Bi-immunity Results for Cheatable Sets

Richard Beigel*
Department of Computer Science
The Johns Hopkins University
Baltimore, MD 21218

January 5, 1995

Abstract

An oracle A is k-cheatable if there is a polynomial-time algorithm to determine the answers to 2^k parallel queries to A from the answers to only k queries to some other oracle B. It is known that 1-cheatable sets cannot be bi-immune for P. In contrast, we construct 2-cheatable sets that are bi-immune for arbitrary time complexity classes. In addition, for each k, we construct a set that is $(k+1)$-cheatable, but not k-cheatable; we show that this separation does not hold with bi-immunity. We show that if a recursive set A is bi-immune for P then there exists an infinite 1-cheatable set that is polynomial-time m-reducible to A. Consequently if NP contains a set that is bi-immune for P then NP contains a set that is not polynomial-time Turing-equivalent to a self-reducible set.

1. Introduction

Complexity theory deals with how hard problems are. Time, space, and alternation have served as measures of difficulty. Recently, researchers have

*Research supported by a Fannie and John Hertz Foundation fellowship and by NSF grant CCR-8808949. Part of this work was done while the author was a graduate student at Stanford University.

looked at the number of queries that must be made to an oracle by a polynomial time algorithm that solves a problem, as a measure of that problem's difficulty [15, 24]. When we consider number-of-queries as a complexity measure, it is natural to consider the "complexity classes" of functions and sets induced by that measure. We call these complexity classes *bounded query classes*, because they contain sets and functions computable by algorithms that make a bounded number of queries to an oracle. Since number-of-queries is not a complexity measure in the sense of Blum, many expected results about complexity measures do not apply to bounded query classes. Several papers [1, 2, 9, 7, 4, 5, **?**] investigate the relationships among different bounded query classes. One basic question is "When do k queries to an oracle enable us to compute more functions in polynomial time than only $k-1$ queries?"

The basic question in the preceding paragraph has at least sixteen interpretations. Queries may be made in series as in a Turing reduction (serial queries have also been called "adaptive" [14] because each query is allowed to depend on the answers to the previous queries), or in parallel as in a truth-table reduction. It may be required that the $k-1$ queries be made to the same oracle as the k queries, or it may be permitted that the $k-1$ queries be made to a different oracle. The set of $k-1$ query strings may be required to be a subset of the set of k query strings, or the set of $k-1$ query strings may be allowed to be any $(k-1)$-element subset of Σ^*.

A set A is defined to be *k-query p-terse* if k parallel queries to A allow us to compute more functions than only $k-1$ serial queries to A. A set A is defined to be *k-query p-superterse* if k parallel queries to A allow us to compute more functions than only $k-1$ serial queries to B for every set B. The reason for defining p-superterseness is that most proofs of p-terseness are in fact proofs of p-superterseness [2, 9, 7, 4, 5]; in addition k-query p-superterseness is the strongest of our sixteen ways of stating that k queries to an oracle enable us to compute more functions in polynomial time than only $k-1$ queries. A special case is 2-query p-terseness; it is known that a set A is 2-query p-terse if and only if A is 2-query p-superterse [9].

A set A is defined to be *k-cheatable* if there exists a set B such that 2^k parallel queries to A *do not* allow us to compute more functions than only k serial queries to B. For large k, k-cheatability can be seen as a dramatic failure to be $(k+1)$-query p-superterse. In the special case $k=1$, this failure is not so dramatic: a set A is 1-cheatable if and only if A is not

2-query p-superterse. Efforts to extend a number of results about 2-query p-superterseness to general results about k-query p-superterseness have been unsuccessful [2, 9, 7, 4]. However, some of those results can be extended to general results about k-cheatability. We think that results about 2-query p-superterseness are best understood as results about cheatability, not as results about p-superterseness.

A set A is *p-superterse* (*p-terse*) if A is k-query p-superterse (p-terse) for some k. A set A is *cheatable* if A is k-cheatable for some k. A number of papers study the properties of non-p-superterse sets [1, 2, 9, 7, 4, 5, ?]. The class of cheatable sets is a proper subset of the class of non-p-superterse sets. While we do not claim that the class of cheatable sets is more interesting than the general class of non-p-superterse sets, there are a number of interesting results about cheatable sets that do not hold for all non-p-superterse sets or else admit only weaker analogies for general non-p-superterse sets. For example, all cheatable sets are recursive [10]; however non-p-superterse sets can be nonrecursive [10, 11]. Cheatability is preserved by polynomial-time Turing reductions [1]; non-p-superterseness is not [1]. A cheatable set cannot be self-reducible unless it is in P [1]; no corresponding result is known for non-p-superterse sets. All cheatable sets are computable in polynomial time with polynomial advice [1], whereas non-p-superterse sets are only known to be computable in exponential time with polynomial advice [1]. It is known that n parallel queries to a cheatable set can be answered by making only a constant number of queries to some oracle B for all n [9]; a corresponding result says that n parallel queries to a non-p-superterse set A can be answered by making $O(\log n)$ queries to some oracle B [5]. (This allows us to classify every set A based on the asymptotic number of queries to a second oracle required in order to answer n parallel queries to A: 0 queries iff A is polynomial-time computable, $\Theta(1)$ queries iff A is cheatable but not polynomial-time computable, $\Theta(\log n)$ queries iff A is non-p-superterse but not cheatable, n queries iff A is p-superterse.) A cheatable set cannot be NP-hard under polynomial-time Turing reductions unless P = NP [9]; the best known corresponding result says that a non-p-superterse set cannot be NP-hard under polynomial-time truth-table reductions unless P = UP [7].

Some results hold both for cheatable sets and for non-p-superterse sets. Because cheatable sets can have arbitrarily high time complexity [2], no recursive set can be hard for the class of cheatable sets under polynomial-time Turing reductions. Therefore the class of cheatable sets does not contain a polynomial-time Turing complete set, and the class of recursive non-p-superterse sets does not contain a polynomial-time Turing complete set. Since every truth-table degree (no time bound) contains a non-p-superterse set [10], no set can be hard for the class of non-p-superterse sets under polynomial time Turing reductions. Therefore the class of non-p-superterse sets does not contain a polynomial-time Turing complete set.

The relationship of cheatability to completeness, self-reducibility, and NP-hardness has already been established. In this paper we investigate the relationship of cheatability to bi-immunity for P. This question was motivated by the recursion-theoretic Nonspeedup Theorem [10], which states that all "recursively" cheatable sets are recursive; by analogy the Nonspeedup Theorem suggests that cheatable sets might be easy in some standard complexity-theoretic sense. Amir and Gasarch [2] and (later) ourselves [8] have shown that 1-cheatable sets cannot be bi-immune for P, so membership in a 1-cheatable set can be decided in polynomial time infinitely often. In contrast, we show that 2-cheatable sets can be bi-immune for P. This is a qualitative distinction between 1-cheatable sets and the general class of cheatable sets, which was unexpected because the class of "recursively" 1-cheatable sets is equal to the class of "recursively" cheatable sets (both are equal to the class of recursive sets). We construct sets that are $(k+1)$-cheatable but not k-cheatable for each k. We show that if a recursive set L is bi-immune for P then L contains a 1-cheatable subset that is not in P but is polynomial-time m-reducible to L. This result has interesting consequences: it shows that the separation of $(k+1)$-cheatable from k-cheatable does not hold with bi-immunity, and it provides a plausible condition under which NP intersects a non-self-reducible polynomial-time Turing degree (partially answering a question of Selman [28]).

2. Bounded Query Reductions

We say that n queries to an oracle are made *in parallel* if a list of all n queries is formed before any of them is made. Otherwise we say that n queries are made *in series*, or simply that n queries are made. The difference is that computation is allowed between serial queries to an oracle — the answer to a prior query may determine what query is to be made next.

Definition 1

- A set A is *polynomial-time k-Turing reducible* to B ($A \leq^P_{k\text{-T}} B$) if there is a polynomial-time algorithm relative to B that determines whether x is in A by making only k queries to B.
- A set A is *polynomial-time k-truth-table reducible* to B ($A \leq^P_{k\text{-tt}} B$) if there is a polynomial-time algorithm relative to B that determines whether x is in A by making only k parallel queries to B.

The definitions above are equivalent to definitions made by Book and Ko [14] and Ladner, Lynch, and Selman [25]. Note that our definitions of polynomial-time k-Turing reducibility and polynomial time k-truth-table reducibility make sense if A or B or both are replaced by polynomial length-bounded functions, instead of oracles.

The function F^A_k defined below is a convenient notation for the results of k parallel queries to A.

Definition 2 $\mathrm{F}^A_k(x_1, \ldots, x_k) = \langle \chi_A(x_1), \ldots, \chi_A(x_k) \rangle$, where χ_A is the characteristic function of A.

The function F^A_k is important because it is 1-Turing complete for the class of functions that are polynomial-time k-truth-table reducible to A. In fact, $f \leq^P_{k\text{-tt}} A$ if and only if $f \leq^P_{1\text{-T}} \mathrm{F}^A_k$.

Definition 3

- A set A is *k-query p-terse* if $\mathrm{F}^A_k \not\leq^P_{(k-1)\text{-T}} A$
- A set A is *p-terse* if A is k-query p-terse for all k.
- A set A is *k-query p-superterse* if $(\forall B)[\mathrm{F}^A_k \not\leq^P_{(k-1)\text{-T}} B]$.
- A set A is *p-superterse* if A is k-query p-superterse for all k.
- A set A is *k-cheatable* if $(\exists B)[\mathrm{F}^A_{2^k} \leq^P_{k\text{-T}} B]$.
- A set A is *cheatable* if A is k-cheatable for some k.

Although we defined two different brands of k-query p-terseness (super and regular), we do not define both analogous brands of k-cheatability because they lead to equivalent definitions of cheatability.

We will need the following obvious result, which is proved in [8].

Theorem 4 *If* $\mathrm{F}^A_{(k+1)} \leq^P_{k\text{-tt}} A$ *then*

i. $(\forall n)[\mathrm{F}^A_n \leq^P_{k\text{-tt}} A)]$.

ii. A *is* k*-cheatable.*

See [10, 11] for some results on bounded-query reductions in recursion theory.

3. An Oracle-Free Definition of Cheatability

The concepts of k-query p-superterseness and k-cheatability can be described in a purely combinatorial fashion without recourse to oracles. In order to do so we present the concept of computability by a set of k polynomial-time computable functions. (The results from this section will be useful in Section 6..)

Definition 5 The total function h is *computable by a set of* k *polynomial-time computable functions* if there exist k polynomial time computable functions $g_1, \ldots, g_k$ such that

$$(\forall x)[h(x) \in \{g_i(x) : 1 \leq i \leq k\}].$$

Equivalently, the function h is computable by a set of k polynomial-time computable functions if, for each x, we can compute in polynomial time a length-k list that includes $h(x)$. When h is computable by a set of k polynomial-time computable functions, we say informally that *there are only* k *possible values for* $h(x)$.

The following theorem [5] provides an equivalence between bounded-query reducibility to an oracle and computability by a set of polynomial-time computable functions.

Theorem 6

i. *If* $(\exists B)[h \leq^P_{k\text{-T}} B]$ *then* h *is computable by a set of* 2^k *polynomial-time computable functions.*

ii. *If* h *is computable by a set of* 2^k *polynomial-time computable functions then* $(\exists B)[h \leq^P_{k\text{-tt}} B]$.

This theorem immediately provides the following necessary and sufficient conditions for p-superterseness and cheatability:

Theorem 7

- *A is k-query p-superterse iff* F_k^A *is not computable by a set of* 2^{k-1} *polynomial-time computable functions.*
- *A is p-superterse iff* F_k^A *is not computable by a set of* 2^{k-1} *polynomial-time computable functions for any value of* k*.*
- *A is k-cheatable iff* $\mathrm{F}_{2^k}^A$ *is computable by a set of* 2^k *polynomial-time computable functions.*
- *A is cheatable iff* $\mathrm{F}_{2^k}^A$ *is computable by a set of* 2^k *polynomial-time computable functions for some* k*.*

In [9] we prove the following:

Theorem 8 *If* $\mathrm{F}_{2^k}^A \leq_{k\text{-}\mathrm{T}}^P B$ *then*

i. for every $n \geq 2^k$*, any* n *parallel queries to* A *can be answered by a polynomial time algorithm that asks only* $2^k - 1$ *of the same queries in parallel.*

ii. $(\forall n)[\mathrm{F}_n^A \leq_{k\text{-}\mathrm{T}}^P B]$.

The name cheatable is motivated by part (i) of this theorem, which states that if A is cheatable then any n queries to A can be answered by a polynomial time algorithm that asks only a fixed number (independent of n) of the same questions. If the answers to a true-false test are given by a cheatable set, then a student up to no good would only need to copy a fixed number of answers in order to determine them all. The theorem provides the following necessary and sufficient conditions for cheatability:

Theorem 9 *A set A is cheatable if and only if there exists a constant k such that for all* n*,* F_n^A *is computable by a set of* k *polynomial-time computable functions.*

4. Are Cheatable Sets Easy Infinitely Often?

If $\mathrm{F}_{2^k}^A$ can be computed by asking k queries to some other oracle within unbounded computation time (*i.e.*, $(\exists B)[\mathrm{F}_{2^k}^A \leq_{k\text{-T}} B]$), then we might say that A is recursively cheatable. However, in [10] we prove the Nonspeedup Theorem, which states that A is "recursively cheatable" in this sense if and only if A is recursive. By analogy, we might conjecture that all cheatable sets are in P. However, Amir and Gasarch [2] have constructed 1-cheatable sets A of arbitrarily great time complexity. Therefore we seek to prove that cheatable sets are easy in some sense that is weaker than polynomial-time computability. In [1], it is proved that all cheatable sets have small circuits. In this section we investigate whether cheatable sets must be easy infinitely often, *i.e.*, whether they cannot be bi-immune for P.

In [3], Balcazar and Schöning formalized the notion of being easy infinitely often, which was previously considered by Berman and Hartmanis [13] and by Rabin [26].

Definition 10

- A set A is *immune* for a class $\mathcal{C}$ if A contains no infinite subset that belongs to $\mathcal{C}$.
- A set A is *co-immune* for a class $\mathcal{C}$ if $\bar{A}$ is immune for $\mathcal{C}$.
- A set A is *bi-immune* for a class $\mathcal{C}$ if A is immune for $\mathcal{C}$ and co-immune for $\mathcal{C}$.

The authors of [3, 13, 26] have noted that a set A is easy infinitely often if and only if A is not bi-immune for P. Amir and Gasarch [2] have found an elegant proof that no 1-cheatable set is bi-immune for P; a constructive proof of that fact can be found in [8]. In contrast to the theorem that no 1-cheatable set can be bi-immune for P, we construct a 2-cheatable set that is bi-immune for P.

Lemma 11 *There exists a set A over the alphabet $\{0\}$ such that A is bi-immune for* P *and such that any three parallel queries to A can be answered in polynomial time by making only two of the same queries in parallel.*

Proof: Define $\mathrm{tow}(0) = 1$ and $\mathrm{tow}(i+1) = 2^{\mathrm{tow}(i)}$. Define $\log^{(0)} x = x$ and $\log^{(i+1)} x = \log\log^{(i)} x$. Define $\log^* x$ to be the greatest i such that $\log^{(i)} x \geq 1$.

By the extended time hierarchy theorem [17], there is a set B in $\mathrm{DTIME}(17^{2^n})$ that is bi-immune for $\mathrm{DTIME}(3^{2^n})$. Let

$$A = \{0^x : 0^{\mathrm{tow}(\log^* x)} \in B\}.$$

Then A is produced by dividing 0^* into blocks such that the block beginning with i contains $2^i - i$ elements; every element of a block belongs to A if and only if the first element of that block belongs to B.

As in the construction of a hard 1-cheatable set [2], we can answer three parallel queries to A by asking only two of them. Proof: If two of the queries belong to the same block, then we ask one of the queries in each block, and the answer to the remaining query is implied. Otherwise the queries belong to three distinct blocks. Assume that the shortest query belongs to the block that starts with 0^z. The length of the longest query must be at least 2^{2^z}. We can directly compute the answer to the smallest query in time $17^{2^z} = (2^{2^z})^{17/2}$, which is a polynomial in the input length. We ask the oracle the answers to the two larger queries.

If A has an infinite $O(n^k)$ time subset S, then we let

$$T = \{0^{\mathrm{tow}(\log^* x)} : 0^x \in S\}.$$

Therefore T is an infinite subset of B. We can determine whether $0^y \in T$ by checking at most $2^y - y$ candidates for x in the definition of T; the length of each candidate is at most 2^y. Thus we can test whether $y \in T$ in time $(2^y - y)(2^y)^k \leq 2^{(k+1)y} = O(3^{2^y})$. Thus $T \in \mathrm{DTIME}(3^{2^n})$, which contradicts the immunity of B. We obtain a similar contradiction if $\bar{A}$ has an infinite $O(n^k)$ time subset. Thus A is bi-immune for P. ∎

By Theorem 4.ii, it follows that there exists a set A over the alphabet $\{0\}$ that is 2-cheatable and bi-immune for P. The existence of a 2-cheatable set A over an arbitrary alphabet such that A is bi-immune for P will follow from the next lemma. (The proof is a straightforward, but notationally cumbersome, modification of the preceding proof.)

Lemma 12 *Let h be a total recursive function, and let Σ be an alphabet. Then there exists a set A over the alphabet Σ such that A is bi-immune for*

DTIME($h(n)$), *and such that any three parallel queries to A can be answered by making only two of the same queries in parallel.*

Proof: Let g be a strictly increasing, time-constructible function that is at least $|\Sigma|^n h(n)$. We will begin by constructing a set A over the alphabet $\{0\}$ such that A is 2-cheatable and bi-immune for DTIME($g(n)$). Define $g^{(0)}(n) = n$ and $g^{(i+1)}(n) = g(g^{(i)}(n))$. Define $g^{(-*)}(n)$ to be the greatest i such that $g^{(i)}(1) \leq n$.

By the extended time hierarchy theorem [17], let B be a set in DTIME($(g(g(n)))^4$) that is bi-immune for DTIME($(g(g(n)))^2$). Let

$$A = \{0^x : 0^{g^{(g^{(-*)}(x))}(1)} \in B\}.$$

Then A is produced by dividing 0^* into blocks such that the block beginning with 0^i contains $g(i) - i$ elements; every element of a block belongs to A if and only if the first element of that block belongs to B.

As in [2], we can answer three parallel queries to A in polynomial time by asking only two of them. Proof: If two of the queries belong to the same block, then we ask one of the queries in each block, and the answer to the remaining query is implied. Otherwise the queries belong to three distinct blocks. Assume that the shortest query belongs to the block that starts with 0^z. The length of the longest query must be at least $g(g(z))$. We can directly compute the answer to the smallest query in time $(g(g(z)))^4$, which is a polynomial in the input length. We ask the oracle the answers to the two larger queries.

If A has an infinite subset $S \in$ DTIME($g(n)$), then we let

$$T = \{0^{g^{(g^{(-*)}(x))}(1)} : 0^x \in S\}.$$

Therefore T is an infinite subset of B. We can determine whether $0^y \in T$ by checking at most $g(y) - y$ candidates for x in the definition of T; the length of each candidate is at most $g(y)$. Thus we can test whether $y \in T$ in time $(g(y) - y)g(g(y)) \leq (g(g(y)))^2$. Thus $T \in$ DTIME($(g(g(y)))^2$), which contradicts the immunity of B. We obtain a similar contradiction if $\bar{A}$ has an infinite subset in DTIME($g(n)$). Thus A is bi-immune for DTIME($g(n)$).

So far we have constructed a set A over the alphabet $\{0\}$ such that A is bi-immune for DTIME($g(n)$), and such that any three parallel queries to A

can be answered by making only two of the same queries in parallel. In order to extend this result to an arbitrary alphabet Σ, let $A' = \{s : 0^{|s|} \in A\}$. Obviously, any three parallel queries to A' can be answered by making only two of the same queries in parallel. If A' has an infinite DTIME$(f(n))$ subset then a straightforward argument shows that A has an infinite DTIME$(|\Sigma|^n f(n))$ subset. Thus A' is immune for DTIME$(g(n)/|\Sigma|^n)$, which is equal to DTIME$(h(n))$. Similarly, A' is co-immune for DTIME$(h(n))$, so A' is bi-immune for DTIME$(h(n))$. ∎

Theorem 13 *Let h be a total recursive function, and let Σ be an alphabet. Then there exists a 2-cheatable set A over the alphabet Σ such that A is bi-immune for* DTIME$(h(n))$.

Proof: This follows from the preceding Lemma and Theorem 4.ii. ∎

The extended time hierarchy theorem is a very powerful tool. It is interesting to note that we did not really need to use it in order to prove the preceding results, because the techniques of Berman and Hartmanis [13] and Balcazar and Schöning [3] can be easily extended to show that if $p(n)$ is time constructible and $p(n) \geq 2^n q(n) \log(q(n))$ then there is a set B in DTIME$(p(n))$ that is bi-immune for DTIME$(q(n))$. By letting $p(n) = (g(g(n)))^4$ and $q(n) = (g(g(n)))^2$, it follows that if $g(n) \geq 2^n$ then there is a set in DTIME$((g(g(n)))^4)$ that is bi-immune for DTIME$((g(g(n)))^2)$. This enables the proof above to go through without the extended time hierarchy theorem, whenever $g(n) \geq 2^n$. The theorem follows for smaller functions, because if a language is bi-immune for DTIME(2^n) then it must be bi-immune for all subsets of DTIME(2^n). Recently, Goldsmith, Joseph, and Young [18] have independently discovered another proof of our result.

The next corollary implies the existence of 2-cheatable sets that are not 1-cheatable.

Corollary 14 *There exists a set A such that*

$$(\forall n \geq 2)[\mathrm{F}_n^A \leq^P_{2\text{-tt}} A \textit{ but } (\forall B)[\mathrm{F}_n^A \not\leq^P_{1\text{-tt}} B]].$$

Proof: Choose a recursive function h that dominates all polynomials, and let A be as in Lemma 12. The set A is constructed so that $\mathrm{F}_3^A \leq_{2\text{-tt}}^P A$. By Theorem 4.i,

$$(\forall n)[\mathrm{F}_n^A \leq_{2\text{-tt}}^P A].$$

Since A is bi-immune for P, A cannot be 1-cheatable so

$$(\forall n \geq 2)(\forall B)[\mathrm{F}_n^A \not\leq_{1\text{-tt}}^P B].$$

∎

5. Recursive Bi-immune Sets Have Hard 1-Cheatable Subsets

In the preceding section we saw that 1-cheatable sets cannot be bi-immune for P. In this section we show that if A is recursive then $\bar{A}$ has an infinite polynomial-time computable subset or A has an infinite 1-cheatable subset. Thus a recursive set cannot be co-immune for P and immune for 1-cheatable. Consequently if a recursive set is bi-immune for P then it contains a 1-cheatable subset that is not in P.

Lemma 15 *Let A be a recursive set.*

i. There is an infinite set $S \in \mathrm{P}$ such that $S \cap A$ is an infinite 1-cheatable set or $S \cap \bar{A}$ is an infinite polynomial-time computable set.

ii. There exist sets B and B' such that

- *$B \subseteq A$ and $B' \subseteq \bar{A}$.*
- *B and B' are 1-cheatable.*
- *$B \leq_m^P A$.*
- *B is infinite or B' is infinite.*

iii. A or $\bar{A}$ contains an infinite 1-cheatable subset.

Proof:

i. Let M be a Turing machine that accepts A. Let $f(n)$ be the running time of M on input 0^n. Let $T(n)$ be a fully time-constructible function that is at least $\max(2^n, f(n))$. (For example, we could let $T(n)$ be the running time of a Turing machine M_T which, on input of length n, computes $\max(2^n, f(n))$ in unary.) Let $T^{(k)}$ be the composition of T with itself k times. Let

$$S = \{0^{T^{(k)}(1)} : k \geq 0\}.$$

Then S is infinite. It is obvious that S is polynomial-time computable. We show that $S \cap A$ is 1-cheatable. Given strings x and y, we can determine in polynomial time whether x and y are in S. If $x \notin S$, then we make one query to $S \cap A$ to determine if $y \in S \cap A$, and we know that $x \notin S \cap A$. Similarly if $y \notin S$. If $x \in S$ and $y \in S$, then $x = y$ or $|x| \geq T(|y|)$ or $|y| \geq T(|x|)$. In the first case, only one query to $S \cap A$ is needed. In the second and third cases, we query $S \cap A$ about the larger query, and we run M on the smaller query, using time that is linear in the length of the input. Thus $S \cap A$ is 1-cheatable. Similarly $S \cap \bar{A}$ is 1-cheatable.

If $S \cap A$ is infinite then $S \cap A$ is an infinite 1-cheatable subset of A. If $S \cap A$ is finite, then $S \cap \bar{A}$ differs from S on only finitely many strings, so $S \cap \bar{A}$ is an infinite polynomial-time computable subset of $\bar{A}$.

ii. Let $B = S \cap A$ and let $B' = S \cap \bar{A}$.

iii. B and B' are 1-cheatable subsets of A. B or B' is infinite.

∎

Consequently we obtain Amir and Gasarch's [2] result:

Corollary 16 *For any total recursive function T, there exists a 1-cheatable set that is not in* DTIME($T(n)$).

Proof: By the extended time hierarchy theorem [17], there exists a recursive set A that is bi-immune for DTIME($T(n)$). By Lemma 15.iii, we can let B be an infinite 1-cheatable subset of A or $\bar{A}$. The bi-immunity of A guarantees that B is not in DTIME($T(n)$). ∎

Lemma 17 *Let A be a recursive set that is bi-immune for* P*.*

i. There exists a set $S \in \mathrm{P}$ such that $S \cap A$ is 1-cheatable, but not in P*.*

ii. There exists a set B such that

- *$B \subseteq A$.*
- *B is 1-cheatable.*
- *$B \leq_m^P A$*
- *$B \notin \mathrm{P}$.*

Proof: Let A be a recursive set that is bi-immune for P.

i. By Lemma 15.i, there is an infinite set $S \in \mathrm{P}$ such that $S \cap A$ is an infinite 1-cheatable set or $S \cap \bar{A}$ is an infinite polynomial-time computable set. Because A is co-immune for P, the second possibility is ruled out, so $S \cap A$ is an infinite 1-cheatable set. Because A is immune for P, the set $S \cap A$ cannot be in P.

ii. Let $B = S \cap A$.

▮

In [27], Schnorr defined self-reducibility.

Definition 18 A set A is self-reducible if there is a polynomial time bounded oracle Turing machine M such that

- All strings queried by M are strictly shorter than the input string.
- The language accepted by M^A (machine M computing with oracle A) is A.

Since SAT (the set of satisfiable Boolean formulas) is self-reducible, every NP-complete set is polynomial-time m-equivalent to a self-reducible set. A natural question is whether every set in NP is polynomial-time Turing equivalent to a self-reducible set. (In [28], Selman posed that question for d-self-reducibility, which is a special case of self-reducibility.)

Theorem 19 *If A is a recursive set that is bi-immune for* P *then there exists a set B such that* *i.* $B \notin \mathrm{P}$.

ii. $B \leq_m^P A$.

iii. $A \not\leq_T^P B$.

iv. *If* S *is self-reducible and* $S \leq_T^P B$ *then* $S \in \mathrm{P}$.

v. *If* S *is self-reducible then* $B \not\equiv_T^P S$.

Proof:

i,ii. Let A be a recursive set that is bi-immune for P. By Lemma 17.ii, there exists a set B such that B is 1-cheatable, $B \leq_m^P A$ and $B \notin \mathrm{P}$.

iii. Let S be any set such that $S \leq_T^P B$. Then S is 1-cheatable [1]. However if S is 1-cheatable then S is not bi-immune for P. Therefore $S \neq A$.

iv. If S is self-reducible and 1-cheatable then $S \in \mathrm{P}$ [1].

v. If $S \in \mathrm{P}$ and $B \notin P$ then $B \not\leq_T^P S$.

∎

This theorem enables us to show that if NP contains a language that is bi-immune for P then there is a non-self-reducible NP p-degree.

Theorem 20 *If* NP *contains a set that is bi-immune for* P *then*

i. $\mathrm{NP} - \mathrm{P}$ *contains a 1-cheatable set.*

ii. $\mathrm{NP} - \mathrm{P}$ *contains a set* B *such that every self-reducible set that is polynomial-time Turing-reducible to* B *is in* P.

iii. $\mathrm{NP} - \mathrm{P}$ *contains a set* B *that is not polynomial-time Turing-equivalent to any self-reducible set.*

Proof: Let A be a set in NP that is bi-immune for P.

i. By Lemma 17.ii, there exists a set B such that $B \leq_m^P A$, $B \notin \mathrm{P}$ and B is 1-cheatable. The first two conditions imply that $B \in \mathrm{NP} - \mathrm{P}$.

ii. By Theorem 19.i,ii,iv there exists a set B such that $B \leq_m^P A$, $B \notin$ P, and every self-reducible set that is polynomial-time Turing-reducible to B is in P.

iii. By Theorem 19.i,ii,v there exists a set B such that $B \leq_m^P A$, $B \notin$ P, and B is not polynomial-time Turing-equivalent to any self-reducible set.

$\blacksquare$

Since Bennett and Gill [12] have shown that NP contains a language that is bi-immune for P under almost all relativizations, our assumption that NP contains a language that is bi-immune for P is plausible. (Since Homer and Maass [21] have constructed a relativized world in which P $\neq$ NP but no language in NP is bi-immune for P, we do not expect a proof of our assumption to be forthcoming.) We think that the conclusions in this theorem are suggestive of the likely behavior of sets in NP. Parts (ii) and (iii) of this theorem partially answer Selman's question in [28]; Ko [23] has also made some progress on Selman's question.

By closely examining the proof of the preceding theorem, we can obtain a slightly stronger, but more complicated result. Let A be a set in NP that is bi-immune for P. Then $A \in \mathrm{DTIME}(2^n)$. As in the proof of Lemma 15.i, we define a set $S = \{0^{\mathrm{tow}(n)} : n \geq 0\}$. Either $S \cap A$ is an infinite 1-cheatable subset of A, or $S \cap \bar{A}$ is an infinite polynomial-time computable subset of $\bar{A}$. Since A is bi-immune for P, we conclude that $B = S \cap A$ is an infinite 1-cheatable set in NP. However, instead of assuming that NP contains a language that is bi-immune for P, it would have been sufficient to assume directly that the set $S = \{0^{\mathrm{tow}(n)} : n \geq 0\}$ contains a subset B that is in NP $-$ P. As in [20], this is easily seen to be equivalent to assuming that there is a tally set in $\cup_{k\geq 0}\mathrm{NTIME}((\mathrm{tow}(n))^k) - \cup_{k\geq 0}\mathrm{DTIME}((\mathrm{tow}(n))^k)$. (Consider $\{0^n : 0^{\mathrm{tow}(n)} \in S\}$.) That is equivalent to assuming that $\cup_{k\geq 0}\mathrm{NTIME}(2^{(\mathrm{tow}(n))^k}) \neq \cup_{k\geq 0}\mathrm{DTIME}(2^{(\mathrm{tow}(n))^k})$. Some results like this were obtained independently in [19].

Using the techniques of this section, we have obtained similar results for near-testable sets [6].

6. A Hierarchy of Cheatable Sets

By Corollary 14, there exists a 2-cheatable set that is not 1-cheatable. The following theorem uses diagonalization to extend this result.

Theorem 21 *If $k \geq 1$ then there exists a k-cheatable set that is k-query p-superterse.*

Proof: We construct a set A such that A is k-cheatable, but F_k^A is not computed by a set of $(2^k - 1)$ polynomial-time computable functions. The second condition implies that A is k-query p-superterse, by Theorem 7.i. Let M_i be the ith Turing machine that computes a set of $(2^k - 1)$ polynomial-time computable functions[1] (in a canonical enumeration of such machines). Let $w_{j,n} = 0^j 1^{n-j}$.

Stage 0:	$n := k$;
	$A := \emptyset$.
Stage $i > 0$:	$n := 2^n$;
	for all strings s whose length is less than n do
1:	if $\chi_A(s)$ has not yet been defined then $\chi_A(s) := 0$;
	simulate M_i for 2^n steps on input $(w_{1,n}, w_{2,n}, \ldots, w_{k,n})$;
2:	define $\mathrm{F}_k^A(w_{1,n}, w_{2,n}, \ldots, w_{k,n})$ so as to make all of M_i's answers wrong.

For each i there are infinitely many Turing machines whose behavior is identical to that of M_i (by standard padding techniques). Since 2^n dominates every polynomial, the simulation in the construction above allows some machine that is equivalent to M_i to run to completion. The construction defeats that machine, and hence defeats M_i. Therefore F_k^A is not computed by any set of $(2^k - 1)$ polynomial-time computable functions.

We say that the string s is used in the diagonalization if $\chi_A(s)$ is defined at line 2, rather than at line 1. The string $s = w_{j,n}$ is used in the diagonalization if and only if n is a tower of 2's and j is between 1 and k. Thus we may determine in polynomial time whether s is used in the diagonalization.

[1]The particular machine model is not important here. For concreteness we can assume that the TM has $2^k - 1$ different output tapes on which to print the values of the $2^k - 1$ polynomial-time computable functions. Since k is a constant, the TM can easily be made to run in polynomial time, by evaluating one polynomial-time computable function at a time.

To show that A is k-cheatable it suffices to show that $\mathrm{F}^A_{k+1} \leq^P_{k\text{-tt}} A$ (by Theorem 4.ii). If one of the $k+1$ input strings s is not used in the diagonalization then $s \notin A$, so we use our k parallel queries to determine the membership of the other k strings.

If all $k+1$ input strings are used in the diagonalization then one of their lengths is logarithmic in the length of the longest string. Let s be such a string. We determine whether $s \in A$ by running the construction at stage $\log^*(|s|)$. The simulation of M_i dominates the running time of the stage. The simulation can be performed in $O(4^{|s|})$ steps [22]. This time is linear in the length of the input. We use our k parallel queries to determine the membership of the other k strings. ∎

We immediately obtain the following:

Corollary 22 *Let $k \geq 1$. Then*

i. There exists a set that is k-cheatable but not $(k-1)$-cheatable.

ii. There exists a set that is k-query p-superterse but not $(k+1)$-query p-terse.

iii. There exists a set A such that

$$(\forall n \geq k)[\mathrm{F}^A_n \leq^P_{k\text{-tt}} A \textit{ but } (\forall B)[\mathrm{F}^A_n \not\leq_{(k-1)\text{-T}} B]].$$

Proof:

i. Follows from the preceding theorem and the definitions.

ii. Follows from the preceding theorem and the definitions.

iii. Follows from the proof of the preceding theorem and Theorem 4.i.

∎

Since some other separation results in complexity theory hold with bi-immunity [3, 16, 17], we might wonder if Corollary 22.i holds with bi-immunity. In other words, does there exists a k-cheatable set A such that neither A nor $\bar{A}$ contains a $(k-1)$-cheatable subset? To the contrary, we show that A or $\bar{A}$ must contain a 1-cheatable subset.

Theorem 23 *If A is k-cheatable then A or $\bar{A}$ contains an infinite 1-cheatable subset.*

Proof: By the Nonspeedup Theorem [10], all k-cheatable sets are recursive. By Lemma 15.iii, if A is recursive then A or $\bar{A}$ contains an infinite 1-cheatable subset. ∎

Corollary 24 *If A is k-cheatable (where $k \geq 1$) then A or $\bar{A}$ contains an infinite $(k-1)$-cheatable subset.*

Proof: The result for $k \geq 2$ is established in the preceding theorem. The result for $k = 1$ follows because no 1-cheatable set is bi-immune for P. ∎

7. Discussion

We have stated and proved the results in this paper without referring explicitly to the bounded query classes $\mathrm{FQ}(k, A)$ and $\mathrm{FQ}_{\|}(k, A)$ defined in many related papers like [2, 5]. Since we do not think that the results in those papers could have been expressed as succinctly without reference to bounded query classes, we feel lucky that the notation $\leq^A_{k\text{-tt}}$, $\leq^A_{k\text{-T}}$, and F^A_k sufficed in order to express all of the results in this paper. We hope that by restraining our use of unfamiliar notation we have made it easier for the readers of this paper to familiarize themselves with the concepts of p-superterseness and cheatability.

8. Acknowledgments

The author expresses his warmest appreciation to his thesis advisor, John Gill, who supervised some of this work. The author wishes to thank Judy Goldsmith for helpful discussions of bi-immune sets; Alan Selman for helpful discussions of self-reducible sets; Bill Gasarch and Amir Amihood for pointing out that our original proof of Lemma 12 generalized to arbitrarily large time bounds, and also for suggesting that we simplify the proof by establishing it first for sets over the alphabet $\{0\}$; and Cathy Schevon and Dwight Wilson for reading early drafts of this paper. The author also wishes to thank the referees for suggesting that the introduction review the contrasts between cheatability and non-p-superterseness, and the relationships between cheatability and other notions in complexity theory. Finally, the author wishes to thank the editor, Ron Book, for suggesting the use of more standard notation.

References

[1] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and nonuniform complexity. In *Proceedings of the 5th Annual Conference on Structure in Complexity Theory*, pages 232–243, 1990.

[2] A. Amir and W. I. Gasarch. Polynomial terse sets. *Inf. & Comp.*, 77:37–56, Apr. 1988.

[3] J. L. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *MST*, 18:1–10, 1985.

[4] R. Beigel. SAT^A is terse with probability 1. TR 87-04, The Johns Hopkins University, Dept. of Computer Science, 1987.

[5] R. Beigel. A structural theorem that depends quantitatively on the complexity of SAT. In *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory*, pages 28–32. IEEE Computer Society Press, June 1987.

[6] R. Beigel. A note on some open problems of goldsmith, joseph, and young. 1988.

[7] R. Beigel. NP-hard sets are p-superterse unless R = NP. TR 88-04, The Johns Hopkins University, Dept. of Computer Science, 1988.

[8] R. Beigel. Query-limited reducibilities. STAN-CS 88-1221, Stanford University, Dept. of Computer Science, July 1988. Ph.D. Thesis.

[9] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, July 1991.

[10] R. Beigel, W. I. Gasarch, J. T. Gill, and J. C. Owings. Terse, superterse, and verbose sets. *Inf. & Comp.*, 103:68–85, 1993.

[11] R. Beigel, W. I. Gasarch, and L. Hay. Bounded query classes and the difference hierarchy. *Archive for Mathematical Logic*, 29(2):69–84, Dec. 1989.

[12] C. H. Bennett and J. Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq$ co-NP^A with probability 1. *SICOMP*, 10:96–112, Feb. 1981.

[13] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SICOMP*, 6:305–322, 1977.

[14] R. V. Book and K.-I. Ko. On sets truth-table reducible to sparse sets. *SICOMP*, 17:903–919, 1988.

[15] W. I. Gasarch. The complexity of optimization functions. TR 1652, University of Maryland, Dept. of Computer Science, 1985.

[16] W. I. Gasarch. Oracles for deterministic versus alternating classes. *SICOMP*, 16:613–627, 1987.

[17] J. Geske, D. Huynh, and A. Selman. A hierarchy theorem for almost everywhere complex sets with application to polynomial complexity degrees. In *Proceedings of the 4th Symposium on Theoretical Aspects of Computer Science*, pages 125–135. Springer-Verlag, 1987. *Vol. 247 of Lecture Notes in Computer Science.*

[18] J. Goldsmith, D. Joseph, and P. Young. A note on bi-immunity and p-closeness of p-cheatable sets in P/poly. *JCSS.* To appear.

[19] J. Goldsmith, D. Joseph, and P. Young. Using self-reducibilities to characterize polynomial time. TR 87-11-11, Dept. of Computer Science, University of Washington, Seattle, Nov. 1987. To appear in *Information and Computation.*

[20] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP − P: Exptime verse nexptime. *Inf. & Control*, 65:158–181, 1985.

[21] S. Homer and W. Maass. Oracle dependent properties of the lattice of NP sets. *Theoretical Computer Science*, 24:279–289, 1983.

[22] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[23] K.-I. Ko. On helping by robust oracle machines. *Theoretical Computer Science*, 52:15–36, 1987.

[24] M. W. Krentel. The complexity of optimization problems. *JCSS*, 36(3):490–509, 1988.

[25] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[26] M. O. Rabin. Degree of difficulty of computing a function and a partial ordering of recursive sets. TR 2, The Hebrew University, Jerusalem, 1960.

[27] C. P. Schnorr. Optimal algorithms for self-reducible problems. In *Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming*, pages 322–337, 1976.

[28] A. L. Selman. Natural self-reducible sets. Tr, Northeastern University, Boston, MA 12115, 1986.