NP-hard Sets are P-Superterse Unless R = NP

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

JHU TR 88-04
Original version August 26, 1988
Re-typeset January 13, 1992

Abstract

A set A is p-terse (p-superterse) if, for all q, it is not possible to answer q queries to A by making only $q-1$ queries to A (any set X). Formally, let $\mathrm{PF}^A_{q\text{-tt}}$ be the class of functions reducible to A via a polynomial-time truth-table reduction of norm q, and let $\mathrm{PF}^A_{q\text{-T}}$ be the class of functions reducible to A via a polynomial-time Turing reduction that makes at most q queries. A set A is p-terse if $\mathrm{PF}^A_{q\text{-tt}} \not\subseteq \mathrm{PF}^A_{(q-1)\text{-T}}$ for all constants q. A is p-superterse if $\mathrm{PF}^A_{q\text{-tt}} \not\subseteq \mathrm{PF}^X_{q\text{-T}}$ for all constants q and sets X.

We show that all NP-hard sets (under $\leq^p_{\mathrm{tt}}$-reductions) are p-superterse, unless it is possible to distinguish uniquely satisfiable formulas from satisfiable formulas in polynomial time. Consequently, all NP-complete sets are p-superterse unless P = UP (one-way functions fail to exist), R = NP (there exist randomized polynomial-time algorithms for all problems in NP), and the polynomial-time hierarchy collapses. This mostly solves the main open question in [4].

Allowing the number of queries q to be a function of the input length n, we obtain nontrivial partial answers to a question posed by Krentel: Namely, for all NP-complete sets A and all sets X,

$$\mathrm{PF}^A_{q(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(q(n)-1)\text{-T}} \qquad \text{unless SAT} \in \mathrm{RTIME}(n^{O(q(n))})$$

This is interesting whenever $q(n) = o(n/\log n)$. As a special case,

$$\mathrm{PF}^A_{\log^i n\text{-tt}} \not\subseteq \mathrm{PF}^X_{(\log^i n-1)\text{-T}} \qquad \text{unless } \mathrm{NTIME}(n^{\mathrm{polylog}\, n}) = \mathrm{RTIME}(n^{\mathrm{polylog}\, n}).$$

*This research was supported in part by grant number CCR-8808949 from the National Science Foundation.

We extend the preceding results to tt-helpers; this allows us to isolate the key properties of NP-complete sets used in the proofs.

Finally, we define pbtt-cylinders, a natural class that includes all $\leq^p_{\mathrm{m}}$-complete sets for NP, PSPACE, and exponential time. For pbtt-cylinders, we show that p-terseness is equivalent to p-superterseness. We also prove a translational lemma for non-p-terse sets; this allows us to extend the preceding result to $\leq^p_{\mathrm{btt}}$-complete sets for NP, PSPACE, and exponential time.

1. Introduction

In [46, 53], Meyer and Stockmeyer defined the polynomial-time hierarchy of sets, which lies between P and PSPACE. NP is at the first level of the hierarchy; P^{NP} is at the second level of the hierarchy; and the entire hierarchy can be defined by bounding the number of alternations of a polynomial-time bounded alternating Turing machine [31].

Recently, researchers have become interested in the fine structure between the first and second levels of the polynomial-time hierarchy (PH). New hierarchies have been defined by bounding the number of oracle queries made by a P^{NP} machine. In [43], Krentel considered the hierarchy of functions computable by a $\mathrm{PF}^{\mathrm{NP}}$ machine with a bounded number of queries. Krentel separated the lower levels of that hierarchy (assuming $\mathrm{P} \neq \mathrm{NP}$), and he proved that many natural optimization problems are complete at different levels. Using different techniques [14], this author obtained Krentel's separation results for the case when the number of queries is a constant.

In related work [27, 28, 29, 42, 50, 57, 60], numerous researchers have considered the hierarchy of sets that are polynomial-time truth-table reducible to an NP-complete set with a bounded number of queries. When the number of queries is a constant (rather than a function of input length), the hierarchy is called the Boolean hierarchy. In [4, 14], Amir, Gasarch, and this author noted relationships between the Boolean hierarchy and the hierarchy of sets accepted by P^{NP} machines with a bounded number of queries. In [39], Kadin separated all levels of the Boolean hierarchy (assuming that the polynomial-time hierarchy is proper). In [59], Wagner extended the separation to the case where the number of queries is a function of the input length.

We define four bounded query hierarchies, which consist of functions and sets that can be computed with a bounded number of queries to an NP oracle, where we distinguish between serial queries (as in a Turing reduction) and parallel queries (as in a truth-table reduction). (In [23] serial queries are called adaptive, and parallel queries are called nonadaptive. For additional background, see [3, 4, 13, 9].)

Henceforth we write n to denote input length. We adopt the following notation from [6]:

Notation 1. Let A be a set, $\mathcal{C}$ a class of sets, and r a reduction.

- P^A_r is the class of all sets that are polynomial-time r-reducible to A.
- PF^A_r is the class of all functions that are polynomial-time r-reducible to A.

- $\mathrm{P}_r^{\mathcal{C}} = \cup_{A \in \mathcal{C}} \mathrm{P}_r^A$.
- $\mathrm{PF}_r^{\mathcal{C}} = \cup_{A \in \mathcal{C}} \mathrm{PF}_r^A$.

We write $q(n)$-tt to denote truth-table reducibility of norm $q(n)$ and $q(n)$-T to denote Turing reducibility with only $q(n)$ queries. (See [6] for definitions of standard concepts in complexity theory.) For example, $\mathrm{P}^A_{q(n)\text{-tt}}$ is the class of all sets that are accepted by a polynomial-time algorithm that makes $q(n)$ parallel queries to A, and $\mathrm{P}^A_{q(n)\text{-T}}$ is the class of all sets that are accepted by a polynomial-time algorithm that makes $q(n)$ serial queries to A.

We are most interested in the classes $\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$, $\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-tt}}$, $\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$, and $\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-tt}}$. Note that if A is NP-complete then $\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}} = \mathrm{PF}^{A}_{q(n)\text{-T}}$, etc. The $\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$ hierarchy has been studied previously by Krentel in [43], because it provides a natural measure of the complexity of NP-hard functions.

Obviously, if $A \equiv_{1\text{-tt}} B$ then $\mathrm{PF}^A_{k\text{-T}} = \mathrm{PF}^B_{k\text{-T}}$, etc. It is also obvious that $\mathrm{PF}^A_{q(n)\text{-T}} \subseteq \mathrm{PF}^A_{(q(n)+1)\text{-T}}$, etc., and that $\mathrm{PF}^A_{q(n)\text{-tt}} \subseteq \mathrm{PF}^A_{q(n)\text{-T}}$, etc.

The following relationships among bounded query classes are known for well-behaved functions $q(n)$:

When q is a constant,

$\mathrm{P}^{\mathrm{NP}}_{q\text{-tt}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{(q+1)\text{-tt}}$	unless NP/poly = co-NP/poly [39].
$\mathrm{P}^{\mathrm{NP}}_{q\text{-T}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{(q+1)\text{-T}}$	unless NP/poly = co-NP/poly [39].
$\mathrm{PF}^{\mathrm{NP}}_{q\text{-tt}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{(q+1)\text{-tt}}$	unless P = NP [14].
$\mathrm{PF}^{\mathrm{NP}}_{q\text{-T}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{(q+1)\text{-T}}$	unless P = NP [14, 43].
$\mathrm{PF}^{\mathrm{NP}}_{q\text{-T}}$	$=$	$\mathrm{PF}^{\mathrm{NP}}_{(2^q-1)\text{-tt}}$	[14].

When $q(n) \le \frac{1}{2}\log n$,

$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-T}}$	unless P = NP [43].

When $q(n) \le (1-\epsilon)\log n$,

$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-T}}$	unless NP/poly = co-NP/poly [59].

When $q(n) = O(\log n)$,

$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{n\text{-T}}$	unless P = NP [43].
$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	$=$	$\mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-1)\text{-tt}}$	[26, 59].

When $q(n) \le n^{1-\epsilon}$,

$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-tt}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}}$	unless NP/poly = co-NP/poly [59].

Note that if NP/poly = co-NP/poly then the polynomial-time hierarchy collapses. Thus for all of the preceding separations it suffices to assume that the PH is proper.

Other work on polynomial-time computations with a bounded number of queries includes [3, 4, 13, 14, 7, 8, 9, 11, 22, 23, 24, 25, 32, 33, 34, 35, 40, 44, 45, 51, 58]. Related work in recursion theory includes [10, 15, 16, 17, 18, 19, 47].

When $2 \leq q(n) \leq \frac{1}{2}\log n$, the preceding results show that, assuming the PH is proper,

$$\begin{array}{rcl} \mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-T}} & \not\subseteq & \mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}} \\ \mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}} & \not\subseteq & \mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-tt}} \\ \mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-T}} & \not\subseteq & \mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}} \\ \mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}} & \not\subseteq & \mathrm{P}^{\mathrm{NP}}_{q(n)\text{-tt}} \\ \mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}} & \subseteq & \mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}. \end{array}$$

In view of the power of oracle queries compared to mere time or space, we consider oracle queries to be a valuable computational resource. The last line above indicates that when accepting languages we can save queries by using Turing reductions instead of truth-table reductions.[1] Is a similar tradeoff possible when computing functions? Even when $q(n)$ is constant, it was previously unknown whether

$$\mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}} \subseteq \mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}},$$

although in [8] we showed that the statement was false under almost all relativizations. In contrast to the relationship between $\mathrm{P}^{\mathrm{NP}}_{q\text{-T}}$ and $\mathrm{P}^{\mathrm{NP}}_{(2^q-1)\text{-tt}}$, we show in this paper that for $q(n) \leq (1-\epsilon)\log n$

$$\mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}} \not\subseteq \mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}},$$

unless we can distinguish uniquely satisfiable formulas from unsatisfiable formulas in polynomial time, which would imply that P = UP (one-way functions fail to exist), that R = NP (there is a randomized polynomial time algorithm for every set in NP), and that the polynomial-time hierarchy collapses.

Since $\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}} = \mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-1)\text{-tt}}$ for $q(n) = O(\log n)$, it follows that $\mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-2)\text{-tt}} \subset \mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$ for $q(n) \leq (1-\epsilon)\log n$ unless $\mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-1)\text{-tt}} = \mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-2)\text{-tt}}$, which would force NP/poly = co-NP/poly by [59]. Similarly $\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}} \subset \mathrm{P}^{A}_{2^{q(n)}\text{-tt}}$ unless NP/poly = co-NP/poly. Thus, all relationships among the four bounded NP query hierarchies are resolved for small functions $q(n)$, assuming that certain complexity classes, currently believed to be distinct, do not coincide.

When q is a constant, we have two different proofs that

$$\mathrm{PF}^{\mathrm{NP}}_{(q+1)\text{-tt}} \not\subseteq \mathrm{PF}^{\mathrm{NP}}_{q\text{-T}}$$

assuming that certain complexity classes do not coincide. Each proof yields a stronger result. The second proof is valid for $q(n) \leq (1-\epsilon)\log n$ where $\epsilon \in (0,1]$. Although

[1]This result probably does not have practical applications because the length of the queries increases.

the first proof generalizes to $q(n) = o(n/\log n)$, the complexity separation that we assume grows stronger as q grows larger. As a bonus, this proof provides a significant partial answer to a question of Krentel's, because it yields the first separation result of the form

$$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}} \subset \mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-T}},$$

for $q(n) \geq \log n$ under plausible complexity-theoretic assumptions.

2. Terminology

P-terseness and p-superterseness were defined in [4] and [14] respectively.

Definition 2.

- A is q-p-terse if $\mathrm{PF}^{A}_{q(n)\text{-T}} \not\subseteq \mathrm{PF}^{A}_{(q(n)-1)\text{-tt}}$.
- A is p-terse if A is q-p-terse for all constant functions q.
- A is q-p-superterse if $\mathrm{PF}^{A}_{q(n)\text{-T}} \not\subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-tt}}$ for all X.
- A is p-superterse if A is q-p-superterse for all constant functions q.

Obviously, if A is p-superterse then

- A is p-terse.
- If $A \leq_{1\text{-tt}} B$ then B is p-superterse.

The classes UP and R were defined in [55] and [2, 48] respectively.

Definition 3.

- UP is the class of languages accepted by a polynomial time-bounded nondeterministic Turing machine such that at most one path accepts each input.
- R is the class of languages accepted by a polynomial time-bounded nondeterministic Turing machine such that either at least half of all paths accept or no paths accept each input. (It is also required that all paths make the same number of nondeterministic choices.)
- UTIME($T(n)$) is the class of languages accepted by a $T(n)$ time-bounded nondeterministic Turing machine such that at most one path accepts each input.
- RTIME($T(n)$) is the class of languages accepted by a $T(n)$ time-bounded nondeterministic Turing machine such that either at least half of all paths accept or no paths accept each input. (It is also required that all paths make the same number of nondeterministic choices.)

A formula is *uniquely satisfiable* if it has exactly one satisfying assignment.

Definition 4.

- SAT is the set of satisfiable Boolean formulas.
- UniqueSAT is the set of uniquely satisfiable Boolean formulas.
- UnSAT is the set of unsatisfiable Boolean formulas.

P-separability was defined in [36].

Definition 5. Two sets A and B are *p-separable* if there is a polynomial-time computable function $f : \Sigma^* \to \{0,1\}$ such that

$$f(x) = \begin{cases} 1 & \text{if } x \in A - B \\ 0 & \text{if } x \in B - A \\ \text{arbitrary} & \text{otherwise.} \end{cases}$$

It is obvious that if UniqueSAT and UnSAT are p-separable then P = UP. Valiant and Vazirani [56] have shown that if UniqueSAT and UnSAT are p-separable then R = NP.

In [9], we defined computability by a set of polynomial-time functions, which is called k-enumerability in [30]. This a central concept in the study of bounded query classes.

Definition 6.

- The total function h is *computable by a set of $f(n)$ $t(n)$-time functions* if there exists a two-place function g such that

  $$(\forall x)[h(x) \in \{g(i,x) : 1 \le i \le f(|x|)\}].$$

  and the running time of $g(i,x)$ is bounded by $t(|x|)$.

- The total function h is *computable by a set of $f(n)$ polynomial time functions* if h is computable by a set of $f(n)$ n^k-time functions for some k.

Informally, we say that *there are only $f(|x|)$ possible values for $h(x)$*. For example, every $0,1$-valued function is computable by a set of two polynomial-time functions (let $g(i,x) = i-1$). However, not every $0,1,2$-valued function is computable by a set of two polynomial-time functions, because we can diagonalize in the standard way.

The following lemma from [9] says that computability by a set of polynomial-time functions captures the information-theoretic aspects of polynomial-time computability by an oracle.

Lemma 7.

i. *If there exists an oracle B such that $h \in \mathrm{PF}^B_{q(n)\text{-T}}$ then h is computable by a set of $2^{q(n)}$ polynomial-time functions.*

ii. *If h is computable by a set of $2^{q(n)}$ polynomial-time functions then there exists an oracle B such that $h \in \mathrm{PF}^B_{q(n)\text{-tt}}$.*

By the preceding lemma we can conclude that every function $\leq^p_{\mathrm{tt}}$-reducible to A is computable by a set of $2^{n^{O(1)}}$ polynomial-time computable functions. However, if A is not p-superterse, then the following lemma from [9] allows us to draw a much stronger conclusion: Every function $\leq^p_{\mathrm{tt}}$-reducible to A is computable by a set of $n^{O(1)}$ functions.

Lemma 8. *If every function in $\mathrm{PF}^A_{q(n)\text{-tt}}$ is computable by a set of $2^{q(n)} - 1$ polynomial-time functions then every function in $\mathrm{PF}^A_{n\text{-tt}}$ is computable by a set of $n^{q(n)}$ $n^{O(q(n))}$-time functions.*

By ordinary padding techniques, the conclusion can be strengthened to say that every function polynomial-time truth-table reducible to A is computable by a set of $n^{O(q(n))}$ $n^{O(q(n))}$-time functions.

The bounded query hierarchies and the concepts of p-terseness and p-superterseness can be relativized to an oracle B by allowing the polynomial time Turing machines involved full access to the oracle B. We write p^B-terse and p^B-superterse to denote the relativized concepts.

We write $A(x) = 1$ if $x \in A$, 0 if $x \notin A$.

3. A Constant Number of Queries

In previous work, we have exploited the fact that a SAT oracle helps us find satisfying assignments in order to show that SAT is not cheatable [14]. In this paper, we exploit a similar, but stronger, property that applies to uniquely satisfiable formulas.

Theorem 9. *All* NP*-hard sets (under $\leq^p_{\mathrm{tt}}$-reductions) are p-superterse unless*

i. UniqueSAT *and* UnSAT *are p-separable, and*

ii. $\mathrm{P} = \mathrm{UP}$*, and*

iii. $\mathrm{R} = \mathrm{NP}$*, and*

iv. *the polynomial-time hierarchy collapses.*

Proof:

i. In [9], we showed that if A is not p-superterse then every function polynomial-time truth-table reducible to A is, in fact computable by a set of $n^{O(1)}$ polynomial-time computable functions.

   Assume that A is NP-hard under $\leq_{\mathrm{tt}}^{p}$-reductions, and that A is not p-superterse.

   We distinguish uniquely satisfiable formulas from unsatisfiable formulas as follows: Let F be a Boolean formula of length n with v indeterminates, $x_1, \ldots, x_v$. Let F_i be the Boolean formula obtained by assigning the value 1 to x_i in F. Let $g(F) = (\mathrm{SAT}(F_1), \ldots, \mathrm{SAT}(F_v))$. Since $g \leq_{\mathrm{tt}}^{p} \mathrm{SAT} \leq_{\mathrm{tt}}^{p} A$, g is computable by a set of $n^{O(1)}$ polynomial-time computable functions. Try all $n^{O(1)}$ possible values for $(\mathrm{SAT}(F_1), \ldots, \mathrm{SAT}(F_v))$, and see if

$$(x_1, \ldots, x_v) = (\mathrm{SAT}(F_1), \ldots, \mathrm{SAT}(F_v))$$

   is a satisfying assignment of F. If one of them is a satisfying assignment, then output 1, else output 0.

   Proof of correctness: If F has exactly one solution then that solution is $(\mathrm{SAT}(F_1), \ldots, \mathrm{SAT}(F_v))$, so the algorithm correctly outputs 1. If F has no solutions then the algorithm cannot find one, so the algorithm correctly outputs 0.

ii. If UniqueSAT and UnSAT are p-separable, then obviously P = UP.

iii. If UniqueSAT and UnSAT are p-separable, then R = NP [56].

iv. If R = NP then NP $\subseteq$ P/poly [1], so the polynomial-time hierarchy collapses [61].

∎

Corollary 10. *All* NP^B*-hard sets (under* $\leq_{\mathrm{tt}}^{p}$*-reductions) are* p^B*-superterse relative to almost all oracles* B*.*

Proof: The preceding results relativize. $\mathrm{R}^B \neq \mathrm{NP}^B$ relative to almost all oracles [20]. (It is also known that $\mathrm{P}^B \neq \mathrm{UP}^B$ for almost all oracles B [12, 21].) ∎

3.1. Extensions to Incomplete Sets

In this section we consider two kinds of sets believed not to be NP-complete. The techniques in the preceding section generalize to show that these sets are p-superterse unless certain complexity classes coincide.

Definition 11. [P-Separability via an Oracle] Let A and B be disjoint sets. We say that

$$(A, B) \leq_r^p C$$

if there is a polynomial-time oracle Turing machine M computing an r-reduction such that for all x

$$M^C(x)\begin{cases}\text{accepts} & \text{if } x \in A\\ \text{rejects} & \text{if } x \in B\\ \text{behaves arbitrarily} & \text{otherwise.}\end{cases}$$

Theorem 12.

i. *Assume that* (UniqueSAT, UnSAT) $\leq^p_{tt}$ A. *Then* A *is p-superterse unless* UniqueSAT *and* UnSAT *are p-separable.*

ii. *All* UP*-hard sets (under* $\leq^p_{tt}$*-reductions) are p-superterse unless* P = UP.

Proof:

i. This is the same as the proof of Theorem 9, because the formulas $F_1, \ldots, F_v$ defined in that proof have at most as many satisfying assignments as F.

ii. The proof is essentially the same. (To see the similarity, note that in part (i) we showed that if the problem of detecting a *promised* unique solution is $\leq^p_{tt}$-reducible to a non-p-superterse set, then in fact the problem belongs to P. In this part we must show that if the problem of detecting a *guaranteed* unique solution is $\leq^p_{tt}$-reducible to a non-p-superterse set, then in fact the problem belongs to P.)

∎

Definition 13. [Bounded-probability Reduction] $A \leq^{\mathrm{bpp}}_{tt} B$ if there is a probabilistic polynomial-time oracle Turing machine M computing a $\leq^p_{tt}$-reduction such that

$$\text{The probability that } M^B(x) \text{ accepts } \begin{cases}\geq 3/4 & \text{if } x \in A\\ \leq 1/4 & \text{otherwise.}\end{cases}$$

By amplification techniques (using polling) [20], we may assume for any polynomial p that M accepts with probability $\geq 1-2^{-p(|x|)}$ if $x \in A$, and with probability $\leq 2^{-p(|x|)}$ if $x \notin A$.

Theorem 14. *All* NP*-hard sets under* ($\leq^{\mathrm{bpp}}_{tt}$*-reductions) are p-superterse unless* R = NP.

Proof: Assume that A is not p-superterse; then every function in $\mathrm{PF}^A_{n\text{-}tt}$ is computable by a set of $n^{O(1)}$ polynomial-time computable functions. Assume that that SAT $\leq^{\mathrm{bpp}}_{tt}$ A; then there is a probabilistic polynomial-time oracle Turing machine M computing a $\leq^p_{tt}$-reduction such that

$$\text{The probability that } M^A(x) \text{ accepts } \begin{cases}\geq 1-1/(2n) & \text{if } x \in \text{SAT}\\ \leq 1/(2n) & \text{otherwise.}\end{cases}$$

Using the technique from the proof of Theorem 9 we probabilistically separate UniqueSAT from UnSAT. Each potential witness depends on n instances of SAT. Since M errs with probability $\leq 1/(2n)$, each potential witness is computed incorrectly with probability $\leq n(1/(2n)) = 1/2$. In particular, we obtain the correct witness (if one exists) with probability at least 1/2. Thus we have a probabilistic polynomial-time Turing machine M' such that

$$M'(x) \begin{cases} \text{accepts with probability} \geq 1/2 & \text{if } x \in \text{UniqueSAT} \\ \text{rejects} & \text{if } x \in \text{UnSAT} \\ \text{behaves arbitrarily} & \text{otherwise.} \end{cases}$$

As shown in [56], this implies that $\mathrm{R} = \mathrm{NP}$. ▮

Definition 15. [BPP-Separability via an Oracle] Let A and B be disjoint sets. We say that

$$(A, B) \leq_{\mathrm{tt}}^{\mathrm{bpp}} C$$

if there is a polynomial-time oracle Turing machine M computing a $\leq_{\mathrm{tt}}^{p}$-reduction such that for all x

$$\text{the probability that } M^C(x) \text{ accepts} \begin{cases} \geq 3/4 & \text{if } x \in A \\ \leq 1/4 & \text{if } x \in B \\ \text{is arbitrary} & \text{otherwise.} \end{cases}$$

The preceding proof yields a stronger result:

Theorem 16. *Assume that* $(\text{UniqueSAT}, \text{UnSAT}) \leq_{\mathrm{tt}}^{\mathrm{bpp}} A$. *Then A is p-superterse unless* $\mathrm{R} = \mathrm{NP}$.

4. Number of Queries A Function of Input Length

In this section, we allow the number of queries to be a function of the input length, and we find a partial solution to a problem of Krentel [43]. Krentel showed that if $q(n) \leq \frac{1}{2}\log n$ then

$$\mathrm{PF}^{\mathrm{SAT}}_{(q(n)-1)\text{-T}} \subset \mathrm{PF}^{\mathrm{SAT}}_{q(n)\text{-T}} \qquad \text{unless } \mathrm{P} = \mathrm{NP}.$$

He asked whether this is also true for larger functions $q(n)$. We show that this is the case unless certain natural complexity classes, which we expect to differ, coincide.

Theorem 17. *Let q be a polynomial-time computable function such that $q(n) \geq 1$. Let A be* NP*-hard under $\leq_{\mathrm{tt}}^{p}$-reductions, and let X be any set.*

$$\mathrm{PF}^{A}_{q(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-T}}$$

unless

i. UniqueSAT *and* UnSAT *are separable in* $\mathrm{DTIME}(n^{O(q(n))})$, *and*

ii. $\mathrm{UP} \subseteq \mathrm{DTIME}(n^{O(q(n^{O(1)}))})$, *and*

iii. $\mathrm{SAT} \in \mathrm{RTIME}(n^{O(q(n))})$, *and*

iv. $\mathrm{NP} \subseteq \mathrm{RTIME}(n^{O(q(n^{O(1)}))})$.

Proof: In [9], we showed that if $\mathrm{PF}^{A}_{q(n)\text{-tt}} \subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-T}}$ then every function polynomial-time truth-table reducible to A is, in fact, computable by a set of $n^{O(q(n))}$ $n^{O(q(n))}$-time functions. The remainder of the proof is essentially the same as the proof of Theorem 9. ∎

Note that parts (i,iii) above are interesting for $q(n) = o(n/\log n)$, while parts (ii,iv) are interesting for $q(n) = n^{o(1)}$.

The special case $q(n) = \log n$ is rather pretty. Define $\mathrm{DTIME}(n^{\mathrm{polylog}\, n}) = \cup_{k\geq 0}\mathrm{DTIME}(n^{\log^k n})$, and define UTIME, NTIME, and RTIME similarly.

Corollary 18. *Let A be* NP*-hard under $\leq^p_{\mathrm{tt}}$-reductions, and let X be any set.*

$$\mathrm{PF}^{A}_{\log^i n\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(\log^i n-1)\text{-T}}$$

unless

- $\mathrm{DTIME}(n^{\mathrm{polylog}\, n}) = \mathrm{UTIME}(n^{\mathrm{polylog}\, n})$, *and*
- $\mathrm{RTIME}(n^{\mathrm{polylog}\, n}) = \mathrm{NTIME}(n^{\mathrm{polylog}\, n})$.

Proof: By the preceding theorem, we obtain $\mathrm{UP} \subseteq \mathrm{DTIME}(n^{O(\log^i n)})$. By a standard translational lemma as in [49], we can replace n by $n^{\log^k n}$, obtaining $\mathrm{UTIME}(n^{\log^k n}) \subseteq \mathrm{DTIME}(n^{O(\log^{ik+k+1} n)})$. The other part is similar. ∎

By weakening the preceding results, we obtain partial answers to Krentel's question.

Corollary 19.

i. Let q be a polynomial-time computable function.

$$\mathrm{PF}^{\mathrm{NP}}_{(q(n)-1)\text{-T}} \subset \mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$$

unless

- UniqueSAT *and* UnSAT *are separable in* $\mathrm{DTIME}(n^{O(q(n))})$, *and*
- $\mathrm{UP} \subseteq \mathrm{DTIME}(n^{O(q(n^{O(1)}))})$, *and*
- $\mathrm{SAT} \in \mathrm{RTIME}(n^{O(q(n))})$, *and*
- $\mathrm{NP} \subseteq \mathrm{RTIME}(n^{O(q(n^{O(1)}))})$.

ii.

$$\mathrm{PF}^{\mathrm{NP}}_{(\log^i n-1)\text{-T}} \subset \mathrm{PF}^{\mathrm{NP}}_{\log^i n\text{-T}}$$

unless

- $\mathrm{DTIME}(n^{\mathrm{polylog}\, n}) = \mathrm{UTIME}(n^{\mathrm{polylog}\, n})$, *and*
- $\mathrm{RTIME}(n^{\mathrm{polylog}\, n}) = \mathrm{NTIME}(n^{\mathrm{polylog}\, n})$.

By combining our techniques with Krentel's techniques, we can improve our result for small functions $q(n)$. Although the separation is valid only for $q(n) \leq (1-\epsilon)\log n$, the complexity assumption involves only polynomial time bounds.

Theorem 20. *Let q be a polynomial-time computable function such that $q(n) \leq (1-\epsilon)\log n$ where $\epsilon \in (0,1]$. Let A be* NP*-hard under $\leq^p_{1\text{-tt}}$-reductions, and let X be any set.*

$$\mathrm{PF}^{A}_{q(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-T}}$$

unless

i. UniqueSAT *and* UnSAT *are p-separable.*

ii. P = UP.

iii. R = NP.

iv. the polynomial-time hierarchy collapses.

Proof:

i. Assume that $\mathrm{PF}^{A}_{q(n)\text{-tt}} \subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-T}}$. Let $|F|$ denote the length of the formula F, and let $||F||$ denote the number of indeterminates in F. We distinguish uniquely satisfiable formulas from unsatisfiable formulas as follows: Perform the following reduction step until we obtain a formula F such that the number of independent indeterminates (defined below) in F is less than $q(|F|)$:

Reduction Step: Let F be a Boolean formula, let $n = |F|$, and let $v = ||F||$. Let $x_1, \ldots, x_v$ be the indeterminates in F. Let F_i be the Boolean formula obtained by assigning the value 1 to x_i in F. Let $g(F) = (\mathrm{SAT}(F_1), \ldots, \mathrm{SAT}(F_{q(n)}))$. Since SAT $\leq_{1\text{-tt}} A$, it follows that $g \in \mathrm{PF}^{A}_{q(n)\text{-tt}} \subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-T}}$ (by assumption) via some oracle Turing machine M. Simulate M for all possible oracle answers. Modify M, if necessary, so that distinct sequences of oracle answers produce distinct values of g. Express each bit of the value $g(F)$ as a Boolean formula of M's oracle answers, which we represent by new indeterminates, $y_1, \ldots, y_{q(n)-1}$. The size of each Boolean formula in disjunctive-normal-form is bounded by $(q(n)-1)2^{q(n)-1}\log v$; so the size of their conjunction F_g is bounded by $q(n)(q(n)-1)2^{q(n)-1}\log v$. Let $F = F \wedge F_g$, and reorder the indeterminates $y_1, \ldots, y_{q(n)-1}, x_{q(n)+1}, \ldots, x_v, x_1, \ldots, x_{q(n)}$. We say that an indeterminate is *dependent* if it has been moved to the end of the list; *independent*, otherwise.

Final Step: We try every distinct assignment to the independent indeterminates (which efficiently determines an assignment to the dependent indeterminates). If one of them is a satisfying assignment, then output 1, else output 0.

Proof of correctness: If the input formula F is uniquely satisfiable, then $g(F)$ is the assignment to F's first $q(n)$ indeterminates in F's unique satisfying assignment. Therefore, the formula produced by the reduction step is uniquely satisfiable. Obviously, if the input formula F is unsatisfiable, then the formula produced by the reduction step is unsatisfiable. The assignments to the dependent indeterminates are determined easily from the Boolean formula F_g. If F is uniquely satisfiable the final formula is uniquely satisfiable, and in particular, satisfiable. If F is unsatisfiable then the final formula is unsatisfiable.

Time analysis: Since $q(n) < \log n$, the final step and all reduction steps run in time that is polynomial in $|F|$. The reduction step reduces the number of dependent indeterminates by 1. Let n be the length of the input formula F. Then, after at most $n-1$ iterations, $||F|| < q(|F|)$. It suffices to show that after each reduction step, $|F|$ is bounded by a polynomial in the size of the input formula.

Let u_s denote $|F|$ after the sth reduction step, and let $n = u_0$. Then

$$u_{s+1} \leq u_s + q(u_s)(q(u_s) - 1)2^{q(u_s)} \log u_s.$$

Let $\epsilon' = \frac{1}{2}\epsilon$, so that for sufficiently large n,

$$u_{s+1} \leq u_s + 2^{(1-\epsilon')\log u_s} = u_s + u_s^{1-\epsilon'}.$$

Let n be sufficiently large. We show by induction on s that, for all $s \leq n-1$,

$$u_s \leq n + sn^{1/\epsilon' - 1}.$$

The base case is established because $u_0 = n$. Assume that the inductive hypothesis is true for some s.

$$\begin{aligned}
u_{s+1} &\leq u_s + u_s^{1-\epsilon'} \\
&\leq n + sn^{1/\epsilon'-1} + (n + sn^{1/\epsilon'-1})^{1-\epsilon'} \\
&\leq n + sn^{1/\epsilon'-1} + (n + (n-1)n^{1/\epsilon'-1})^{1-\epsilon'} \qquad \text{because } s \leq n-1 \\
&= n + sn^{1/\epsilon'-1} + (n + n^{1/\epsilon'} - n^{1/\epsilon'-1})^{1-\epsilon'} \\
&\leq n + sn^{1/\epsilon'-1} + (n^{1/\epsilon'})^{1-\epsilon'} \qquad \text{because } \epsilon' \leq 1/2 \\
&= n + sn^{1/\epsilon'-1} + n^{1/\epsilon'-1} \\
&= n + (s+1)n^{1/\epsilon'-1}.
\end{aligned}$$

The claim is established. Thus, for all $s \leq n-1$,

$$u_s \leq n + (n-1)n^{1/\epsilon'-1} = n + n^{1/\epsilon'} - n^{1/\epsilon'-1} \leq n^{1/\epsilon'}.$$

Since $|F|$ is bounded by a polynomial after every reduction step, the proof is complete.

(ii-iv) Same as in the proof of Theorem 9.

▮

The reduction in the proof above owes a lot to the work of Krentel, who only considered the recurrence for $q(n) \leq \frac{1}{2}\log n$. Using Krentel's original reduction and our analysis of the recurrence, we can extend Krentel's result to $q(n) \leq (1-\epsilon)\log n$.

Theorem 21. *Let q be a polynomial-time computable function such that $q(n) \leq (1-\epsilon)\log n$ where $\epsilon \in (0,1]$. Then for all X, $\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}} \not\subseteq \mathrm{PF}^{X}_{(q(n)-1)\text{-T}}$ unless $\mathrm{P} = \mathrm{NP}$.*

We can extend Theorem 17 in the same way that we extended Theorem 9 to obtain Theorems 12, 14, and 16 in Section 3.1. We leave the details to the reader.

We can also extend Theorem 20 in the same way that we extended Theorem 9 to obtain Theorem 12 in Section 3.1. However, we cannot readily generalize to $\leq^{\mathrm{bpp}}_{1\text{-tt}}$-hard sets, because amplification of probabilities also increases the size of truth-tables.

In [43], Krentel showed that if $\mathrm{PF}^{\mathrm{SAT}}_{n\text{-T}} \subseteq \mathrm{PF}^{\mathrm{SAT}}_{\log n\text{-T}}$ then $\mathrm{P} = \mathrm{NP}$. We present an easy generalization discovered by Gasarch and Krentel.

Theorem 22. *Let $q(n) \geq \log n$. Let A be NP-hard under $\leq^{p}_{1\text{-tt}}$-reductions, and let X be any set.*

$$\mathrm{PF}^{A}_{n\text{-T}} \not\subseteq \mathrm{PF}^{X}_{q(n)\text{-T}}$$

unless

- $\mathrm{SAT} \in \mathrm{DTIME}(2^{q(n)})$.
- $\mathrm{NP} \subseteq \mathrm{DTIME}(2^{q(n^{O(1)})})$.

Proof:

- It is possible to find a satisfying assignment to a formula F (if one exists) by making n serial queries to SAT [43]. Since A is NP-hard under $\leq^{p}_{1\text{-tt}}$-reductions, n serial queries to A work as well. Assume that $(\exists X)[\mathrm{PF}^{A}_{n\text{-T}} \subseteq \mathrm{PF}^{X}_{q(n)\text{-T}}]$ via some oracle Turing machine M. Then, in order to find a satisfying assignment for a formula F, we can simulate M for all possible oracle answers, yielding $2^{q(n)}$ possible satisfying assignments. If one of those assignments satisfies F, then F is satisfiable; otherwise F is unsatisfiable.
- By (i), $\mathrm{SAT} \in \mathrm{DTIME}(2^{O(q(n))})$. Therefore $\mathrm{NP} \subseteq \mathrm{DTIME}(2^{q(n^{O(1)})})$.

▮

5. A Relationship Between Helping and P-Terseness

In this section we pinpoint the properties of satisfiability that led to the results in the preceding sections.

In [41], Ko described the concept of helping by an oracle. We rephrase his definition of helping, which turns out to be strongly related to p-terseness. Let $\leq_r^p$ represent a polynomial-time r-reduction. For example, r could be m, k-tt, tt, or T.

Definition 23.

i. Let $B \in \mathrm{NP}$. A set A *r-helps* B on C if there exist a polynomial p and a polynomial-time computable relation R such that

$$B = \{x : (\exists y . |y| \leq p(|x|))[R(x,y)]\},$$

and there exists a *witness function* $w \leq_r^p A$ such that

$$[x \in B \cap C \Rightarrow R(x, w(x))].$$

ii. A set A *r-one-helps* B if A r-helps B on B.

Intuitively, A r-helps B on C if for every string x in $B \cap C$, A efficiently provides a witness to the fact $x \in B$.

Observation 24. SAT *tt-helps* SAT *on* UniqueSAT.

Proof: Assume that $F \in \mathrm{SAT} \cap \mathrm{UniqueSAT} = \mathrm{UniqueSAT}$. As in the proof of Theorem 9, $(\mathrm{SAT}(F_1), \ldots, \mathrm{SAT}(F_v))$ is the unique satisfying assignment of F, and hence a witness to F's membership in SAT. ∎

The following theorem is a generalization of Theorem 9, and can be proved in the same way. (Theorem 17 generalizes similarly.)

Theorem 25. *Let $B \subseteq A$, and suppose that X tt-helps A on B. If X is not p-superterse then B and $\bar{A}$ are p-separable.*

It is well known that a SAT oracle permits us to find satisfying assignments in polynomial time [41].

Observation 26. SAT *T-one-helps* SAT.

We used that observation implicitly in [9] to obtain the following result:

Corollary 27. *Let X be hard for Δ_2^{P} under $\leq_{\mathrm{tt}}^p$-reductions. The set X is p-superterse unless* $\mathrm{P} = \mathrm{NP}$.

Proof: Because SAT T-one-helps SAT, a witness function w is polynomial-time Turing reducible to SAT. Because X is $\leq_{\mathrm{tt}}^{p}$-hard for P^{NP}, the witness function w is polynomial-time truth-table reducible to X. Therefore, X tt-one-helps SAT. If X is not p-superterse then, by Theorem 25, SAT and UnSAT are p-separable, so SAT $\in$ P. ∎

T-one-helping has previously been applied to p-terseness in less complicated ways. The following generalization of Theorem 22 is implicit in work by Balcazar [5]. (The theorem can be generalized to $q(n) > \log n$ by replacing P with $\mathrm{DTIME}(2^{q(n^{O(1)})})$.)

Theorem 28. *Suppose that A T-one-helps B. Let X be any set.*

$$\mathrm{PF}^{A}_{n^{\epsilon}\text{-T}} \not\subseteq \mathrm{PF}^{X}_{O(\log n)\text{-T}}$$

unless $B \in \mathrm{P}$.

Proof: Assume that $\mathrm{PF}^{A}_{n^{\epsilon}\text{-T}} \subseteq \mathrm{PF}^{X}_{q(n)\text{-T}}$. By padding the input to length $n^{k/\epsilon}$, we obtain

$$\mathrm{PF}^{A}_{n^{k}\text{-T}} \subseteq \mathrm{PF}^{X}_{O((k/\epsilon)\log n)\text{-T}} = \mathrm{PF}^{X}_{O(\log n)\text{-T}}.$$

Therefore, every function polynomial-time Turing reducible to A is computable by a set of $n^{O(1)}$ polynomial time functions. Since A T-one-helps B, the set B has a witness function that is polynomial-time Turing reducible to A. Therefore, there is a set of $n^{O(1)}$ polynomial-time functions one of which produces a witness for $x \in B$ (if $x \in B$). By testing all potential witnesses, we determine whether $x \in B$. ∎

Corollary 29.

i. *Let $\mathcal{C}$ be one of the classes* UP, FewP, *or* NP, *and let A be $\leq_{\mathrm{T}}^{p}$-hard for $\mathcal{C}$. For all sets X,*

$$\mathrm{PF}^{X}_{O(\log n)\text{-T}} \not\subseteq \mathrm{PF}^{A}_{n^{\epsilon}\text{-T}}$$

unless $\mathrm{P} = \mathcal{C}$.

ii. *If* (UniqueSAT, UnSAT) $\leq_{\mathrm{T}}^{p} A$, *then for all sets X*

$$\mathrm{PF}^{X}_{O(\log n)\text{-T}} \not\subseteq \mathrm{PF}^{A}_{n^{\epsilon}\text{-T}}$$

unless UniqueSAT *and* UnSAT *are p-separable.*

Proof:

i. In the proof that SAT T-one-helps SAT, it is easy to see that the reduction does not increase the number of satisfying assignments. Therefore $\leq_{\mathrm{m}}^{p}$-hard sets for $\mathcal{C}$ T-one-help all sets in $\mathcal{C}$. Since T-one-helping is preserved under $\leq_{\mathrm{T}}^{p}$-reductions, $\leq_{\mathrm{T}}^{p}$-hard sets for $\mathcal{C}$ T-one-help all sets in $\mathcal{C}$. The conclusion follows from the preceding Theorem.

ii. Similar.

∎

Note that we have extended Theorem 22 to NP-hard sets under $\leq_{\mathrm{T}}^{p}$-reductions. Note also that this corollary applies whenever we define $\mathcal{C}$ by placing a reasonable bound on the number of witnesses.

The proofs of Theorem 20 and Theorem 21 use a stronger property of SAT than helping. Not only do we need to be able to produce a witness with a polynomial number of queries, we need to be able to produce any subsequence of a witness with a number of queries equal to the length of the subsequence. In Theorem 20, the witness was unique, so we could have produced several subsequences of the witness independently; however, that additional property does not seem to allow any improvement of the result.

6. P-Selective sets

P-selective sets are an interesting class defined by Selman in [52]. They are analogous to the semi-recursive sets defined by Jockusch in [37].

Definition 30. C is *p-selective* if there is a polynomial-time computable function f of two arguments such that

$$f(x,y) = \begin{cases} x & \text{if } x \in C \text{ and } y \notin C, \\ y & \text{if } y \in C \text{ and } x \notin C, \\ \text{arbitrary} & \text{otherwise.} \end{cases}$$

It is easy to verify that p-selective sets are not p-terse [3]. Therefore we obtain the following corollary:

Corollary 31.

i. NP*-hard sets (under* $\leq_{\mathrm{tt}}^{p}$*-reductions) are not p-selective unless* UniqueSAT *and* UnSAT *are p-separable,* P = UP, R = NP *and the* PH *collapses.*

ii. Δ_2^{P}*-hard sets (under* $\leq_{\mathrm{tt}}^{p}$*-reductions) are not p-selective unless* P = NP*.*

Proof:

i. By Theorem 9.

ii. By Corollary 27.

∎

Those results were obtained independently by Toda in [54]. He also obtained some other interesting results about p-selective sets.

7. NP-Complete Sets are P-Terse iff They are P-Superterse

In [4], Amir and Gasarch defined p-terseness. In some of our work [13, 14, 9] we have preferred to study p-superterseness. Since all p-superterse sets are p-terse, our proofs of p-superterseness are automatically proofs of p-terseness. Because p-superterseness can be characterized in a combinatorial way in terms of computability by a set of polynomial-time functions (Lemma 7), we can bring a powerful tool to bear on p-superterseness (Lemma 8).

However, one may suspect that by studying p-superterseness we might have overlooked some subtle properties of p-terseness. In this section we show that for many kinds of sets p-terseness is equivalent to p-superterseness. In particular NP-complete sets are p-terse iff they are p-superterse; the same applies to PSPACE-complete and exponential-time-complete sets. This is further justification (after the fact) of our preference to study p-superterseness instead of p-terseness.

7.1. r-Cylinders

We will define r-cylinders, show that for pbtt-cylinders p-terseness is equivalent to p-superterseness, and show that many important sets are pbtt-cylinders. For example, we show that all NP-complete sets are p-superterse iff they are p-terse. The definition of an r-cylinder below is a generalization of Jockusch's definition of a p-cylinder (Jockusch used "p" to denote positive-truth-table, not polynomial-time) in recursion theory, and is related to the definition of a cylinder in complexity theory.[2]

Definition 32. Let r be a reduction.

- A set B is polynomial-time *positive* r-reducible to A (denoted $B \leq_{\mathrm{p}r} A$) if $B \leq_r^p A$ via an oracle Turing machine M such that

$$C \subseteq D \Longrightarrow M^C \subseteq M^D.$$

- A set B is polynomial time *bounded* r-reducible to A (denoted $B \leq_{\mathrm{b}r} A$) if $B \leq_r^p A$ via an oracle Turing machine that makes at most k queries to its oracle for some constant k.

- A set A is an *r-cylinder* if

$$B \leq_r^p A \Longrightarrow B \leq_{\mathrm{m}}^p A.$$

Definition 33. A is polynomial-time Boolean formula reducible to B (denoted $A \leq_{\mathrm{Bf}}^p B$) if there exist polynomial-time computable functions q and H such that

[2] A is a cylinder if $B \leq_{\mathrm{m}}^p A \Longrightarrow B \leq_1^p A$.

$q(x) = \langle k, q_1, \ldots, q_k \rangle$,

$H(x)$ is a Boolean formula, and

$A(x) = H(x)(A(x_1), \ldots, A(q_k))$.

Because every truth table (tt) with k inputs can be expressed as a Boolean formula (Bf) of size $k2^k$, polynomial-time bounded-truth-table reductions (btt) are the same as polynomial-time bounded-Boolean-formula reductions (bBf). Note that all T-cylinders are pT-cylinders, all pT-cylinders are ptt-cylinders, all ptt-cylinders are pBf-cylinders, and all pBf-cylinders are pbtt-cylinders. We show that many kinds of sets are pbtt-cylinders.

Kadin considered $\log n$-T-cylinders implicitly [38]. He claimed that A is a $\log n$-T-cylinder if $A \times A \leq_{\mathrm{m}}^{p} A$ and $\bar{A} \leq_{\mathrm{m}} A$. However, his proof is incorrect, because he composed polynomial-time reductions $\log n$ times. If a reduction maps strings of length n to strings of length n^2, then its $\log n$th iterate maps strings of length n to strings of length n^n. To maintain a polynomial-time bound, it is possible in general to iterate polynomial-time reductions only a constant number of times. We present a valid version of Kadin's claim.

Proposition 34. *A is a btt-cylinder iff $A \times A \leq_{\mathrm{m}}^{p} A$ and $\bar{A} \leq_{\mathrm{m}} A$.*

Proof: Since $\bar{A} \leq_{\mathrm{m}}^{p} A$, we can eliminate negations and we can turn disjunctions into conjunctions, which we can collapse because $A \times A \leq_{\mathrm{m}}^{p} A$; thus we can turn a $\leq_{\mathrm{btt}}^{p}$-reduction into an $\leq_{\mathrm{m}}^{p}$-reduction. The converse is obvious. ∎

Because nondeterministic time classes are not known to be closed under complement, we will be most interested in positive reductions.

Lemma 35. *A is a pbtt-cylinder iff $A \times A \leq_{\mathrm{m}}^{p} A$ and $\bar{A} \times \bar{A} \leq_{\mathrm{m}}^{p} \bar{A}$,*

Proof: If $A \times A \leq_{\mathrm{m}}^{p} A$ and $\bar{A} \times \bar{A} \leq_{\mathrm{m}}^{p} \bar{A}$ then we can collapse disjunctions and conjunctions, thus turning a $\leq_{\mathrm{pbtt}}^{p}$-reduction into an $\leq_{\mathrm{m}}^{p}$-reduction. The converse is obvious. ∎

Theorem 36.

i. If A is $\leq_{\mathrm{m}}^{p}$-complete for DTIME$(f(n))$, DSPACE$(f(n))$, *or* NSPACE$(f(n))$, *where $f(n) \geq n$, then A is a T-cylinder.*

ii. If A is $\leq_{\mathrm{m}}^{p}$-complete for NTIME$(f(n))$, *where $f(n) \geq n$, then A is a pT-cylinder.*

iii. If A is $\leq_{\mathrm{m}}^{p}$-complete for DSPACE$(f(n))$ *or* NSPACE$(f(n))$, *where $f(n) \geq \log n$, then A is a pbtt-cylinder.*

Proof:

i. If $A \in \mathrm{DTIME}(f(n))$ and $B \leq_{\mathrm{T}}^{p} A$, then $B \in \mathrm{DTIME}(f(n^{O(1)}))$. Since A is $\leq_{\mathrm{m}}^{p}$-hard for $\mathrm{DTIME}(f(n))$, it follows that $B \leq_{\mathrm{m}}^{p} A$. The proof for DSPACE is similar. For NSPACE, use Savitch's theorem.

ii. Let A be $\leq_{\mathrm{m}}^{p}$-complete for $\mathrm{NTIME}(f(n))$. If $B \leq_{\mathrm{pT}}^{p} A$ then we can guess a sequence of oracle answers and we can guess witnesses to the positive answers. Then we can check that those answers lead to acceptance and that the witnesses are correct, so $B \in \mathrm{NTIME}(f(n^{O(1)}))$. Therefore $B \leq_{\mathrm{m}}^{p} A$.

iii. By the linear speedup theorem, those classes are closed under union and intersection. The conclusion follows from the preceding lemma.

∎

For example, $\leq_{\mathrm{m}}^{p}$-complete sets for NP, PSPACE, $\mathrm{DSPACE}(\log^2 n)$, and exponential time are pbtt-cylinders.

Theorem 37. *Let k and j be constants. Let R be a pbtt-cylinder.*

i. $(\exists X)[\mathrm{PF}_{k\text{-tt}}^{R} \subseteq \mathrm{PF}_{j\text{-T}}^{X}]$ *iff* $\mathrm{PF}_{k\text{-tt}}^{R} \subseteq \mathrm{PF}_{j\text{-T}}^{R}$.

ii. R is k-p-superterse iff R is k-p-terse.

iii. R is p-superterse iff R is p-terse.

Before proving this theorem, we require a definition and a combinatorial lemma.

Definition 38. Let $C_1, \ldots, C_k$ be subsets of a universe U. A set F is a *positive combination* of $C_1, \ldots, C_k$ if

- $F = U$, or
- $F = C_i$ for some i, or
- $F = F_1 \cup F_2$ where F_1 and F_2 are positive combinations, or
- $F = F_1 \cap F_2$ where F_1 and F_2 are positive combinations.

Equivalently, F is a positive combination of $C_1, \ldots, C_k$ if F is an element of the lattice $(\{C_i, \ldots, C_k\}; \subseteq; \cup; \cap)$.

We write $|S|$ to denote the cardinality of a finite set S.

Lemma 39. *Let T be an n-by-k array of 0's and 1's. Call the rows $R_1, \ldots, R_n$. Call the columns $C_1, \ldots, C_k$. Interpret each row R_i as a subset of $\{1, \ldots, k\}$. Interpret each column C_i as a subset of $\{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.*

Assume that $(\forall i \neq j)[R_i \neq R_j]$, that is, the rows of T are distinct sets. Then, given any number m such that $0 < m \leq n$, there is a positive combination F of $C_1, \cdots, C_k$ such that $|F| = m$. Furthermore the combination can be found in time polynomial in k and n.

Proof: Proof by induction on k. Base case ($k = 0$): Since the rows of T are distinct, $n = 1$. Therefore $m = 1$. Let $F = U = \{1\}$.

Inductive case ($k > 0$):

- If $|C_k| = p \geq m$ then let $k' = k - 1$, $n' = p$, and $m' = m$. Let $C_i' = C_i \cap C_k$ for $1 \leq i \leq k - 1$. Let i' be the ith smallest element of C_k, and let $R_i' = R_{i'} - \{k\}$ for $1 \leq i \leq p$. Let the columns of T' be $C_1', \ldots, C_{k-1}'$, and the rows of T' be $R_1', \ldots, R_p'$.

  In order to apply the inductive hypothesis, we must show that the rows of T' are distinct. Assume $i < j \leq p$. Let i' be the ith smallest element of C_k, and let j' be the jth smallest element of C_k. Then $k \in R_{i'}$ and $k \in R_{j'}$. By assumption, $R_{i'} \neq R_{j'}$, so $R_{i'}$ and $R_{j'}$ must differ on some element of $\{0, \ldots, k-1\}$. Thus $R_i' \neq R_j'$.

  Apply the inductive hypothesis to obtain the positive combination F' of the columns $C_1', \ldots, C_{k-1}'$. Let $F = C_k \cap F'$. Since $C_i' = C_i \cap C_k$, the distributive law allows us to simplify F by replacing C_i' with C_i.

- If $|C_k| = p < m$ then let $k' = k-1$, $n' = n-p$, and $m' = m-p$. Let $C_i' = C_i \cap \bar{C}_k$ for $1 \leq i \leq k - 1$. Let i' be the ith smallest element of $\bar{C}_k$ for $1 \leq i \leq n - p$. Let the columns of T' be $C_1', \ldots, C_{k-1}'$, and the rows of T' be $R_1', \ldots, R_{n-p}'$.

  As in the previous case, the rows of T' are distinct. Apply the inductive hypothesis to obtain the positive combination F' of the columns $C_1', \ldots, C_{k-1}'$. Let $F = C_k \cup F'$. Since $C_i' = C_i \cap \bar{C}_k$, the distributive law allows us to simplify F (and obtain a positive combination) by replacing C_i' with C_i.

The polynomial-time algorithm is implicit in the construction. ∎

At this point the preceding lemma will either seem obscure or obvious. In the former case, an example will be helpful.

Example 40. Let $m = 4$, and let

$$T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

In this example, $p = |C_3| = 3 < m$ so we are in the second case. We will let $F = C_3 \cup F'$. Proceeding recursively, we have $m' = m - p = 1$, and

$$T' = \begin{pmatrix} 0 & 0 \\ \\ 0 & 1 \\ \\ 1 & 0 \\ \\ 1 & 1 \end{pmatrix}.$$

Now $p = |C_2| = 2 \geq m$, so we are in the first case. We will let $F = C_2 \cap F'$. Proceeding recursively, we have $m' = m = 2$, and

$$T' = \begin{pmatrix} \\ 0 \\ \\ \\ 1 \end{pmatrix}.$$

Now $p = |C_1| = 1 \geq m$, so we are in the first case. We will let $F = C_1 \cap F'$. Proceeding recursively, $m' = m = 1$, and $k' = k - 1 = 0$ so we are in the base case. We let $F = U$. Thus

$$\begin{aligned} F &= C_3 \cup (C_2 \cap (C_1 \cap U)) \\ &= C_3 \cup (C_2 \cap C_1) \\ &= \{2, 4, 6, 7\}, \end{aligned}$$

so $|F| = 4$ as desired.

The preceding lemma allows us to prove the theorem easily.

Proof of Theorem 37:

i. Let R be a pbtt-cylinder. Suppose that $\mathrm{PF}^R_{k\text{-tt}} \subseteq \mathrm{PF}^X_{j\text{-T}}$. By simulating for all possible oracle answers, we find 2^j possible values $R_1, \ldots, R_{2^j}$ for $R \cap \{x_1, \ldots, x_k\}$. Without loss of generality assume that these values are distinct. We know that

$$R_z = R \cap \{x_1, \ldots, x_k\}$$

for a unique $z \leq 2^j$. We will determine z via a binary search. For $1 \leq i \leq k$, let

$$C_i = \{s : x_i \in R_s\}.$$

By the preceding Lemma, we can find a monotone combination F of $C_1, \ldots, C_k$ such that $|F| = 2^{j-1}$. By determining whether $z \in F$ we reduce the problem size by a factor of two. By construction,

$$F = C_k \circ_k (C_{k-1} \circ_{k-1} (\cdots \circ_2 C_1 \cdots)),$$

where $\circ_i$ is either $\cup$ or $\cap$. Therefore

$$z \in F \iff (z \in C_k) \circ_k ((z \in C_{k-1}) \circ_{k-1} (\cdots \circ_2 (z \in C_1) \cdots)),$$

where $\circ_i$ is either *and* or *or*. However, $z \in C_i \iff x_i \in R_z = R$. Let

$$S = \{(x_1, \ldots, x_k) : (x_k \in R) \circ_k ((x_{k-1} \in R) \circ_{k-1} (\cdots \circ_2 (x_1 \in R) \cdots))\}.$$

Then $z \in F$ iff $(x_1, \ldots, x_k) \in S$. Obviously $S \leq^p_{\mathrm{pbtt}} R$. Since R is a pbtt-cylinder, $S \leq^p_{\mathrm{m}} R$. Therefore we can determine whether $z \in F$ by making a single query to R. This rules out half of the rows. Repeat a total of j times to determine the correct z.

The converse is obvious.

ii. Let $j = k - 1$ in part (i).

iii. Part (ii) is true for all k.

∎

Corollary 41. *Let A be $\leq^p_m$ complete for* NP, PSPACE*, or exponential time. Then A is p-terse iff A is p-superterse.*

7.2. A Translational Lemma

In this section we show that if A is not p-terse then there exists a real number $r < 1$ such that for all k, $\mathrm{PF}^A_{k\text{-tt}} \subseteq \mathrm{PF}^A_{k^r\text{-T}}$. While this translational lemma is quantitatively weaker than the similar lemma that we proved in [9], it does not require a different oracle. For that reason, the translational lemma will allow us to extend the results of the preceding section to $\leq^p_{\mathrm{btt}}$-completeness.

We write $f \circ g$ to denote the composition of the functions f and g. We define the composition of sets of functions by lifting the ordinary composition operator:

Definition 42. If C_1 and C_2 are two sets of functions then

$$C_1 \circ C_2 = \{f_1 \circ f_2 : f_1 \in C_1 \text{ and } f_2 \in C_2\}.$$

We define a companion operation to the composition of two functions. If x and y are two lists (we also treat scalars as singleton lists) then we write $x \,\|\, y$ to denote the concatenation of x and y as lists (not as strings).

Definition 43.

- If f_1 and f_2 are two functions, then

$$(f_1 \| f_2)(x) = f_1(x) \| f_2(x).$$

- If C_1 and C_2 are two sets of functions then

$$C_1 \| C_2 = \{f_1 \| f_2 : f_1 \in C_1 \text{ and } f_2 \in C_2\}.$$

Note that the $\|$ operator is associative.

The following obvious result from [14] shows the connection between the bounded query classes and the $\circ$ and $\|$ operators.

Lemma 44.

i. $\mathrm{PF}^X_{(j+k)\text{-T}} = \mathrm{PF}^X_{j\text{-T}} \circ \mathrm{PF}^X_{k\text{-T}}$.

ii. $\mathrm{PF}^X_{(j+k)\text{-tt}} = \mathrm{PF} \circ (\mathrm{PF}^X_{j\text{-tt}} \| \mathrm{PF}^X_{k\text{-tt}})$.

Along the same lines, it is easy to see that

$$\mathrm{PF}^A_{n_r\text{-tt}} \circ \cdots \circ \mathrm{PF}^A_{n_1\text{-tt}}$$

is the class of problems that can be solved in polynomial time by making n_1 parallel queries to A, followed by n_2 parallel queries to A, ..., followed by n_r parallel queries to A. Similarly,

$$\mathrm{PF} \circ (\mathrm{PF}^A_{n_r\text{-tt}} \| \cdots \| \mathrm{PF}^A_{n_1\text{-tt}})$$

is the class of problems that can be solved in polynomial time by making n_1 parallel queries to A, simultaneous with n_2 parallel queries to A, ..., simultaneous with n_r parallel queries to A. (See [7] for formalities and generalizations.) These observations lead to a neat tradeoff.

Lemma 45 (Synchronization).

$$\underbrace{\mathrm{PF}^A_{k\text{-T}} \| \cdots \| \mathrm{PF}^A_{k\text{-T}}}_{j} \subseteq \underbrace{\mathrm{PF}^A_{j\text{-tt}} \circ \cdots \circ \mathrm{PF}^A_{j\text{-tt}}}_{k}$$

Proof: The left-hand side is the class of all sets than can be computed by j polynomial-time Turing machines, working in parallel, that each make at most k serial queries to A. If we force the Turing machines to synchronize before each query, then we can make all the queries in k rounds of j parallel queries. ∎

Theorem 46. *If* $\mathrm{PF}^A_{j\text{-tt}} \subseteq \mathrm{PF}^A_{k\text{-T}}$ *then for every natural number* t

$$\mathrm{PF}^A_{j^t\text{-tt}} \subseteq \mathrm{PF}^A_{k^t\text{-T}}.$$

Proof: By induction on t. The theorem is identically true in the base case ($t = 0$). Assume that the theorem is true for $t - 1$.

$$\begin{aligned}
\mathrm{PF}^A_{j^t\text{-tt}} &= \mathrm{PF} \circ \underbrace{\mathrm{PF}^A_{j^{t-1}\text{-tt}} \,\|\, \cdots \,\|\, \mathrm{PF}^A_{j^{t-1}\text{-tt}}}_{j} \\
&\quad \text{because we can make the queries in } j \text{ groups of } j^{t-1} \text{ queries} \\
&\subseteq \mathrm{PF} \circ \underbrace{\mathrm{PF}^A_{k^{t-1}\text{-T}} \,\|\, \cdots \,\|\, \mathrm{PF}^A_{k^{t-1}\text{-T}}}_{j} \qquad \text{by the induction hypothesis} \\
&\subseteq \mathrm{PF} \circ \underbrace{\mathrm{PF}^A_{j\text{-tt}} \circ \cdots \circ \mathrm{PF}^A_{j\text{-tt}}}_{k^{t-1}} \qquad \text{by the Synchronization Lemma} \\
&\subseteq \mathrm{PF} \circ \underbrace{\mathrm{PF}^A_{k\text{-T}} \circ \cdots \circ \mathrm{PF}^A_{k\text{-T}}}_{k^{t-1}} \qquad \text{by assumption} \\
&= \mathrm{PF}^A_{k(k^{t-1})\text{-T}} \\
&= \mathrm{PF}^A_{k^t\text{-T}}.
\end{aligned}$$

∎

Corollary 47. *If A is not p-terse, then there exists $r < 1$ such that for all sufficiently large j*

$$\mathrm{PF}^A_{j\text{-tt}} \subseteq \mathrm{PF}^A_{j^r\text{-T}}.$$

Proof: Suppose that $\mathrm{PF}^A_{(k+1)\text{-tt}} \subseteq \mathrm{PF}^A_{k\text{-T}}$. Choose t such that

$$(k+1)^t < j \leq (k+1)^{t+1}.$$

Then

$$\begin{aligned}
\mathrm{PF}^A_{j\text{-tt}} &\subseteq \mathrm{PF}^A_{(k+1)^{t+1}\text{-tt}} \\
&\subseteq \mathrm{PF}^A_{k^{t+1}\text{-T}} \qquad \text{by the preceding Theorem.}
\end{aligned}$$

Furthermore, $t < \log_{k+1} j$, because we chose t such that $(k+1)^t < j$. Therefore,

$$\begin{aligned}
k^{t+1} &< k^{1+\log_{k+1} j} \\
&= k\left(k^{\log_{k+1} j}\right) \\
&= k\left(j^{\log_{k+1} k}\right) \\
&< j^{\frac{1}{2}(1+\log_{k+1} k)},
\end{aligned}$$

for sufficiently large j. Consequently,

$$\mathrm{PF}^A_{j\text{-tt}} \subseteq \mathrm{PF}^A_{(j^{\frac{1}{2}(1+\log_{k+1} k)})\text{-T}},$$

for sufficiently large j. ∎

It is not known whether the preceding result is as tight as possible.

Lemma 48.

- *Assume that* $A \equiv^p_{\rm btt} B$. *Then* A *is p-terse iff* B *is p-terse.*
- *Assume that* $A \leq^p_{\rm btt} B$. *If* A *is p-superterse then* B *is p-superterse.*

Proof:

- Assume that $A \equiv^p_{k\text{-tt}} B$. It suffices to show that if A is not i-p-terse for some value of i, then B is not j-p-terse for some value of j. Suppose that A is not i-p-terse. By Theorem 46 there exists a real number $r < 1$ such that if kj is sufficiently large then
$$\mathrm{PF}^A_{kj\text{-tt}} \subseteq \mathrm{PF}^A_{(kj)^r\text{-tt}}.$$
Choose j large enough so that $(kj)^r \leq \lfloor (j-1)/k \rfloor$. Then
$$\mathrm{PF}^A_{kj\text{-tt}} \subseteq \mathrm{PF}^A_{\lfloor (j-1)/k \rfloor\text{-tt}}.$$
Since $B \leq^p_{k\text{-tt}} A$,
$$\mathrm{PF}^B_{j\text{-tt}} \subseteq \mathrm{PF}^A_{\lfloor (j-1)/k \rfloor\text{-tt}}.$$
Since $A \leq^p_{k\text{-tt}} B$,
$$\mathrm{PF}^B_{j\text{-tt}} \subseteq \mathrm{PF}^B_{(j-1)\text{-tt}}.$$
Thus B is not j-p-terse.
- Assume that $A \leq^p_{k\text{-tt}} B$. It suffices to show that if B is not i-p-superterse for some value of i then A is not j-p-superterse for some value of j. Suppose that B is not i-p-superterse. Then, as shown in [9], there exists a set X such that
$$\mathrm{PF}^B_{kj\text{-tt}} \subseteq \mathrm{PF}^X_{\lceil i \log(kj) \rceil\text{-tt}}.$$
Choose j large enough so that $i \log(kj) \leq j-1$. Then
$$\mathrm{PF}^B_{kj\text{-tt}} \subseteq \mathrm{PF}^X_{(j-1)\text{-tt}}.$$
Since $A \leq^p_{k\text{-tt}} B$,
$$\mathrm{PF}^B_{j\text{-tt}} \subseteq \mathrm{PF}^X_{(j-1)\text{-tt}}.$$
Thus B is not j-p-superterse.

∎

Corollary 49.

i. If B *is* $\leq^p_{1\text{-tt}}$*-equivalent to a pbtt-cylinder then* B *is i-p-terse iff* B *is i-p-superterse.*

ii. If B *is* $\leq^p_{\rm btt}$*-equivalent to a pbtt-cylinder then* B *is p-terse iff* B *is p-superterse.*

Proof:

i. Assume that $B \equiv_{\text{1-tt}} A$, where A is a pbtt-cylinder. Obviously, B is i-p-terse iff A is i-p-terse iff A is i-p-superterse iff B is i-p-superterse.

ii. Assume that $B \equiv_{\text{btt}} A$, where A is a pbtt-cylinder. By the preceding lemma, B is p-terse iff A is p-terse iff A is p-superterse iff B is p-superterse.

∎

Theorem 50. *Let $\mathcal{C}$ be one of the following classes:*

- DTIME($T(n)$) *for* $T(n) \geq n$.
- NTIME($T(n)$) *for* $T(n) \geq n$.
- DSPACE($S(n)$) *for* $S(n) \geq \log n$.
- NSPACE($S(n)$) *for* $S(n) \geq \log n$.

i. If B is $\leq^p_{\text{1-tt}}$-complete for $\mathcal{C}$ then B is i-p-superterse iff B is i-p-terse.

ii. If B is $\leq^p_{\text{btt}}$-complete for $\mathcal{C}$ then B is p-superterse iff B is p-terse.

Proof: Let A be $\leq^p_{\text{m}}$-complete for one of the complexity classes above. Then A is a pbtt-cylinder. The conclusions follow from the preceding corollary. ∎

Corollary 51. *Let A be $\leq^p_{\text{btt}}$ complete for* NP, PSPACE, *or exponential time. Then A is p-terse iff A is p-superterse.*

We can obtain an even stronger result for NP-complete, PSPACE-complete, and exponential-time-complete sets.

Theorem 52. *Let R be a ptt-cylinder.*

i. $(\exists X)[\mathrm{PF}^R_{q_1(n)\text{-tt}} \subseteq \mathrm{PF}^X_{q_2(n)\text{-T}}]$ *iff* $\mathrm{PF}^R_{q_1(n)\text{-tt}} \subseteq \mathrm{PF}^R_{q_2(n)\text{-T}}$.

ii. R is q-p-superterse iff R is q-p-terse.

Proof: Similar to the proof of Theorem 37. ∎

Corollary 53. *Let A be $\leq^p_{\text{btt}}$-complete for* NP, PSPACE, *or exponential time. Then A is q-p-terse iff A is q-p-superterse.*

8. Open Questions

We have shown that all NP-hard sets (under $\leq_{tt}^{p}$-reductions) are p-superterse unless certain complexity classes coincide. Several questions remain.

- Find necessary and sufficient conditions for all NP-complete sets to be p-superterse.
- Is there an oracle B such that $\mathrm{P}^B \neq \mathrm{NP}^B$ but all NP^B complete sets fail to be p^B-superterse?
- Is there a relativized world in which $\mathrm{P} = \mathrm{NP}$, but for all NP complete sets A, $\mathrm{PF}^A_{\log n\text{-T}} = \mathrm{PF}^A_{(\log n - 1)\text{-T}}$?
- If PSPACE-complete sets are not p-superterse, then we can conclude that the PH collapses. Can we also conclude that PSPACE collapses? Because only sets in NP can be helped, this question may require a new approach.

9. Summary of Interesting Relations Among Bounded Query Classes

The following relationships are now known for well-behaved $q(n)$:

When q is a constant,

$\mathrm{PF}^{\mathrm{NP}}_{q\text{-tt}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{(q+1)\text{-tt}}$	unless P = NP.

When $q(n) \leq (1-\epsilon)\log n$,

$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-tt}}$	$\not\subseteq$	$\mathrm{PF}^{\mathrm{NP}}_{(q(n)-1)\text{-T}}$	unless UniqueSAT and UnSAT are p-separable.
$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{(q(n)+1)\text{-T}}$	unless P = NP.
$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-T}}$	unless NP/poly = co-NP/poly.
$\mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-2)\text{-tt}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	unless NP/poly = co-NP/poly.
$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{2^{q(n)}\text{-tt}}$	unless NP/poly = co-NP/poly.

When $q(n) = O(\log n)$,

$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-tt}}$	$\not\subseteq$	$\mathrm{PF}^{\mathrm{NP}}_{(q(n)-1)\text{-T}}$	unless $\mathrm{RTIME}(n^{\mathrm{polylog}\, n}) = \mathrm{NTIME}(n^{\mathrm{polylog}\, n})$, etc.
$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{n\text{-T}}$	unless P = NP.
$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-T}}$	$=$	$\mathrm{P}^{\mathrm{NP}}_{(2^{q(n)}-1)\text{-tt}}$.	

When $q(n) \leq n^{1-\epsilon}$,

$\mathrm{P}^{\mathrm{NP}}_{q(n)\text{-tt}}$	$\subset$	$\mathrm{P}^{\mathrm{NP}}_{(q(n)+1)\text{-tt}}$	unless NP/poly = co-NP/poly.

When $q(n) = o(n/\log n)$,

$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-tt}}$	$\not\subseteq$	$\mathrm{PF}^{\mathrm{NP}}_{(q(n)-1)\text{-T}}$	unless UniqueSAT and UnSAT are p-separable in $\mathrm{DTIME}(n^{O(q(n))})$
$\mathrm{PF}^{\mathrm{NP}}_{q(n)\text{-T}}$	$\subset$	$\mathrm{PF}^{\mathrm{NP}}_{n\text{-T}}$	unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{q(n^{O(1)})})$.

The results in this table are due directly, or indirectly, to a large number of researchers. In order to keep the table concise, the references are omitted. The table in the introduction contains references.

10. Acknowledgments

This paper has benefited from helpful suggestion's made by Ron Book and others during the 1988 Workshop on Computational Complexity Theory held in the department of mathematics at U. C. Santa Barbara. I would like especially to thank Lane Hemachandra for suggesting the extension to UP-hard sets. I also thank Bill Gasarch for proofreading and helpful discussions.

References

[1] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[2] L. Adleman and K. Manders. Reducibility, randomness, and intractibility. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, pages 151–153, 1977.

[3] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings of the 5th Annual Conference on Structure in Complexity Theory*, pages 232–243. IEEE Computer Society Press, July 1990.

[4] A. Amir and W. I. Gasarch. Polynomial terse sets. *Information and Computation*, 77:37–56, Apr. 1988.

[5] J. L. Balcázar. Only smart oracles help. Manuscript, July 1987.

[6] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, New York, 1988.

[7] R. Beigel. *Query-Limited Reducibilities*. PhD thesis, Stanford University, 1987. Also available as Report No. STAN-CS-88-1221.

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

[9] 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.

[10] R. Beigel. When are $k+1$ queries better than k? TR 88-06, The Johns Hopkins University, Dept. of Computer Science, 1988.

[11] R. Beigel. Why are counting problems hard to approximate? TR 88-05, The Johns Hopkins University, Dept. of Computer Science, 1988.

[12] R. Beigel. On the relativized power of additional accepting paths. In *Proceedings of the 4th Annual Conference on Structure in Complexity Theory*, pages 216–224. IEEE Computer Society Press, June 1989.

[13] R. Beigel. Bi-immunity results for cheatable sets. *Theoretical Comput. Sci.*, 73:249–263, 1990.

[14] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Comput. Sci.*, 84, 1991.

[15] R. Beigel and W. I. Gasarch. On the complexity of finding the chromatic number of a recursive graph I: The bounded case. *Annals of Pure and Applied Logic*, 45(1):1–38, Nov. 1989.

[16] R. Beigel and W. I. Gasarch. On the complexity of finding the chromatic number of a recursive graph II: The unbounded case. *Annals of Pure and Applied Logic*, 45(3):227–247, Dec. 1989.

[17] R. Beigel, W. I. Gasarch, J. T. Gill, and J. C. Owings, Jr. Terse, superterse, and verbose sets. *Information and Computation*. To appear.

[18] R. Beigel, W. I. Gasarch, and L. Hay. Bounded query classes and the difference hierarchy. *Archive for Mathematical Logic*, 29(1):69–84, Dec. 1989.

[19] R. Beigel, W. I. Gasarch, and J. C. Owings, Jr. Nondeterministic bounded query reducibilities. *Annals of Pure and Applied Logic*, 41:107–118, 1989.

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

[21] M. Blum, 1988. Personal communication.

[22] R. V. Book. Bounded query machines: On NP and PSPACE. *Theoretical Comput. Sci.*, 15:27–39, 1981.

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

[24] R. V. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM J. Comput.*, 13:461–487, 1984.

[25] R. V. Book and C. Wrathall. Bounded query machines: On NP() and NPQUERY(). *Theoretical Comput. Sci.*, 15:41–50, 1981.

[26] S. R. Buss and L. E. Hay. On truth table reducibility to SAT. *Information and Computation*, 91(1):86–102, Mar. 1991.

[27] J. Cai. *On Some Most Probable Separations of Complexity Classes*. PhD thesis, Cornell University, Ithaca, NY, 1986.

[28] J. Cai, T. Gundermann, J. Hartmanis, L. A. Hemachandra, V. Sewelson, K. W. Wagner, and G. Wechsung. The Boolean hierarchy I: structural properties. *SIAM J. Comput.*, 17(6):1232–1252, Dec. 1988.

[29] J. Cai and L. A. Hemachandra. The Boolean hierarchy: Hardware over NP. In *Structure in Complexity Theory*, pages 105–124. Springer-Verlag, June 1986. Lecture Notes in Computer Science 223.

[30] J. Cai and L. A. Hemachandra. Enumerative counting is hard. *Information and Computation*, 82(1):34–44, July 1989.

[31] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

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

[33] W. I. Gasarch and S. R. Pearlman. The complexity of optimization problems related to partition. TR 2028, University of Maryland, Dept. of Computer Science, 1988.

[34] 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. TR 87-06-02, Department of Computer Science, University of Washington, Seattle, June 1987. An extended abstract appeared in *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory*, IEEE Computer Society Press, June 1987, pp. 50–59.

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

[36] J. Grollman and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, Apr. 1988.

[37] C. G. Jockusch, Jr. Semirecursive sets and positive reducibility. *Transactions of the AMS*, 131:420–436, May 1968.

[38] J. Kadin. $\mathrm{P}^{\mathrm{NP}[\log n]}$ and sparse Turing-complete sets for NP. In *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory*, pages 33–40. IEEE Computer Society Press, June 1987.

[39] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM J. Comput.*, 17(6):1263–1282, Dec. 1988.

[40] K.-I. Ko. On adaptive versus nonadaptive query machines. Manuscript.

[41] K.-I. Ko. On helping by robust oracle machines. *Theoretical Comput. Sci.*, 52:15–36, 1987.

[42] J. Köbler, U. Schöning, and K. W. Wagner. The difference and truth-table hierarchies for NP. *RAIRO Theoretical Informatics and Applications*, 21:419–435, 1987.

[43] M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.

[44] S. A. Kurtz. On the random oracle hypothesis. *Information and Control*, 57(1):40–47, Apr. 1983.

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

[46] A. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[47] J. C. Owings, Jr. A cardinality version of Beigel's Nonspeedup Theorem. *Journal of Symbolic Logic*, 54(3):761–767, Sept. 1989.

[48] C. Rackoff. Relativized questions involving probabilistic algorithms. *J. ACM*, 29(1):261–268, Jan. 1982.

[49] S. Ruby and P. C. Fischer. Translational methods and computational complexity. In *Proceedings of the Sixth Annual IEEE Symposium on Switching, Circuit Theory, and Logical Design*, pages 173–178, 1965.

[50] D. A. Russo. *Structural Properties of Complexity Classes*. PhD thesis, University of California at Santa Barbara, Mar. 1985.

[51] U. Schöning and K. W. Wagner. Collapsing oracle hierarchies, census functions, and logarithmically many queries. TR 140, Universitat Augsburg, Institut fur Mathematik, 1987.

[52] A. L. Selman. Analogues of semirecursive sets and effective reducibilities to the study of NP complexity. *Information and Control*, 52(1):36–51, Jan. 1982.

[53] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Comput. Sci.*, 3:1–22, 1977.

[54] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to p-selective sets. *Mathematical Systems Theory*, 1991. to appear.

[55] L. G. Valiant. The relative complexity of checking and evaluating. *Inf. Process. Lett.*, 5:20–23, 1976.

[56] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Comput. Sci.*, 47:85–93, 1986.

[57] K. W. Wagner. More complicated questions about maxima and minima and some closures of NP. In *Proceedings of the 13th International Colloquium on Automata, Languages, and Programming*, pages 434–443, 1986. Volume 226 of *Lecture Notes in Computer Science*.

[58] K. W. Wagner. Log query classes. TR 145, Universitat Augsburg, Institut fur Mathematik, 1987.

[59] K. W. Wagner. Number-of-query hierarchies. TR 4, Universitat Augsburg, Institut fur Mathematik, 1989.

[60] G. Wechsung and K. Wagner. On the Boolean closure of NP. In *Proceedings of the 1985 International Conference on Fundamentals of Computation Theory*, pages 485–493. Springer-Verlag, 1985. Volume 199 of *Lecture Notes in Computer Science*.

[61] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Comput. Sci.*, 26:287–300, 1983.