SELF-REDUCIBLE, P-SELECTIVE, NEAR-TESTABLE, AND P-CHEATABLE SETS: THE EFFECT OF INTERNAL STRUCTURE ON THE COMPLEXITY OF A SET

by

Judy Goldsmith

Deborah Joseph

Paul Young

Computer Sciences Technical Report #743

January 1988

Self-Reducible, P-Selective, Near-Testable, and P-Cheatable Sets: The Effect of Internal Structure on the Complexity of a Set

Judy Goldsmith and Deborah Joseph
Computer Sciences and Mathematics Departments
University of Wisconsin - Madison

Paul Young
Computer Science Department
University of Washington

June 1987

Revised July 1987

Self-Reducible, P-Selective, Near-Testable, and P-Cheatable Sets: The Effect of Internal Structure on the Complexity of a Set*

Judy Goldsmith and *Deborah Joseph*

Computer Sciences and Mathematics Departments
University of Wisconsin - Madison
1210 West Dayton Street
Madison, WI 53706

Paul Young

Computer Science Department
University of Washington, FR-35
Seattle, WA 98195

Abstract

In this paper we discuss the relationship between certain structural properties of a set and the set's computational complexity. We study four classes of sets for which the membership question for one element of the domain can be related to the membership question of other *smaller* (with respect to *some* ordering) elements: self-reducible sets, p-selective sets, near-testable sets and p-cheatable sets. The purpose of the paper is to suggest that a continuing systematic study of the relationship between this type of internal structure and the computational complexity of a set is in order. Although some new results are presented, much of the paper is an attempt to review known results and suggest unifying concepts.

1. Introduction

Structural complexity theory is often concerned with the inter-relationship between sets in a complexity class, (e.g. *Is set A complete for class C?*), and inclusion relationships between classes, (e.g. *Does* $P = NP$*?*). Additionally, structural complexity theorists have studied the internal properties of sets (e.g. *Do all* NP*-complete sets have polynomially decidable subsets?*). It is this later type of internal structure that we will be concerned with. In this paper we advocate a systematic study of the ordering structure that exists within a set and the effect that that structure has on the complexity of the set. This line of research is not new, as considerable research has been done on *self-reducible* sets and *p-selective* sets, both of which have the type of internal structure that we are interested in. Self-reducible sets have an internal structure that ties the membership question for any element of the domain to the membership question for polynomially many smaller elements. The most famous self-reducible set is SAT. Balcázar, Berman, Book, Fortune, Ko, Mahaney, Meyer, Paterson, Schöning, Schnorr, Selman, Trakhtenbrot and many others have studied properties of self-reducible sets. P-selective sets are sets for which there is a polynomially computable function that

* This work was done with the partial support of NSF Grant DCR-8402375 and an NSF Presidential Young Investigator's Grant with matching support from AT&T Bell Labs. This is an expanded version of the abstract presented under this same title in the Proceedings of the Second Annual Structures in Complexity Theory Conference in 1987 ([GJY87]) and is, with minor corrections and addenda, identical to the University of Washington Technical Report distributed under this title at that conference.

selects from any pair of elements of the domain, an element that is in the set (if either element is). This turns out to induce a (dense) linear ordering structure on the set. Selman and Ko have studied these sets extensively.

Although quite a bit of work has been done on self-reducible and p-selective sets, only a few results have explicitly related their internal ordering structure to their complexity, and little has been done to discuss internal structure in a unified way. The purpose of this paper is to motivate research in this direction.

We begin by surveying work that has been done on self-reducible and p-selective sets. Next we introduce a new class of sets, the *near-testable* sets, and discuss their internal structure and complexity. Finally, we discuss recent work by Amir, Beigel, Gasarch, Gill, Hay and Owings on p-cheatable sets and give some new results about sets with this type of structure.

2. Self-Reducible Sets

The first class of sets that we will consider is the class of self-reducible sets. These sets will turn out to have well-founded partial orderings associated with their domains. We will see, however, that an apparently minor variation in the definition will yield sets (self-1-helpers) whose structure has not yet been characterized. A set A is *self-reducible* if there is a polynomial time oracle machine M such that (i) M^A is a recognizer for A, and (ii) on inputs of length n all of M's queries have length less than n. Obviously, every set in P is self-reducible. It is also easy to see that every self-reducible set is in $PSPACE$. Thus,

Observation. *$P \subseteq$ Self-Reducible $\subseteq PSPACE$.*

Probably the best known self-reducible set is SAT. The fact that SAT is self-reducible, and hence that there is a self-reducible NP-complete set, was heavily exploited by Berman [B78], Fortune [F79], and Mahaney [M82] to show that unless $P = NP$, there are no sparse or co-sparse NP-complete sets.

Balcázar, Book and Schöning have shown, with the following theorem, that it is highly unlikely that any self-reducible sets in the polynomial time hierarchy can be reduced to sparse sets.

Theorem [BBS86]. *Suppose that A is self-reducible and that there is a $k \geq 0$ such that $A \in \Sigma_k^P/poly$. Then $\Sigma_2^P(A) \subseteq \Sigma_{k+2}^P$.*

Thus, it is not likely that any of the natural self-reducible sets have polynomial size circuit complexity, particularly if they are complete.

Now, for the purpose of considering the internal structure imposed by self-reducibility and in order to get a definition that is polynomially invariant, it is useful to look at a slightly different definition. Meyer, Paterson [MP79] and Ko [K83] have given more general definitions of self-reducibility.

Definition [K83]. *A partial ordering $\prec$ on Σ^* is polynomially well-founded and length related (for short: polynomially related) if there is a polynomial p such that*

(i) $x \prec y$? is polynomially testable,

(ii) $x \prec y$ implies that $|x| \leq p(|y|)$, and

(iii) the length of a $\prec$-descending chain is shorter than p of the length of its maximal element.

Definition [K83]. *A set $A \subseteq \Sigma^*$ is tt-self-reducible if there is a polynomially related ordering $\prec$ on Σ^*, a polynomially computable function f and a polynomial p such that, for all sufficiently large $x \in \Sigma^*$*

(i) $f(x)$ is a tt-condition $\langle \alpha, \langle x_1, ..., x_k \rangle \rangle$ where α is a k-ary boolean formula that can be evaluated in time $p(|x|)$ and $x_i \prec x$ for all $i \leq k$, and

(ii) $x \in A$ *iff* $\alpha(\chi_A(x_1), ..., \chi_A(x_k)) = 1$, *where* χ_A *is the characteristic function for* A.

If the truth-table condition α is a simple disjunction or conjunction, then A is said to be *disjunctive (or conjunctive) self-reducible.* If, for any assignment of truth values to the variables, changing the truth value of any variable from "false" to "true" can never change the resulting evaluation of the truth-table from "true" to "false,", then A is said to be *positive truth-table self-reducible.* SAT is obviously disjunctive self-reducible, and it has been conjectured that all NP-complete sets are disjunctive self-reducible. This, however, remains an open question, even assuming $P \neq NP$.

A set is *Turing self-reducible* if there is a polynomial time oracle machine M such that (i) M^A is a recognizer for A, and (ii) on input x all of M's queries are to elements that precede x in the $\prec$-ordering. Obviously, any set that is tt-self-reducible is Turing self-reducible.

These definitions make explicit the idea that for self-reducible sets there is polynomial ordering of Σ^* such that the membership question for an element is fully answered by considering the membership questions for polynomially many preceding elements. Clauses (ii) and (iii) in the definition ensure that the partial ordering is *well-founded. The purpose of this paper is to raise the question of what types of sets have ordering properties of this sort and what effect these orderings and membership relations have on the complexity of the set.* We will keep returning to these questions as we consider additional classes of sets that have similar ordering properties.

Before we conclude this section we will consider one other type of set that is closely related to the self-reducible sets.

Robustness and Helping

The notions of *robustness* and *helping* introduced by Schöning and studied by Balcázar, Ko, Hartmanis and Hemachandra are related to the idea of Turing or truth-table self-reducibility. Schöning defined an oracle Turing machine M to be *robust* if for every oracle A the language accepted by M^A is the same. Thus, changing the oracle can influence the resources used by the machine on an input (i.e. the computation), but it cannot change the acceptance or rejection of the input. With this definition Schöning showed the following.

Theorem [S85].
(i) $L \in NP \cap coNP$ *if and only if there is a robust, deterministic oracle Turing machine* M *and an oracle* A *such that* M^A *decides membership in* L *and* M^A *runs in polynomial time.*
(ii) $L \in NP$ *if and only if there is a robust, nondeterministic oracle Turing machine* M *and an oracle* A *such that* M^A *decides membership in* L *and* M^A *runs in polynomial time.*

Now, returning to our claim that the notion of robustness is related to self-reducibility, consider the definition given by Ko of a *self-1-helper.*

Definition [K87]. *A is a self-1-helper if there is a robust, deterministic oracle Turing machine* M *accepting* A *such that for all* $x \in A$, $M^A(x)$ *halts in polynomial time.*

Ko shows that any set that is disjunctive self-reducible, for instance SAT, is a self-1-helper.

Theorem [K87]. *If A is disjunctive self-reducible, then A is a self-1-helper.*

The proof of this theorem essentially rests on the fact that a robust, deterministic oracle Turing machine can be constructed to aid in quickly searching the self-reducibility tree for any disjunctive self-reducible set.

Recall that we mentioned above that it is not known whether all NP-complete sets are disjunctive self-reducible. Ko has observed that if every polynomial Turing degree in NP contains a disjunctive self-reducible set, then there are no log^*-sparse sets in $NP - P$, and hence $EXP \not\subseteq NP$.

One of the major open problems left by Ko's work is: What is the relationship between self-helping and self-reducibility? For instance, is the converse of the above theorem true? If A is a self-1-helper, then it appears that A is close to being Turing self-reducible: the machine M^A decides membership in A in polynomial time, but what length queries are made to A? Unfortunately, there is no obvious reason that the fact that A is a self-1-helper should guarantee that there is an ordering of Σ^* relative to which all queries are length decreasing. In [K87] Ko discusses further conditions that can be added to the definition of self-1-helper so that this is true.

Balcázar, [Ba87], made a further attempt to characterize the class of self-1-helpers. He said that a set $A \in NP$ has *self-computable witnesses* if two conditions hold. First, there is a set $B \in P$ such that

$$x \in A \text{ iff } (\exists y)[|y| \leq p(|x|) \text{ and } \langle x, y\rangle \in B]$$

for some polynomial p; that is, there is a polynomially recognizable set of short witnesses for A. Second, there is a function f that is polynomially computable relative to A that finds short witnesses for A; that is, if $x \in A$, then $\langle x, f(x)\rangle \in B$. Notice that SAT has self-computable witnesses and this follows directly from the disjunctive self-reducibility property of SAT. Balcázar proves the stronger fact that all self-1-helpers have self-computable witnesses.

Theorem [Ba87]. *A set A is a self-1-helper if and only if it has self-computable witnesses.*

At this point we pose the following problem. The self-1-helpers seem quite similar to the self-reducible sets. Meyer, Paterson and Ko have given very nice characterizations of the Turing and truth-table self-reducible sets in terms of a partial ordering on Σ^* and a polynomially computable membership relation between elements of the ordering. Can one give a similar characterization for these sets, considered either as self-1-helpers, or as sets having self-computable witnesses?

3. P-Selective Sets

The second class of sets that we will discuss is the p-selective sets. Unlike the self-reducible sets, these sets have an ordering relation that is not necessarily well-founded.

The p-selective sets are the polynomial time analog of the semi-recursive sets studied by Jockusch [J68]. Selman [S79] gave the following definition for the p-selective sets.

Definition [S79]. *A set $A \subseteq \Sigma^*$ is p-selective if there is a polynomially computable function f such that for all $x, y \in \Sigma^*$,*
(i) $f(x,y) = x$ or $f(x,y) = y$, and
(ii) if $x \in A$ or $y \in A$, then $f(x,y) \in A$.

Thus, if all of the elements of A are "less than" the elements of $\overline{A}$, then the function f could always return the least element of a pair (x, y). To make this more precise, let $\prec$ be a polynomially testable linear ordering of Σ^* (it need not be well-founded). A set A is an initial segment of the ordering $(\Sigma^*, \prec)$ if $x \in A$ and $y \prec x$ implies that $y \in A$. Selman [S82a] observed that any initial segment of a polynomially testable linear ordering is p-selective.

Theorem [S82a]. *If A is an initial segment of some polynomially testable linear ordering of Σ^*, then A is p-selective.*

A partial converse to this theorem was also proved by Selman.

Theorem [S82b]. *A tally language is p-selective if and only if it is an initial segment of some polynomially testable linear ordering of $\{0,1\}^*$.*

In a later paper Ko [K83] fully characterized the p-selective sets in terms of initial segments of polynomial orderings.

Definition [K83]. *A binary relation $\preceq$ is called a preorder if it is reflexive and transitive. A preorder $\preceq$ on Σ^* is partially polynomial time computable if there is a polynomially computable function f such that*
(i) $f(x,y) = f(y,x) \in \{x,y\}$ *if* $x \preceq y$ *or* $y \preceq x$,
(ii) $f(x,y) = x$ *if* $x \preceq y$, *but not* $y \preceq x$ $(x \neq y)$, *and*
(iii) $f(x,y) = \perp$ *if neither* $x \preceq y$ *nor* $y \preceq x$, *where* $\perp \notin \Sigma$.

Given any preorder of Σ^* we can define a corresponding equivalence relation on Σ^* by letting $x \equiv y$ if and only if $x \preceq y$ and $y \preceq x$. For the equivalence classes of Σ^* with respect to $\equiv$ there is an induced preorder that we will denote $\preceq'$. Ko [K83] has shown the following.

Theorem [K83]. *A set $A \subseteq \Sigma^*$ is p-selective if and only if there is a partially polynomial time computable preorder $\preceq$ on Σ^* such that if $\equiv$ and $\preceq'$ are the corresponding equivalence relation and induced preorder, then*
(i) $\preceq'$ *is a linear ordering, and*
(ii) A *is the union of an initial segment of* $(\Sigma^*/\equiv, \preceq')$.

A natural question to ask at this point is how difficult can the membership question be for p-selective sets. The first theorem above due to Selman has as a corollary that every standard left cut is a p-selective set. (If $r \in [0,1]$ is a real number, then the standard left cut of r is $\{d \in \Sigma^+ : d < r\}$ where the elements of Σ^+ are viewed as numbers written to the base $|\Sigma|$.) Since it is known that there are arbitrarily difficult real numbers, there can be arbitrarily difficult p-selective sets.

Theorem [S79]. *There exist arbitrarily difficult p-selective sets.*

Nevertheless, there are certain sets for which we know that if they are p-selective, then they will be in P.

Theorem [S82b]. *If A is ptt-self-reducible (positive truth-table) and p-selective, then $A \in P$.*

Corollary [S79]. *If SAT is p-selective, then $SAT \in P$.*

Finally, it follows directly from Ko's characterization of the p-selective sets that all p-selective sets have polynomial size circuit complexity. In fact this result can be proved for a somewhat larger class of sets, the *weakly p-selective sets*, which contain both the p-selective sets and the *left cuts of real numbers.*

Definition [K83]. *A partial ordering $\preceq$ on Σ^* is p-linear if for every n, the set $\Sigma_n = \{x : |x| \leq n\}$ can be decomposed into at most $p(n)$ many pairwise disjoint and incomparable subsets each of which is linearly ordered by $\preceq$. A set $A \subseteq \Sigma^*$ is called weakly p-selective if and only if there is a partially polynomial time computable preorder $\preceq$ on Σ^* such that if $\equiv$ and $\preceq'$ are the induced equivalence relation and partial ordering, and there is a polynomial q such that*
(i) $\preceq'$ *is p-linear on* $\Sigma^*/\equiv$, *and*
(ii) for every n, A_n *is the union of initial segments of at most* $q(n)$ *many* $\preceq'$*-chains in* Σ^n.

Theorem [K83]. *If A is weakly p-selective, then $A \in P/poly$.*

What we have seen in this review of work by Ko, Selman and others is that the ordering structure relative to which p-selective sets have polynomially computable selection functions can be more complicated than the ordering structure found in self-reducible sets; it need not be well-founded. As a result, p-selective sets can be arbitrarily difficult while self-reducible sets are all in $PSPACE$.

The restrictive nature of the membership relation given by the selection function that relates x and y by telling us that $([f(x,y) = y] \Rightarrow [x \in A \Rightarrow y \in A])$, yields that the p-selective sets are in

P/poly, while the more complicated membership relation derived from the self-reduction function probably does not lead to small circuit characterizations.

Thus, both the *ordering structure* and the *type of membership information* obtainable in polynomial time play roles in determining computational complexity of the set. We will build on this idea in the remaining sections.

At this point, we stop to compare what we know about these two classes of sets, as well as to preview near-testable and p-cheatable sets. The properties displayed in Table 1, which follows, include our primary focus, the ordering relation; the testable relation; the complexity with respect to Turing machines and to Boolean circuits; and immunity. (Immunity is more of a negative measure of complexity: can a set in one class "avoid" all of the sets that are polynomially decidable?)

Finally, it will be useful to build on some of the ideas developed earlier in the study of semi-recursive sets. Thus, we conclude this section with a brief review of some of their properties.

Semi-Recursive Sets

Jockusch defined the semi-recursive subsets of the natural numbers, N, and used these sets to prove a variety of results concerning reducibilities and degrees of r.e. sets. It happens that these sets have a tree structure or partial order very similar to that of the cheatable sets discussed in Section 5. In fact, many of the techniques Jockusch used in proofs about semi-recursive sets carry over to the cheatable sets.

Definition [J68]. *A set $A \subseteq N$ is semi-recursive if there is a total recursive function f such that for all $x, y \in N$,*
(i) $f(x,y) = x$ or $f(x,y) = y$, and
(ii) if $x \in A$ or $y \in A$, then $f(x,y) \in A$.

Jockusch presented three theorems that provide nice information about the structural (ordering) properties of semi-recursive sets.

Definition. *A set A is regressive if there is some enumeration of A without repetitions (it need not be effective) for which there is a partial recursive function f such that*
(i) if $x \in A$, then $f(x)$ is defined, and
(ii) if $x \in A$, then $f(x) = x$ if x is the smallest element with respect to the enumeration of A, and $f(x)$ is the next smallest element with respect to the enumeration of A, otherwise.

If A is regressive and its complement is r.e., then the members of A and $\overline{A}$ can be viewed as nodes on an infinite tree, where the regressing function is a total recursive function that always returns a node's parent. If a node is in A, then its parent will always be less than the node with respect to the enumeration for A, and all elements of the regressive set, A, lie on a single branch of the tree. It is this idea that we will build on in later sections. We note here that this tree is a well-founded partial order, since we can regress from any node to the root in a finite number of iterations of f.

Theorem [J68]. *If a set $A \subseteq N$ is r.e. and $\overline{A}$ is regressive, then A is semi-recursive.*

In addition, Jockusch presented a partial converse to this result, attributed to Martin.

Definition. *A set $A \subseteq N$ is retraceable if there is a partial recursive function f such that*
(i) if $x \in A$, then $f(x)$ is defined, and
(ii) if $x \in A$, then $f(x) = x$ if x is the smallest element of A, and $f(x)$ is the next smallest element of A otherwise. (Here we use the standard less than ordering on N.

Theorem [J68]. *(Martin) If A is an infinite semi-recursive set, then A has an infinite co-r.e. retraceable subset.*

Nevertheless, not all semi-recursive sets are regressive.

Self-Reducible Sets

Ordering on Σ^* : Polynomially well-founded, polynomially testable partial ordering.

Testable Relation

Given: x

Find: a truth table t, and elements $x_1, ..., x_n$ such that $x_i < x$ and $[x \in A \iff t(\chi_A(x_1), ..., \chi_A(x_n))]$. For 2-disjunctive self-reducible sets this relation gives $[x \in A \iff x_1 \in A \vee x_2 \in A]$.

Complexity

Sequential: P $\subseteq$ Self-Reducible $\subseteq$ PSPACE

Parallel: It is conjectured that [Self-Reducible - P] $\cap$ P/poly $= \emptyset$.

P-Selective Sets

Ordering on Σ^* *:* Partially polynomially computable preordering, which may be dense and need not be well-founded.

Testable Relation

Given: x and y.

Select one of the following relations: $[y \in A \rightarrow x \in A]$ *or* $[x \in A \rightarrow y \in A]$.

Complexity

Sequential: Arbitarily difficult.

Parallel: P-Selective $\subseteq$ P/poly.

Immunity: There are recursively bi-immune p-selective sets.

Near-Testable Sets

Ordering on Σ^* : Polynomially testable, exponentially well-founded, linear ordering.

Testable Relation

Given: x

Compute one of the following relations: $[x \in A \iff x+1 \in A]$ or $[x \in A \iff x+1 \notin A]$.

Complexity

Sequential: P $\subseteq$ NT $\subseteq EXP \cap PSPACE$.

Parallel: Little is known.

Polynomial Immunity: Any near-testable set A that is not in P is immune of polynomially decidable, uniformily dense sets D for which $D \cap A \in P$.

P-Cheatable Sets

Ordering on Σ^**:* All known p-cheatable sets have a partial ordering similar to that of a p-selective set.

Testable Relation: (2 for 1)-pcheatable sets.

Given: x_1 and x_2.

Find one of the following: $[x_1 \in ?A]$ *or* $[x_2 \in ?A]$ *or* $[x_1 \in A \iff x_2 \in ?A]$

Complexity

Sequential: Arbitrarily difficult.

Parallel: May be in *P/poly* without being $p-close$; otherwise little is known.

Polynomial Immunity: No (2 for 1)-pcheatable set is polynomially bi-immune; otherwise, p-cheatable sets may be recursively bi-immune.

Table 1.

Theorem [J68]. *If A is semi-recursive and bi-immune, then neither A nor $\overline{A}$ is regressive.*

In Section 5, when we discuss cheatable sets, we will see that the tree structures common to retraceable and regressive sets often makes these sets cheatable. Thus, we will see that the semi-recursive, retraceable, regressive and cheatable sets are very closely related in terms of internal structure.

4. Near-Testable Sets

In this section we introduce a new class of sets that we call the *near-testable* sets. These sets have a very simple ordering property associated with them, and they also have very restrictive membership information that can be obtained in polynomial time. Like the self-reducible sets, this class lies somewhere between P and $PSPACE$. Despite the restrictive nature of the membership relation, we conjecture that not all of these sets are in P.

Before we define the near-testable sets, we will motivate our definition by considering a class that Balcázar called the *word-decreasing queries* self-reducible sets.[1]

Definition [Ba87]. *A set A is polynomial time wdq-self-reducible if there is a polynomial time deterministic oracle Turing machine M such that M^A decides membership in A and for each input x, all queries to A have length less than or equal to the length of x and if the query has the same length as x, then it must lexicographically precede x.*

Balcázar's reason for studying these sets was very similar to our reason for writing this paper. He wanted to provide a unifying structure for a collection of results. The results he was concerned with all had the form:

If class C is in class $D/poly$, then something collapses.

Balcázar's primary result is that the theorem quoted at the beginning of Section 2 from [BBS86] is true for wdq-self-reducible sets, as well as for Turing self-reducible sets. That is,

Theorem [Ba87]. *Suppose that A is wdq-self-reducible and that there is a $k \geq 0$ such that $A \in \Sigma_k^P/poly$. Then $\Sigma_2^P(A) \subseteq \Sigma_{k+2}^P$.*

With this result Balcázar is able to show that many known results follow from a single proof technique.

In this section we study a very simple subclass of wdq-self-reducible sets. Among other properties, these sets have the property that the membership question for an element x can always be polynomially self-reduced to the membership question for $x - 1$, in the lexicographic ordering of Σ^*.

Definition. *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*

$$f(x) = \chi_A(x) + \chi_A(x+1) \ (mod\ 2),$$

is polynomially computable.

Thus, with respect to the lexicographic ordering on Σ^* we can fully relate the membership questions for x and $x + 1$, where $x + 1$ denotes the lexicographic successor of x. This gives us a linear, well-founded, effective ordering on A.

---

1. The near-testable sets were introduced and studied by the current authors before they were aware of Balcázar's work. In hindsight, his notion of word-decreasing query reducibility provides a nice motivation for the investigation of near-testable sets.

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 last element of each contiguous sequence of elements of A and of $\overline{A}$. Formally, we have

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

We can picture a set as follows:

Let us say that A is represented by the thicker blocks, $\overline{A}$ by the thinner blocks, and $boundary(A)$ by the |'s. Notice that $boundary(A) = boundary(\overline{A})$.

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

Notice that the definitions we have given are absolutely dependent on the order in which a set is encoded. We will generally consider subsets of $\{0,1\}^*$ under that lexicographic order, where "$x+1$" means the next element in that order. When we consider sets such as $\{primes\}$, or SAT, we may think of them as being over other alphabets. As we know, there are standard polynomially computable mappings from decimal to binary. However, we need to be careful to specify which encoding we will be using, as other isomorphic encodings may (unfortunately) yield quite different boundaries.[2]

We could have given a more general definition of *near-testable* that would have avoided some of these problems. Such a definition might say that a set, A, is near-testable if there is some effective, well-founded linear order $\preceq$, and a polynomially computable function $f(x) = \chi_A(x) + \chi_A(x+1)\ (mod\ 2)$, where $x+1$ is the successor of x in the $\preceq$ ordering. As stated, this definition would allow us to construct sets that are near-testable, but are not decidable in polynomial space or exponential time. If we wanted to ensure that $NT \subseteq PSPACE \cap EXP$, then we would have to further require that the $\preceq$ ordering be exponentially well-founded and that x and $x+1$ be polynomially related in size. Having generalized the definition in this fashion, we could then show that NT is invariant under polynomial isomorphisms. Nevertheless, this more general definition seems to give less information about the structure of individual sets, so for now we will continue to concentrate on the original, simpler, definition.

The main question that we will address in this section is: What is the time complexity of near-testable sets? We do not even know whether $NT = P$, although we will present evidence that this is not likely.

We begin by noting that all near-testable sets are exponentially decidable. On the other hand, a straightforward slow diagonalization allows us to construct sets that are exponentially decidable but not in NT.

Observation.
(i) If $A \in NT$, then $A \in EXP \cap PSPACE$, and
(ii) there is an $A \in EXP$, such that $A \notin NT$.

At this point it is reasonable to ask:

---

2. Notice that a similar problem arises for the self-reducible sets. Under the natural encoding and the natural lexographic ordering of Σ^*, SAT is self-reducible. However, under other polynomial encodings, SAT might not be self-reducible.

Question. *Are there sets in NP, $coNP$, ZPP, etc. that are not in NT?*

NT appears to be sensitive, both to the distribution of elements in a set, and to density. The next observation follows from the fact that primes (except for 2) are distributed only throughout the *odd* integers.

Observation. *If $\{primes\} \in NT$, then $\{primes\} \in P$.*

Since the set of primes is known to be in ZPP, this tells us that if the set of primes is not in P, then $ZPP \not\subseteq NT$.

We can formalize the argument used to show that sets such as $\{primes\}$ are not in NT unless they are in P, as follows.

Definition. *We say a set D is uniformly dense if there is a polynomial $p(n)$ such that for any string x, there is an element of D within $p(|x|)$ 'steps' of x in the lexicographic ordering.*

Lemma. *If there is a uniformly dense set D such that $D \in P$ and $D \cap A \in P$, then $A \in NT$ implies $A \in P$.*

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

Theorem. *If $P \neq NP$, then there is a "reasonable" encoding of the NP-complete set $3SAT$ that is not in NT.*

Proof. We will assume that $3SAT$ is the set of satisfiable Boolean formulas in 1, 2, or 3 conjunctive normal form. We need to show that there is a uniformly dense polynomial set that intersects $3SAT$ in a polynomially recognizable set. This will show that if $3SAT$ is in NT, then it is in P. The uniformly dense polynomial subset of $3SAT$ we will use is $Domain(2SAT \cup 1SAT)$. Since this encoding of $3SAT$ contains the usual $3SAT$ in a polynomially recognizable manner, all of the usual reductions to $3SAT$ will still hold. Thus, the salient feature, its NP-completeness, is preserved.

We will code Boolean formulas as

$$nliteral\sharp nliteral\sharp ...\sharp nliteral\sharp\sharp k$$

where n tells whether or not the literal was negated, and $k = 1$, 2, or 3. The k tells us to divide up the string into clauses of exactly k literals, except perhaps the last clause, which gets the remainder of the literals. We need this sloppiness with respect to the last clause so that there are strings of *every* length in $Domain(2SAT \cup 1SAT)$, in order to make $Domain(2SAT \cup 1SAT)$ *uniformly* dense. This will guarantee that for *each* string of literals, there are Boolean formulas in $2SAT$ and $1SAT$ form with that string of literals. (Each string of literals gives rise to three Boolean formulas, since it may be followed by a 1, 2, or 3.) We have added the extra symbol $\sharp$ to the alphabet for ease of exposition. In fact, we can assume that all of the encoding is done over $\{0,1\}^*$.

Since $Domain(2SAT \cup 1SAT) \in P$ and $(Domain(2SAT \cup 1SAT)) \cap 3SAT = 2SAT \cup 1SAT \in P$, the conditions of the lemma are satisfied. If $3SAT$, under this encoding, were in NT we would have $3SAT \in P$ and thus $P = NP$. $\Rightarrow\Leftarrow$ ∎

The Berman-Hartmanis conjecture states that all NP-complete sets are p-isomorphic. Thus if it is true, then a form of the above theorem holds for all NP-complete sets, since $3SAT$ could then be considered an effective (though perhaps not reasonable) encoding of any NP-complete set.

Although the previous results show that it is not likely that all sets in NP, $coNP$, and ZPP are in NT, they do not address the converse question of whether there are *any* sets in NT that are not in P. We believe that these sets exist and our next results give evidence of this.

Definition. *f is a one-way function if f is polynomially computable, polynomially honest (i.e. f^{-1} is polynomially bounded), 1-1, and f^{-1} is not polynomially computable.*

Theorem. *If there is a one-way function, then there is a set Q that is in $NT - P$.*

To prove this result it is useful to introduce the notion of the *parity* of a set.

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

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

Useful Observation. *$A \in P$ if and only if the parity of the boundary of a set A is in P.*

Let us consider the earlier picture of the boundary of a set that we used to illustrate the definition of $boundary(A)$. In order to decide whether a given x is in A, we need to know whether it is in a thick-blocked region. This will be the case if there are an even number of |'s preceding it, which is precisely when $parity_{boundary(A)}(x) = 0$. (Of course, if the thinner blocks had represented A, we would have had $x \in A \Leftrightarrow parity_{boundary(A)}(x) = 1$.) Therefore, calculating membership in A is precisely the same as calculating $parity_{boundary(A)}$. If one of these calculations is in P, the other must be.

Proof of the theorem. We will implicitly construct the set $Q \in NT - P$ by constructing $T = boundary(Q)$, so that $T \in P$ but $parity_T \notin P$. By the **Useful Observation**, this will place $Q \in NT - P$, as desired. T will be composed of pairs $\langle x, f^{-1}(x)\rangle$, where $\langle x, y\rangle$ will not be the usual pairing function, but one more suited to our needs. We will show that if $parity_T \in P$, then $f^{-1}(x)$ is polynomially computable.

Let f be a one-way function, and let $q(n)$ be a strictly increasing polynomial bound for both f and f^{-1}. We will only be concerned about pairs (x, y) where y could possibly be $f^{-1}(x)$. This is the case only when $|y| \leq q(|x|)$. We will call a pair (x, y) *relevant* if $|y| \leq q(|x|)$.

What sort of properties do we need our pairing function to have? We need to have the pairs sorted by their first element, so all $\langle x, y\rangle$'s come before all $\langle x+1, z\rangle$'s; we need the pairing function to be computable in polynomial time in the first element (remember that for relevant pairs, the length of the second element is also bounded by a polynomial in the length of the first); and that the decoding function is polynomially computable in the length of the string representing the pair.

Formally, we define $T = \{\langle x, f^{-1}(x)\rangle : x \in range(f)\}$, where $\langle x, y\rangle$ is the following pairing function:

- If (x, y) is irrelevant, then $\langle x, y\rangle = 0$.
- If (x, y) is relevant, then $\langle x, y\rangle$ is a string of length $|x| + 2q(|x|)$, with prefix x. The suffix of the string will be y interleaved with $1^{|y|}$, followed by sufficient 0's to make the length of the suffix $2q(|x|)$, like this:

$$x_1 x_2 \ldots x_n \underbrace{1y_1 1y_2 \ldots 1y_k 0 \ldots 0}_{2q(|x|) \text{ bits}}.$$

Notice the following properties, where we use $<$ for the lexicographic order:

- $\langle x,y\rangle < \langle z,w\rangle < \langle x,u\rangle \Rightarrow x = z$.
- The decoding map from $\langle x,y\rangle$ to x, y is computable in time bounded by a polynomial in $|\langle x,y\rangle|$. (Given the string $\langle x,y\rangle$, in order to find x and y, we must first find n so that $|\langle x,y\rangle| = n+2q(n)$. Since q is strictly increasing, $n + 2q(n)$ is as well, and thus is invertible in time polynomial in n. Once we have n, we simply read off x; decoding y is equally straightforward.)
- The encoding of (x,y) as $\langle x,y\rangle$ is computable in time bounded by a polynomial in $|x|$.

We need to show that $T \in P$, and that $parity_T \notin P$. To decide if $x \in T$, we first "decode" x. If $x \neq \langle a,b\rangle$ for any (a,b), then we know $x \notin T$. If $x = \langle a,b\rangle$, then $x \in T \leftrightarrow f(b) = a$. Since f is polynomially computable, this can be determined in time polynomial in $|x|$.

Suppose $parity_T$ were in P. Then we would be able to calculate $f^{-1}(x)$ in time polynomial in $|x|$, using the following algorithm. First we must decide if $f^{-1}(x)$ exists. This will be the case if and only if $\langle x,y\rangle \in T$ for some y. If this is the case, we will use binary search to find it.

Notice that $\langle x,a\rangle < \langle x,b\rangle \Leftrightarrow a < b$. We will write $z+1$ to indicate the next string in lexicographic order after z. Notice also that the strings immediately preceding $\langle x,0\rangle$ and immediately following $\langle x,1^{q(|x|)}\rangle$ do not code pairs. Now, if $f^{-1}(x) = y$, then the string $\langle x,y\rangle$ will be the only string in T between $\langle x,0\rangle - 1$ and $\langle x,1^{q(|x|)}\rangle + 1$, and otherwise there will be no string in this interval in T. Therefore $parity_T(\langle x,0\rangle - 1) \neq parity_T(\langle x,1^{q(|x|)}\rangle)$ if and only if $f^{-1}(x)$ exists. Once we have determined that it does exist, we use binary search on the strings that encode (x,y), for the $2^{O(q(|x|))}$ relevant y's, to locate it. This will take time $O(q(|x|))$. Since f was chosen to be one-way, this is a contradiction. Therefore, $parity_T$ is not in P, and thus by the **Useful Observation,** if T is the boundary of a set, that set cannot be in P either. Therefore we define Q such that its boundary is T. Explicitly, $0 \notin Q$ and $boundary(Q) = T$. ∎

The **Useful Observation** tells us that if $A \in NT - P$, then the parity of the boundary of A cannot be in P. But, how close to P can it be? The next theorem shows that if there are one-way functions that are *onto*, then the parity of the boundary of Q can be in $NP \cap coNP$.

Theorem. *If f in the previous construction is both one-way and onto*[3]*, i.e. if $f^{-1}(x)$ is uniquely defined for each x, then $parity_T \in (NP \cap coNP) - P$.*

Proof. The first thing that we need in order to define an NP (and $coNP$) algorithm for $parity_T$ is the following observation about the pairing function: strings of length $n + 2q(n)$ encode pairs (x,y) for an even number of x's of length n (2^n of them, to be precise). Thus $parity_T(1^{n+2q(n)}) = 0$, and for $|z| \neq n + 2q(n)$ for any n, $parity_T(z) = 0$.

If $|z| = n + 2q(n)$, let x be the prefix of z of length n. Let i be the rightmost bit of x. This will tell us whether there is an even or odd number of w's of length n preceding x. Notice that for each such w, there is precisely one string $\langle w,u\rangle$ in T.

Nondeterministic step: guess $y = f^{-1}(x)$.
Deterministically verify that $f(y) = x$.
If $\langle x,y\rangle \leq z$ **then** $parity_T(z) = 1 - i$
else $parity_T(z) = i$.

---

3. Grollman and Sellman show in Theorem 11 of [GS84] that the existence of such a function is equivalent to $P \neq U \cap coU$. Our theorem is in fact a variation of a result from Brassard, Fortune, and Hopcroft, [1978], which is quoted as Exercise 13.24 in Hopcroft and Ullman ([HU79]), that asks the reader to prove $\{\langle x,y\rangle : f^{-1} < y\} \in (NP \cap coNP) - P$ if f is a one-way function.

This is the same verification procedure for both $parity_T$ and $\overline{parity_T}$. This is because of the definition of T, rather than because of the symmetric properties of $parity_T$. ∎

Notice that the set Q of the previous proof is *not* polynomially bi-immune. We know by a preceding lemma that there cannot be a uniformly dense polynomial set which intersects Q in a polynomial set. Thus Q and $\overline{Q}$ must be polynomially bi-immune with respect to uniformly dense sets. The set of all strings z, where $|z| \neq n + 2q(n)$ for any n, forms a huge polynomial subset of $\overline{Q}$. This does not contradict the lemma, because this polynomial set is not *uniformly* dense, since it contains no elements at all of certain lengths. On the other hand, the original Q we constructed, without the assumption that f was onto, does not necessarily have this property.

The question remains, whether we actually require a one-way function in order to show that $NT \neq P$. We would like to show that the existence of a set in $NT - P$ implies the existence of a one-way function.[4] What we have is the following partial converse to the preceding two theorems.

Theorem. *If there is a set $Q \in NT - P$ with a sparse boundary, then there is a polynomially-many-to-one, polynomially computable, polynomially honest one-way function.*

As is standard, we define $f^{-1}(x) = \{y : f(y) = x\}$. Notice that if f is polynomially-many-to-one, it is conceivable that, given x, we could find the entire set $\{y : f(y) = x\}$ in time polynomial in $|x|$.

Proof. Let $S = boundary(Q)$. Let $q(n)$ be a polynomial bound on the census function for S. We define

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

Since $Q \in NT$, $S \in P$, 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 $f^{-1}(x)$ *were* polynomially computable. By the sparseness of S, given x we could compute $f^{-1}(0)$, $f^{-1}(0^2)$, ..., $f^{-1}(0^{|x|})$ in time polynomial in $|x|$, and we could count all those strings less than x which we have enumerated, to determine $parity_S(x)$. But we assumed that $Q \in NT - P$, which, by the **Useful Observation**, implies that the parity of the boundary of Q ($parity_S$) is not polynomially computable.⇒⇐ Therefore, f must be one-way. ∎

There are a number of questions that remain open concerning NT sets. The main question is obviously whether $NT = P$.

Main Question. *Is $NT = P$?*

A second question concerns the parallel complexity of sets in NT. Is $NT \subseteq P/poly$? To better understand this question it is useful to observe the following.

Observation. *If A or $\overline{A}$ is sparse and $A \in NT$, then $A \in P$.*

Observation. *If $boundary(A)$ is sparse, then $A \in P/Poly$, that is A has polynomial size circuits.*

Proof. To determine whether $x \in A$ it is sufficient to know the elements of $boundary(A)$ that are less than or equal to x. If $boundary(A)$ is sparse, then we can code the information about elements of the $boundary(A)$ into an oracle, B, in such a way that B is sparse and the information

4. Added in revision, July 1987: Lane Hemachandra has shown that *parity-P* $\neq P$ if and only if $NT \neq P$. The existence of one-way functions also implies *parity-P* $\neq P$. These and related results will be contained in a forthcoming paper by Hemachandra and the current authors.

can be retrieved in a polynomial fashion. Thus $A \leq_T^P B$. This implies that $A \in P/poly$, which in turn implies that A has polynomial size circuits. ∎

The hypothesis of our preceding theorem was that there is a set in $NT - P$ with a sparse boundary. We do not in fact know that such sets exist.

Question. *If $A \in NT - P$, can boundary(A) be sparse?*

Notice that if $A \in NT$ and either A or $\overline{A}$ is sparse then $A \in P$ by the observation made above. Therefore, if $A \in NT - P$ and *boundary*(A) sparse, then A and $\overline{A}$ are both unions of intervals many of which contain exponentially many strings.

Finally, it is reasonable to ask *exactly* how robust is the definition of near-testable. One way to formalize this question is to ask whether it is preserved under polynomially computable isomorphisms.

Observation. *If there is an $A \in NT - P$ with a polynomial padding function, then near-testability is not preserved by polynomial isomorphisms.*

Proof. Let $T = \{x0 : x \in A\}$. Since A is paddable T is also, and thus $A \cong^P T$. We would expect, therefore, that $T \in NT - P$. However, since every other string is not in T, we know that $x \in T$ if and only if $x \in boundary(T)$. Therefore $boundary(T) \in P \Rightarrow T \in P$. In other words, if $T \in NT$, then $T \in P$.⇒⇐ ∎

It is natural here to look at the relationship between p-isomorphisms of sets and p-isomorphisms of their boundaries. We might expect that p-isomorphic sets would have p-isomorphic boundaries, but this is not always the case.

Observation. *If $A \cong^P T$, this does not imply that boundary$(A) \cong^P$ boundary(T).*

Let $A = \{0x : x \in \{0,1\}^*\}$ and $T = \{x0 : x \in \{0,1\}^*\}$. $A \cong^P T$, but the boundary of T is all of $\{0,1\}^*$, and the boundary of A contains only two elements of each length, so is sparse. Therefore $boundary(A) \not\cong^P boundary(T)$. ∎

If we look at a p-isomorphism between boundary sets, it is likely to "scramble" the pairs of elements of the set that define intervals in the base set. Therefore, we would not expect all p-isomorphisms of boundary sets to induce p-isomorphisms between the base sets. In fact, even if we restrict our attention to p-isomorphisms of boundaries that respect *pairs* of elements, these do not necessarily induce isomorphisms on the corresponding intervals.

Observation. *There are sets S and T such that boundary$(S) \cong^P$ boundary(T) and $S \not\cong^P T$.*

Construction. Both S and T will be the unions of exponentially long intervals. S is the set of all strings of lengths from $2^{2k-1} + 1$ to 2^{2k} for each positive k.

$$boundary(S) = \{1^{2^n} : n \in \mathbf{N}\}.$$

If we think of *boundary*(S) as dividing $\{0,1\}^*$ into intervals I_n, then $S = \bigcup_{n \text{ even}} I_n$. We set

$$boundary(T) = \{1^{2^r} : r \text{ odd}\} \cup \{0^{2^t} : t \text{ even}\}.$$

It is easy to see that *boundary*$(S) \cong^P$ *boundary*(T), and the p-isomorphism respects pairs of elements.

We can think of $T = \bigcup_{n \text{ even}} I'_n$, where for n even, $I'_n = I_n$ minus all strings of length 2^{n+1}, except for 0^{n+1}.

To show that a particular p-isomorphism i does not map S to T, choose n large enough so that i cannot map any element of I_n to anything in I'_{n+2}, or higher. Here, we are using the fact that these intervals are exponentially long. There are $2^{2^{n+1}} - 1$ more strings of lengths $\leq 2^{n+1}$ in S than there are in T, and i cannot map them to any other strings in T. Therefore, i is not an p-isomorphism from S to T. ∎

What we have seen in this section is that, by defining a class of sets that have a very simple ordering structure and a fairly strong polynomially computable membership relation, we have obtained a class of sets of bounded complexity ($PSPACE \cap EXP$). Unfortunately, without knowing more about the boundaries of NT sets we do not know whether they are in $P/poly$. It is worth observing that for a p-selective set, with respect to its preordering relation, there is essentially only one point in the boundary. (For a weakly p-selective set, A, there are at most polynomially many elements in the boundary of A^n, A restricted to elements of size n.) This is what places these sets in $P/poly$.

5. P-Cheatable Sets

Amir, Beigel, Gasarch, Gill, Hay, and Owings have studied a class of sets they call the *p-cheatable* sets. This class and related classes that they call *terse, super terse* and *verbose* have many of the same ordering properties that we have been considering.

Before we discuss p-cheatable sets *per se,* let us take another look at the definition of near-testable sets. A set is near-testable if the membership relation function

$$f_A(x) = \chi_A(x) + \chi_A(x+1) \pmod 2$$

is polynomially computable. Notice that if we make the membership relation function a function of two arbitrary arguments, that is

$$f'_A(x,y) = \chi_A(x) + \chi_A(y) \pmod 2,$$

then the existence of a polynomially computable membership relation function f'_A implies that $A \in P$. However, suppose we consider a slightly different modification; suppose that A has a polynomially computable membership relation function $f''_A(x,y)$ that computes *either* a relationship between the membership question for x and y, i.e., $f''_A(x,y) = \chi_A(x) + \chi_A(y) \pmod 2$, *or* decides membership for x or y. That is, for each x and y,

$$f''_A(x,y) = \begin{cases} \chi_A(x) + \chi_A(y) \pmod 2, \text{ or} \\ x \in A \ (x \notin A), \text{ or} \\ y \in A \ (y \notin A). \end{cases}$$

Note that if we required f''_A to compute either a relationship or decide membership for both x *and* y, then we would again have that the existence of a polynomially computable f''_A implies $A \in P$. However with the definition of a membership relation function that we have just given we can prove several results that will apply to p-cheatable sets.

Theorem 1. *Let A be any set. If there exists a polynomially computable function f_A such that*

$$f_A(x,y) = \begin{cases} \chi_A(x) + \chi_A(y) \pmod 2, \text{ or} \\ x \in A \ (x \notin A), \text{ or} \\ y \in A \ (y \notin A), \end{cases}$$

then A is decidable.

Proof. Suppose that we wish to decide whether or not a large element, say 10,000,000,000,000, is in A. We will begin by computing $f_A(0,1)$. This computation will either answer the membership question for one of these elements or it will relate their membership questions. If we learn the answer for one of the elements we simply dispense with that element, say 1, and repeat the process by computing $f_A(0,2)$. We continue in this fashion. The important thing to note at this point is that with each step of computing f_A on a new input pair, we either (i) learn the answer of a membership question for one of the inputs, or (ii) a relationship between the inputs, for instance one is in A if and only if the other is. Therefore if we repeatedly compute f_A on more and more input pairs, we will find out the membership question for more and more elements, or we will end up building a long chain of elements each with related membership questions. The important thing is that at most one infinite chain can develop. This is because if we have two chains developing, then by computing f_A on a pair containing one element from each chain we will be able to either join the chains to form a single chain or we will discover the actual answer to the membership question for one of the elements, and hence for all of the elements in its chain.

Thus for a given set A one of two things must be the case, (i) the above procedure never results in an infinite chain, or (ii) an infinite chain develops and its minimal element is x_0. If an infinite chain will never develop we simply consider more and more pairs, in some systematic fashion, until the membership question for 10,000,000,000,000 is resolved. If an infinite chain develops then we consider more and more pairs until either 10,000,000,000,000 is added to a chain containing x_0 or its membership is determined. If it is added to a chain containing x_0, then x_0 is either in A or $\overline{A}$ and using the membership relations that have been calculated we can answer the question for 10,000,000,000,000. ∎

Notice that in the proof of the theorem we did not use the fact that f_A is polynomially computable and even if it is, a straight-forward diagonalization allows us to construct sets that satisfy the hypotheses of Theorem 1 and have arbitrarily high complexity. Thus we have the following theorem as well.

Theorem. *There are sets satisfying the hypotheses of Theorem 1 that have arbitrarily high time complexity.*

Thus a seemingly minor weakening in the definition of near-testable allows sets to obtain much greater complexity.

What we would like to show now is that all p-cheatable sets satisfy the hypotheses of Theorem 1. In the form that we have stated Theorem 1 this is probably not true, and hence we need to prove a slightly stronger version of Theorem 1. Although formally awkward, the version of Theorem 1 give below will be useful when discussing p-cheatable sets.

Theorem 1'. *Let A be any set. Suppose there exist polynomially computable functions f_A and B such that*

$$f_A(x_1, x_2, ..., x_n) = \begin{cases} X, \text{ or} \\ x_i \in A \ (x_i \notin A) \text{ for some } i \leq n, \end{cases}$$

where X is a subset of $n-2$ of the x_i's with the property that if values are given for these x_i's then B can compute a membership relation between the remaining two x_i's. Formally,

$$B(X, y, z, v_1, v_2, ..., v_{n-2}) = \chi_A(y) + \chi_A(z) \ (mod\ 2),$$

where y, z are the x_i's not in X and the v_i's are values for the $x_i's \in X$. Then A is decidable.

Sketch of the proof. The proof of this theorem is essentially the same as the previous proof. However, in this case we need to assume that there are $n-2$ elements at the base of any infinite

chain whose membership in A or $\overline{A}$ is known. Again, with respect to a given set A at most one infinite chain can develop. ∎

We are now ready to relate Theorem 1' to the notion of p-cheatable sets.

P-Cheatable Sets

One of the scenarios used by Amir, Beigel, Gasarch, Gill, Hay, and Owings to motivate the study of the class of *p-cheatable* sets is the following. One is given a large collection of inputs, say 2^k different inputs, and one would like to determine whether or not each input is in a known set A. It is assumed that the membership problem for A is difficult and therefore if knowing that some x_i is in A (or not in A) helps in determining that another x_j is (or is not) in A, then one would like to use that information. To determine membership in A, one designs an oracle machine M that when given A as oracle will decide the membership question for n different inputs by asking the oracle as few questions as possible and doing as little computation as possible. They hope to design machines that run in polynomial time and for a fixed k decide membership for 2^k inputs by asking only k questions. When this can be done they say that A is *p-cheatable.* (We will be more explicit and call such sets $(2^k \ for \ k) - pcheatable.$)

We conclude this section by showing that all $(2^k \ for \ k)$-pcheatable sets satisfy the hypotheses of Theorem 1', and hence are decidable. This decidability result was first proved by Beigel, Gasarch, Gill and Owings [BGGO86]. We now give an exposition of this result.

We will begin by making the definition of cheatable somewhat more precise.

Definition. *A set A is $(n \ for \ k) - cheatable$ if there is an oracle machine M such that if M is given inputs $< x_1, ..., x_n >$ and an oracle for A, then with k or fewer queries to the oracle M determines membership in A for each of $x_1, ..., x_n$. If the oracle machine M runs in time that is polynomial in $| < x_1, ..., x_n > |$, then we will call A (n for k)-pcheatable.*

Note that the queries that M asks in the course of deciding membership for $x_1, ..., x_n$ are not restricted to come from the set $x_1, ..., x_n$. (In fact, the exact questions asked and the oracle to which they are addressed turn out to be irrelevant for our purposes. The only feature that matters is the *number* of questions that are asked.)[5] We will be primarily concerned with sets that are $(2^k \ for \ k)$-pcheatable for some fixed integer k. However, we will use the term *p-cheatable* when the particular values of n and k are not important or are obvious from the context.

Obviously, any set that is recursive is $(n \ for \ k)$-cheatable for all n and k. Similarly, any set that is polynomially decidable is $(n \ for \ k)$-pcheatable for all n and k. So the interesting situation arises when a set is *not* recursive (or *not* polynomially decidable) but is cheatable (or p-cheatable).

As stated above, we will show that all $(2^k \ for \ k)$-pcheatable sets satisfy the hypotheses of Theorem 1' and hence are decidable. This result is originally due to Beigel *et al.* In fact, [BGGO-87] gives a more subtle, more difficult, argument showing that all $(2^k \ for \ k)$-cheatable sets are decidable.

---

5. Beigel's definition of p-cheatable sets (what we call $(2^k \ for \ k)$-pcheatable) allows M access to *any* fixed oracle. However, the definitions given by Amir and Gasarch for $(2^k - 1 \ for \ k)$-pcheatable sets require that M use A as the oracle. (These sets are called *verbose* by Amir and Gasarch.)

Theorem. *Let A be any set. Suppose that there is a polynomial time oracle program M, an oracle C, and an integer k such that for all input vectors $< x_1, x_2, ..., x_{2^k} >$, there exists an $m \leq k$ such that with at most m queries to the oracle M^A decides membership with respect to A for more than $2^k - 2^{k-m}$ of the inputs. Then A satisfies the hypotheses of Theorem 1'.*

Observation. *If A is (2^k for k)-pcheatable, then A satisfies the hypotheses of the above theorem.*

Proof. We begin by considering the problem when $k = 1$. Suppose that we have a set A, an oracle C and a polynomial time oracle program M that takes as input pairs $< x_1, x_2 >$. If for every input pair $< x_1, x_2 >$, M^C asks at most one question of the oracle and outputs a vector $v =< v_1, v_2 >$ with $v_i = 1$ if $x_i \in A$ and $v_i = 0$ if $x_i \notin A$, then we claim that there is a polynomial time algorithm for A that satisfies the hypotheses of Theorem 1. We will describe such an algorithm.

We begin by simulating M^C on the input pair $< x_1, x_2 >$. When/if M^C queries C about a string q we simulate M^C on the two possible answers $q \in C$ and $q \notin C$. This will produce two possible output vectors u and v. If $u_i = v_i = 1$, then we know that $x_i \in A$ and if $u_i = v_i = 0$ we know that $x_i \notin A$. On the other hand, if $u_1 \neq v_1$ and $u_2 \neq v_2$, then we either know that $x_1 \in A$ iff $x_2 \notin A$ or that $x_1 \in A$ iff $x_2 \notin A$. The values of u and v tell us which we know. For instance, if $u =< 0, 0 >$ and $v =< 1, 1 >$, then we know that $x_1 \in A$ iff $x_2 \in A$, but if $u =< 0, 1 >$ and $v =< 1, 0 >$, then we know that $x_1 \in S$ iff $x_2 \notin A$. The other two cases are the same.

Thus, after simulating M^C on input $< x_1, x_2 >$, we either know the answer to the membership question for at least one of the inputs, or we know a relationship between the membership questions. Notice that, if given more than two inputs, M^C could still determine their membership values with a single oracle query, then the above algorithm would *work even better.* It would obtain more information when comparing the output vectors u and v.

We now discuss an algorithm for the more general problem: $k \geq 2$ and $m \leq k$. As before, we simulate M^C on the input vector $< x_1, x_2, ..., x_{2^k} >$. If M^C makes m queries, then we will obtain 2^m possible output vectors. Each output vector contains more than $2^k - 2^{k-m}$ zeros and ones and some number of *don't knows.* With this number of zeros and ones in each vector, we are guaranteed that there is a subset of 2^m inputs that have an output value of zero or one in each output vector, i.e., they are never assigned a *don't know* value. We will henceforth consider only this subset of inputs. Our remaining problem is solved by the proof of the following lemma.

Lemma. *Given 2^m distinct boolean vectors each containing 2^k values, $m \leq k$, either*

(i) there is a subset of $2^m - 2$ variables such that if the answers to the membership questions for these variables are known, then a relationship for the membership question for the last 2 variables is known, e.g. [$v_i \in S$ iff $v_j \notin S$], or

(ii) the answer to the membership question is known for at least one variable.

Proof. The idea is to show that there are $2^m - 2$ x_i's that allow us to distinguish between all but two of the possible output vectors. That is, if we knew the membership question for these $2^m - 2$ inputs, then we could be assured that there are at most two possibilities for the values of all 2^k of the x_i's. In other words, there are at most two vectors,

$$v_i =< i_1, i_2, ..., i_{2^k} > \qquad v_j =< j_1, j_2, ..., j_{2^k} >$$

that could be the output of M^C on input $< x_1, x_2, ..., x_{2^k} >$. Thus we are essentially back to the situation that we had when $k = 1$.

Now to see that these $2^m - 2$ x_i's exist we observe the following. Since each of the 2^k output vectors is assumed to be different, there must be an element x_{i_1} such that its value in output

vector v_1 differs from its value in v_2. Therefore if we knew its real membership value we could *rule out* one of these vectors as the output of M^C. Similarly, there must be an element x_{i_2} such that knowing the real membership value for x_{i_1} and x_{i_2} would allow us to *rule out* two of v_1, v_2, and, v_3 as possible outputs for M^C. Continuing in this fashion we build up a set of $2^m - 2$ x_i's that allow us to distinguish between the first $2^m - 1$ of the possible output vectors. More formally, the following program builds a set X containing the necessary x_i's.

$X = \emptyset$
$j = 2$
while $|X| < 2^m - 2$ **do**
  let $x = \min x_i$ such that $x_i \notin X$
    and $\{x_i\} \cup X$ distinguishes $v_1, v_2, ..., v_j$
  $X \leftarrow X \cup \{x\}$;
  $j \leftarrow j + 1$; **od**
end.

Now assume that we know the real membership values for the x_i's in X. Then we can narrow down the possible output vectors to one of the first $2^m - 1$ vectors and perhaps the $2^m th$ vector. (The $2^m th$ vector is possible only if it agrees with the known values, so we may assume that this is the case). At this point we have two 2^k element vectors that agree on $2^m - 2$ values. Since $k \geq m$ we are back in the situation covered by our base case.

Thus the function f_A simply computes X and the function B does back-substitution to build the relationship between the remaining two inputs. Since k is fixed and hence 2^m is bounded by a constant, the simulation of M^C takes time polynomial in the runtime of M^C. Thus, the algorithms that we have described run in polynomial time if M^C runs in polynomial time. ∎

The previous result, and the stronger result of [BGGO86] that preceded it, showed that *all* (2^k *for* k)-pcheatable sets are decidable. But, as with arbitrary sets that satisfy the hypotheses of Theorem 1', these sets can have arbitrarily high complexity, see [AG87]. In fact, we will prove a much stronger result: we will show that for any deterministic time class $Dtime(T(n))$ there are (n *for* 2)-pcheatable sets that are bi-immune with respect to $Dtime(T(n))$. This is in contrast to the situation for (2 *for* 1)-pcheatable sets, which are easily shown to have polynomially decidable subsets in either the set or its complement.

Recall that a set A is $(n\ for\ k) - pcheatable$ (n and k fixed constants) if there is a polynomial time oracle machine M such that if M is given inputs $< x_1, ..., x_n >$ and an oracle for A, then with k or fewer queries to the oracle, M determines membership in A for each of $x_1, ..., x_n$. If n can vary, that is, if the algorithm never makes more than k queries no mater how many inputs it is given, then we say that A is $k - pcheatable$.

Theorem. *Let $Dtime(T(n))$ be any deterministic time class. There is a 2-pcheatable set A that is bi-immune with respect to $Dtime(T(n))$. That is, neither A nor $\overline{A}$ has a $Dtime(T(n))$ decidable subset.*[6]

Proof. Let $\{M_i\}_{i \in N}$ be a canonical enumeration of total programs that contains all programs that run in $Dtime(T(n))$ and let $L(M_i) = \{x : M_i(x) = 1\}$. In addition, let $f(n)$ be a monotonically increasing function such that

6. Added in revision, July 1987. This theorem shows that a conjecture made in [Be86] is false. Beigel has informed us that he too has realized this and proved a similar theorem, which is presented in [Be87b].

(i) $f(n)$ is polynomially honest, that is, for all n the complexity of computing $f(n)$ is polynomially related to the length of $f(n)$,

(ii) $f(n)$ bounds the summation of the runtimes of all programs M_i, $i < n$, on all inputs of lengths less than or equal to $f(n-1)$, plus a little additional time to cover the overhead of the simulation.

We will divide the strings in $\{0,1\}^*$ into intervals $I_n = (1^{f(n-1)}, 1^{f(n)}]$ using the lexicographic ordering of the strings. At stage n in the construction all strings in I_n will be placed into either A or $\overline{A}$. The stages of the construction will perform a diagonalization argument to ensure that if $L(M_i)$ is infinite, then it is not a subset of A or $\overline{A}$.

Stage 0: Assume that $I_0 = \{0,1\}$, let $I_0 \subseteq A$ and place M_0 on the *active lists* for A and for $\overline{A}$.

Stage n:

1) Place M_n onto the *active lists* for A and for $\overline{A}$.
2) Run all programs on the active lists on all inputs in the interval I_n. Let n_0 be the smallest index of an active program such that $M_{n_0}(z) = 1$ for some $z \in I_n$, if such a program exits. If M_{n_0} is on the active list for A, then place I_n into $\overline{A}$, ensuring that $L(M_{n_0}) \not\subseteq A$, and remove M_{n_0} from A's active list. Otherwise, place I_n into A, ensuring that $L(M_{n_0}) \not\subseteq \overline{A}$, and remove M_{n_0} from $\overline{A}$'s active list. If no program M_{n_0} exists, place I_n into A.

Bi-immunity: By induction, if $L(M_n)$ is infinite, then there is a pair of stages (n_1, n_2) such that $L(M_n)$ contains elements in the intervals I_{n_1} and I_{n_2} and at these stages it is the smallest active program to contain elements in the intervals. During the first of these stages we will have ensured that $L(M_n)$ is not a subset of A by placing I_{n_1} into $\overline{A}$ and during the second will have similarly ensured that $L(M_n)$ is not a subset of $\overline{A}$ by placing I_{n_2} into A.

P-cheatability: We must give a polynomial time oracle algorithm that when given inputs $(z_1, ..., z_{2^k})$, k a variable, decides membership in A for each of the inputs and makes at most 2 queries to A.

Assume that the inputs are sorted lexicographically and consider the positions of the inputs in the intervals used to construct A. Since f is polynomially honest, in polynomial time we can determine the interval in which each z_i is contained. Assume that they are contained in the intervals I_0 through I_n. Notice that z_{2^k} is large enough that our entire diagonalization construction up through interval I_{n-2} can be recomputed in time polynomial in $|z_{2^k}|$. (This is because $|z_{2^k}| > f(n-1)$ and $f(n-1)$ was defined so that this would be true.) Therefore to decide membership for all the z_i's in the intervals $I_0, ..., I_{n-2}$ we simply repeat the construction. To decide membership in the intervals I_{n-1} and I_n requires only 2 queries to the oracle, since the intervals are each entirely contained within A or $\overline{A}$. ∎

At this point it is interesting to recall the definition of semi-recursive sets given in Section 3. Because of the ordering structure present in semi-recursive sets, all semi-recursive sets are $(2^k - 1\ for\ k)$-cheatable. Jockusch [J68] has shown that it is possible to construct semi-recursive sets that are bi-immune and thus there are $(2^k - 1\ for\ k)$-cheatable sets that are bi-immune. (This fact and some other results of Beigel, Gasarch and Owings, [BGO87], can be regarded as extensions of Jockusch's results.) If one is careful in reconstructing Jockusch's proof one can in fact ensure that the bi-immune semi-recursive set is p-cheatable and at the same time in $P/poly$, thus it is decidable by polynomial size circuits.[7]

7. Added in revision, July 1987: In fact, with little additional work, one can guarantee that the set constructed here will not only be *not p-close, but not even recursively-close.* A proof of this will be contained in an upcoming technical report.

We conclude with the questions: What type of partial ordering must exist on a p-cheatable set? Can the p-cheatable sets be characterized by ordering structure?

Acknowlegements

The authors would like to thank the many researchers who have proved the theorems that make this paper possible. In addition we thank the people who have pointed out additions and corrections to the paper since its first draft. Special thanks go to Eric Allender for bringing Balcázar's recent work to our attention, and to Jose Balcázar for a careful reading and useful comments on the penultimate draft of the paper.

Bibliography

[AG87] A. Amirhood and W. Gasarch, "Polynomially terse sets," *Proceedings of the Second Annual Structure in Complexity Conference* (June 1987), 22-27.

[Ba87] J. Balcázar, "Self-Reducibility," *STACS Proceedings* (1987).

[BBS86] J. Balcázar, R. Book, and U. Schöning, "The polynomial time hierarchy and sparse oracles," *J. ACM* **33** (1986), 603-617.

[B78] P. Berman, "Relationship between density and deterministic complexity of NP-complete languages," *Symposium on the Math. Found. of Comput. Sci., Lecture Notes in Computer Science* **62** (1978), 63-71.

[BH77] L. Berman and J. Hartmanis, "On isomorphisms and density of NP and other complete sets," *SIAM J. of Comput.,* **6** (1977), 305-322.

[Be86] R. Beigel, "Bounded Queries to SAT and the Boolean hierarchy," *Preprint* (Nov. 1986), 1-14.

[Be87a] R. Beigel, "A structural theorem that depends quantitatively on the complexity of SAT," *Proceedings of the Second Annual Structure in Complexity Conference,* IEEE Computer Society (June 1987), 33-40.

[Be87b] R. Beigel, "Immunity and separation results for cheatable sets," *Preprint* (Apr 87).

[BGGO87] R. Beigel, W. Gasarch, J. Gill and J. Owings, "Terse, superterse and verbose sets," *Technical Report* **TR-1806**, University of Maryland (March 1987), 1-25.

[BGH87] R. Beigel, , W. Gasarch, and L. Hay, "Bounded query classes and the difference hierarchy," *Technical Report* **TR-1847**, University of Maryland (1987), 1-26.

[BGO87] R. Beigel, , W. Gasarch, and J. Owings, "Terse sets and verbose sets," *Recursive Function Theory: Newsletter* **36** (Feb. 1987), 13-14.

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

[F79] S. Fortune, "A note on sparse complete sets," *SIAM J. on Comput.* **8** (1979), 431-433.

[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 (June, 1987), 50-60.

[HH87] 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.

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

[J68] C. Jockusch, Jr., "Semirecursive sets and positive reducibility," *Transactions of the AMS* **131** (1968), 420-436.

[K83] K. Ko, "On self-reducibility and weak P-selectivity," *Journal of Computer and System Sciences* **26** (1983), 209-221.

[K87] K. Ko, "On helping by robust oracle machines," *Proceedings of the Second Annual Structure in Complexity Conference,* IEEE Computer Society (June, 1987), 182-190.

[M82] S. Mahaney, "Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis," *J. Comput. Systems Sci.* **25** (1982), 130-143.

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

[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.