of* D *within* $p(|x|)$ *'steps' preceding* x *in the ordering* $\prec$.[11]

Observation 6.3. *Suppose that* $S \in NT^*$. *If there is a uniformly dense set* D *such that* $D \in P$ *and* $D \cap S \in P$*, then* $S \in P$.

Proof. Suppose the polynomial $p(n)$ bounds the number of steps from any string of length n to the nearest element of D. Suppose also that both D and $D \cap S$ can each be polynomially decided. To decide $x \in S$?, we enumerate the $p(|x|)$ strings immediately preceding x, and we then run the decision procedure for D on each of these strings. Once we find $y \in D$, we quickly decide whether $y \in D \cap S$. We then use the near-testability algorithm to decide membership for all of the strings from y to x, including x. ∎

[11] An inverse of this notion, that of a set being *uniformly sparse* in the standard lexicographic ordering is used in [HIS85] and also (in a very strong form which we use in Theorem 6.7) in [GJY87c].

The preceding observations show that if certain uniformly dense sets are in NT^*, then they are also in P. The next observation shows that if the boundary of an NT^* set is uniformly dense then the set and its complement are many-one inter-reducible.

Observation 6.4. *Suppose that $S \in NT^*$ and that boundary(S) is uniformly dense w.r.t. $\prec_S$, then $S \leq_m^P \overline{S}$. (Thus, if S is in NP or in coNP, then $S \in NP \cap coNP$.)*

Proof. Since *boundary*$(S) \in P$ and is uniformly dense, a polynomial many-one reduction from S to $\overline{S}$ can be defined as follows. Given x, we can find (in polynomial time) the first $y \preceq_S x$ such that $y \in$ *boundary*(S). If $y \in ROOTS$ then $x \in S$ if and only if $y \in S$, and for $y \in ROOTS$ this is polynomially decidable. (Thus as long as $\overline{S}$ is nonempty we can pick an appropriate element to reduce x to.) Else $x \in S$ if and only if *pred*$(y) \in \overline{S}$. ∎

The results in Section 5 show that it is not likely that all sets in NP, $coNP$, or PP are near-testable or even in NT^*. The question of whether there are *any* near-testable sets that are not in P was addressed in Section 2 where we showed that $NT \neq P$ if and only if $\oplus P \neq P$. As pointed out there, the existence of one-way functions implies that *unique-P*, and hence $\oplus P$, is not equal to P. Thus if one-way functions exist $NT \neq P$.

A natural question is whether we actually need the existence of one-way functions in order to prove that there are sets in NT or in $\oplus P$ that are not in P. One result which suggests that one-way functions are not needed is the construction in [HH87] of an oracle relative to which one-way functions do not exist but $P^A \neq \oplus P^A$. Our next result, however, gives a partial answer pointing in the other direction. Instead of asking about polynomially invertible one-one functions, we instead ask about *strongly invertible* many-one functions. Recall that in the proof of Observation 4.7 we used inverses of polynomially-many-to-one, polynomially computable, polynomially honest functions. If f is such a function, let $f^{-1}(x) = \{y : f(y) = x\}$. Notice that if f is polynomially many-to-one, it is conceivable that, given x, we can find $\{y : f(y) = x\}$ in time polynomial in $|x|$. Functions for which we can always complete the listing of all of $\{y : f(y) = x\}$ in time polynomial in $|x|$ are called (many-one) *strongly invertible*. Functions for which we cannot always completely list $\{y : f(y) = x\}$ in polynomial time are said to be *one-way*.

Many-one, one-way functions have been studied in other contexts, and are discussed extensively in [AR86]. For example, for strong invertibility on 0^*, Allender and Rubinstein prove that the following are equivalent:

i) There is an honest polynomially many-to-one function computable in polynomial time that is not strongly invertible on 0^*.
ii) There is a polynomially sparse set in $FewP - P$.
iii) $EXP^{sparse(FewP)} \neq EXP$.
iv) There is a polynomially sparse set in P that is not P-printable.
v) There is a polynomially sparse set in *Dlogspace* that is not P-printable.

In this context, we have the following partial converse to Corollary 2.6. (Recall that any set which is polynomially sparse has a polynomially sparse boundary, *but the converse is not true.*)

Theorem 6.5. *If there is a set $S \in NT^* - P$ with a polynomially sparse boundary, then there is a polynomially-many-to-one, polynomially computable, polynomially honest function that is not strongly invertible on 0^*.*

Proof. Let $B = boundary(S)$. Let $q(n)$ be a polynomial bound on the census function for B, and let $\prec$ be the underlying tree ordering used in witnessing that $S \in NT^*$. We define

$$f(x) = \begin{cases} 0^{|x|} & \text{if } x \in B; \\ x & \text{otherwise.} \end{cases}$$

Since $S \in NT^*$, $B \in P$, so $f(x)$ is polynomially computable. Since $|f(x)| = |x|$, f is honest. Notice that $f^{-1}(0^n)$ has at most $q(n)$ elements, all of length n. Suppose f^{-1} *were* polynomially computable on 0^*. Given x, in polynomial time we could then compute all of $f^{-1}(0)$, $f^{-1}(0^2)$, $\ldots f^{-1}(0^{|x|})$, throwing away any z's found in some $f^{-1}(0^n)$ $(n \le |x|)$ for which $z \in 0^*$ but $z \notin B$, and throwing away any z for which $z \not\preceq x$. Thus, in time polynomial in $|x|$, we can find all z's $\preceq x$ which lie on the boundary of S. Clearly, this would enable us to determine membership of S in polynomial time. Thus, if $S \notin P$, f must be a one-way function.

(Notice that if S has at most one element of any given length, then f is a *one-to-one* one-way function.) ∎

The proof of the previous theorem requires not only a set in $NT^* - P$, but one that has a sparse boundary. Given a one-way function, Corollary 2.4 and Theorem 2.3 let us construct sets in $NT - P$, however these sets do *not* have sparse boundaries, and we do not know interesting conditions which imply the existence of sets in $NT - P$ with sparse boundaries.[12] We can construct, although we will not do so here, an oracle A relative to which there exist sets in $NT^A - P^A$ which do have boundaries with at most one element of any given length.

In closing, we give one more method which may construct sets that are *near-testable* but are not polynomially decidable. This construction not only gives additional evidence that $NT \neq P$, but it shows that the combination of near-testability and P-selectivity is unlikely to guarantee that sets are decidable in polynomial time. This is in contrast to the case for p-cheatability, since it can be shown that near-testable sets that are (2^k *for* k) p-cheatable are all decidable in polynomial time, ([GJY87c]).[13]

To do our final construction, we need to assume the existence of *very* sparse sets in P that are not *P-printable.*

Definition 6.6. *A set S is uniformly log^*-sparse if for all x and y in S, $x < y$ implies $2^{|x|} < |y|$.*

The existence of uniformly log^*-sparse sets is discussed in [GJY87c].

Theorem 6.7. *If there is a uniformly log^*-sparse set in P that is not P-printable, then there is a P-selective near-testable set that is not polynomially decidable.*

[12] Recently Cai and Hemachandra have shown that every set in $FewP$ is in $\oplus P$, ([CH88]). One might hope to use this result together with the reduction from sets in $\oplus P$ to sets in NT given in Theorem 2.3 to strengthen Theorem 6.5. Unfortunately this does not seem possible because Cai and Hemachandra's result increases the number of accepting paths so that it is no longer polynomially bounded.

[13] For information about P-selective sets, see [S79], [S82a], [S82b], [Ko83], [GJY87a], and [GJY87c]. Amir, Beigel and Gasarch have independently shown this last result for the special case of *(2 for 1)* p-cheatability, (personal communication).

Proof. Let S be a uniformly *log**-sparse set in P that is not *P-printable*. Define a rapidly growing function f by $f(0) = 2$ and $f(n+1) = 2^{f(n)}$. We will let I_n be the interval of strings of length $f(n)$ up to length $f(n+1)$. From the definition of uniformly *log**-sparse sets, we know that $|S \cap I_n| \leq 1$ for each n.

$$\text{Let} \quad S_0 = S \bigcap \cup_n\{I_{3n}\}, \quad S_1 = S \bigcap \cup_n\{I_{3n+1}\}, \quad \text{and} \quad S_2 = S \bigcap \cup_n\{I_{3n+2}\}.$$

Then $S_i \in P$ for each i, and for at least one i, S_i is not P-printable. Without loss of generality, we will assume that S_0 is not P-printable.

We then define the desired set T as follows:

$$T \;=\; \{x : \; x \in I_{3n} \cup I_{3n+1} \text{ and } \exists y \in I_{3n} \cap S_0, \; y \leq x\} \;\bigcup\; \{I_{3n+2} - \{f(3n+3) - 1\}\}.$$

In other words, all but the last element of I_{3n+2} is always in T, and the last element of I_{3n+2} is always a boundary point. If there is an element $y \in I_{3n} \cap S_0$, then this y serves as a "breakpoint," dividing $I_{3n} \cup I_{3n+1}$ into a lefthand subinterval in $\overline{T}$ and a righthand subinterval in T. If there is no such element, then both I_{3n} and $I_{3n+1} \subset \overline{T}$.

The boundary of T thus consists of the rightmost element of I_{3n+2}, the element of $I_{3n} \cap S_0$ if there is one, plus the lefthand endpoint of I_{3n+2} when $I_{3n} \cap S_0 = \emptyset$. By the definition of the I_k's, if y is the lefthand endpoint of I_{3n+2}, then in time polynomial in $|y|$, we can test whether there is an element of S_0 in I_{3n}. Thus, $boundary(T) \in P$, so T is near-testable.

Assuming $x \leq y$, the following function is a *P-selection* function for T:

$$s(x,y) = \begin{cases} x & \text{if } x \in I_{3m+2} \text{ for some } m, \\ y & \text{if } x, y \in I_{3m} \cup I_{3m+1} \cup I_{3m+2} \text{ for some } m, \\ x & \text{if } x \in I_{3m} \cup I_{3m+1},\, y \in I_{3n} \cup I_{3n+1} \cup I_{3n+2} \text{ for some } n > m \text{ and } x \in T, \\ y & \text{if } x \in I_{3m} \cup I_{3m+1},\, y \in I_{3n} \cup I_{3n+1} \cup I_{3n+2} \text{ for some } n > m \text{ and } x \notin T. \end{cases}$$

Note that in the last two cases we can test $x \in T$ in time polynomial in $|y|$.

Finally, suppose that $T \in P$. Then the following algorithm *P-prints* S_0: for each k, if $0^k \in T$ and $0^{k+1} \notin T$, then there is exactly one boundary point for T of length k. The assumption that $T \in P$ thus enables us to find this boundary point in polynomial time by using binary search. By definition of T, any boundary point for T has one of three forms, and two of these, the elements of the form $f(3n+3) - 1$ and of the form $f(3n+2)$ simply aren't in S_0. Any other boundary point is in S_0. Since *all* elements of S_0 are boundary points of T, this gives a polynomial algorithm for printing S_0, a contradiction.

Thus, T is a *P-selective, near-testable* set which is not in P. ∎

7. CONCLUSIONS.

In this paper we have introduced the class of *near-testable* sets. This class is of interest because it is a class of sets which have a particularly simple internal ordering structure and, like other self-reducible sets, near-testable sets lie somewhere between P and *Pspace*. We originally became interested in these sets because they are a particularly easily defined extension of P for which it seems difficult to prove that the extension is proper. Our results show that there are near-testable sets that are not polynomially decidable if and only if there are sets in $\oplus P$ that are not polynomially decidable. Nevertheless, we leave open the major problem of obtaining nontrivial upper and lower bounds on the computational complexity of near-testable sets.

ACKNOWLEDGEMENTS.

The authors would like to thank Eric Allender, Klaus Ambos-Spies, José Balcázar, and Alan Selman for useful discussions.

8. BIBLIOGRAPHY.

[AR86] E. Allender and R. Rubinstein, "P-printable sets," submitted for publication, preliminary version appeared as 'The complexity of sparse sets in P,' *Proc First Annual Structure in Complexity Theory Conference, Springer-Verlag Lecture Notes in Comput Sc,* **223** (1986), 1-11.

[BGS75] T. Baker, J. Gill, and R. Solovay, "Relativizations of the $P =? NP$ question," *SIAM J Comput,* **4** (1975), 431-442.

[Ba87] J. Balcázar, "Self-Reducibility," *Proc Symposium on the Theory of Automata and Computing, Springer-Verlag Lecture Notes in Comput Sc,* **247** (1987), 136-147.

[BG81] C. Bennett and J. Gill, "Relative to a random oracle A, $P^A \neq NP^A$ with probability one," *SIAM J Comput,* **10** (1981), 96-113.

[CH87] J. Cai and L. Hemachandra, "On the power of parity polynomial time," *Columbia University Technical Report,* **CUCS-274-87** (1987), 1-11.

[GJY87a] 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, preliminary abstract," *Proc Second Annual Structure in Complexity Theory Conference* (1987), 50-59. Also *in more complete form* as *University of Washington Technical Report,* **# 87-06-02,** and as *University of Wisconsin Technical Report,* **# 743**, 1987, 1-22.

[GJY87b] J. Goldsmith, D. Joseph, and P. Young, "A note on bi-immunity and p-closeness of p-cheatable sets in P/poly," *University of Washington Technical Report,* **# 87-11-05,** and *University of Wisconsin Technical Report,* **# 741,** (1987), 1-13.

[GJY87c] J. Goldsmith, D. Joseph, and P. Young, "Using self-reducibilities to characterize polynomial time," *University of Washington Technical Report,* **# 87-11-11,** and *University of Wisconsin Technical Report,* **# 749,** (1987), 1-20.

[GS84] J. Grollman and A. Selman, "Complexity measures for public key cryptosystems," *Proc 25th IEEE Symposium on Foundations of Computer Science,* (1984), 495-503.

[GP86] L. Goldschlager and I. Parberry, "On the construction of parallel computers from various bases of boolean functions," *Theoretical Computer Science,* **43** (1986), 43-58.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson, "Sparse sets in $NP - P$; $EXPTIME$ versus $NEXPTIME$," *Information and Control,* **65** (1985), 159-181.

[HH87] J. Hartmanis and L. Hemachandra, "One-way functions, robustness, and the non-isomorphism of *NP*-complete sets," *Proc Second Annual Structure in Complexity Theory Conference, IEEE Computer Society,* (1987), 160-173.

[He87] L. Hemachandra, "$P^A \neq NT^A$ with probability one," *Preprint,* (1987).

[Ko83] K. Ko, "On self-reducibility and weak P-selectivity," *Journal of Computer and System Sciences,* **26** (1983), 209-221.

[Ku82] S. Kurtz, "On the random oracle hypotheses," *14th ACM Symposium on the Theory of Computing,* (1982), 224-230.

[MY78] M. Machtey and P. Young, *An Introduction to the General Theory of Algorithms,* Elsevier, New York (1978), 1-264.

[MY85] S. Mahaney and P. Young, "Reductions among polynomial isomorphism types," *Theoretical Computer Science,* **39** (1985), 207-224.

[MP79] A. Meyer and M. Paterson, "With what frequency are apparently intractable problems difficult?" *MIT/LCS/***TM-126** (1979).

[PZ82] C. Papadimitriou, and S. Zachos, "Two Remarks on the Power of Counting," *Springer-Verlag Lecture Notes in Computer Science,* **145** (1983) 269-275.

[Ra82] C. Rackoff, "Relativized questions involving probabilistic algorithms," *J ACM,* **29** (1982), 261-268.

[S79] A. Selman, "P-selective sets, tally languages, and the behavior of polynomial reducibilities on NP," *Math Systems Theory,* **13** (1979), 55-65.

[S82a] A. Selman, "Analogues of semi-recursive sets and effective reducibilities to the study of NP complexity," *Information and Control* **52** (1982), 36-51.

[S82b] A. Selman, "Reductions on NP and P-selective sets," *Theoretical Computer Science,* **19** (1982), 287-304.

[Va79] L. Valiant, "The complexity of enumeration and reliability problems," *SIAM J Comput,* **8** (1979), 410-421.

[VV86] L. Valiant and V. Vazirani, "NP is as easy as detecting unique solutions," *Theoretical Computer Science,* **47** (1986), 85-93.