Some Connections between Bounded Query Classes and Non-Uniform Complexity

Amihood Amir* Richard Beigel† William I. Gasarch‡

Georgia Inst. of Tech. Yale University University of Maryland

November 15, 1996

Abstract

Consider the function $F_k^A(x_1, \ldots, x_k) = A(x_1) \cdots A(x_k)$. We show that if F_k^A can be computed with less than k queries to some set X then $A \in \mathrm{P/poly}$. A generalization of this result has applications to bounded query classes, circuits, and enumerability. In particular we obtain the following. (1) Assuming $\Sigma_3^p \neq \Pi_3^p$ the hierarchy of functions computed with a bounded number of queries to SAT does not collapse through $O(\log n)$ queries. Similar results are obtained for many other classes. (2) Computing F_k^A with an oracle circuit with $k-1$ queries to some set X is about as hard as computing without oracles. (3) Assuming that $\mathrm{PH} \neq \Sigma_4^p \cap \Pi_4^p$, and $\epsilon < 1$, #SAT is not 2^{n^ϵ}-enumerable. Similar results are obtained for many other functions.

1. Introduction

The standard complexity classes (e.g. P, NP) were defined to classify sets. To classify functions several researchers [33, 51] have defined complexity classes based on the *number of queries* that a polynomial-time algorithm has to make to a particular set. For example, Krentel [51] has shown that to compute the chromatic number of a graph, $\Theta(\log n)$ queries to SAT are necessary and sufficient (assuming $\mathrm{P} \neq \mathrm{NP}$). Krentel [51] and Gasarch [33] have classified many functions in terms of queries to SAT. (See [32] for a survey of such results.)

*College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280. Email: amir@cc.gatech.edu.

Supported in part by NSF grants CCR-8803641, CCR-9223699, and IRI-9013055. Part of this research was performed at the University of Maryland at College Park.

†Dept. of Computer Science, P.O. Box 208285, Yale Station, New Haven, CT 06520-8285. Email: beigel-richard@cs.yale.edu.

Supported in part by NSF grants CCR-8808949 and CCR-8958528. Parts of this research were performed at the Johns Hopkins University.

‡Dept. of Computer Science and Institute for Advanced Studies, University of Maryland, College Park, MD 20742. Email: gasarch@cs.umd.edu.

Supported in part by NSF grants CCR-8803641 and CCR-9020079.

More generally, we have defined a complexity measure by the number of queries made by a polynomial-time algorithm to a set A [6, 11]. The corresponding complexity classes are called "bounded query classes." While the results obtained in this framework do not depend on special properties of the oracle set, they often yield corollaries about computations with a bounded number of queries to an NP oracle [14, 12]; in addition they yield answers to questions ostensibly unrelated to counting oracle queries [13].

A natural function to look at in this context is the following:

Definition 1.1. If $A \subseteq \{0,1\}^*$ and $k \in \mathsf{N}$ then $F_k^A : (\Sigma^*)^k \rightarrow \{0,1\}^k$ is defined by

$$F_k^A(x_1, \ldots, x_k) = A(x_1) \cdots A(x_k),$$

where $A(x)$ is the characteristic function of A: 1 if $x \in A$, 0 otherwise.

The function F_k^A is easily seen to be computable with k queries to A. When can it be computed with $k-1$ queries to A? to some other set X? The first theme of this paper is that if F_k^A can be computed with $k-1$ queries to A (or to some set X) then A is easy in some sense. Along these lines we show the following:

- If $(\exists k)(\exists X)$ such that $F_{2^k}^A$ can be computed with k queries to X then $A \in$ P/poly. In addition, $A \in \mathrm{EL}_2$, the second level of the extended low hierarchy (defined in [9] and in Definition 3.1 of this paper). (Corollary 3.6)

- If $(\exists k)(\exists X)$ such that F_k^A can be computed with $k-1$ queries to X then $A \in$ P/poly. (Theorem 4.2)

These theorems, and the techniques used to obtain them, are applied to extend (though not generalize) previously known theorems about bounded query classes, circuits, and enumerability. We state the previously known theorems and our extensions (for readability we do not state the strongest form of either the previous results or of ours).

- Krentel [51] showed that if P $\neq$ NP then, for any increasing $f \in$ PF, $f(n) \leq (1-\epsilon)\log n$, $f(n)$ queries to SAT are more powerful than $f(n)-1$. [1]

  We show that if $\Sigma_3^p \neq \Pi_3^p$ then for any $f(n) = O(\log n)$, $f(n)$ queries to SAT are more powerful than $f(n)-1$ (Corollary 5.2).

- Cai [23] showed that if a constant depth oracle circuit computes k length-n instances of PARITY, with only $k-1$ queries, it must have size $2^{n^{\Omega(1)}}$. We show that if there is a circuit of size s and depth d that computes $F_{k(n)}^{A^{=n}}$ while making only $k-1$ queries to some set X then there is a circuit of size $n^{O(k)}s$ and depth $d+O(1)$ that computes $F_{k(n)}^{A^{=n}}$ without making any queries. By applying the lower bound on parity ([21, 38, 77]) we easily obtain Cai's result.

[1] Krentel actually showed this just for $f(n) \leq \frac{1}{2}\log n$, but his proof can be modified to $f(n) \leq (1-\epsilon)\log n$. See [12].

- Cai and Hemachandra [27] proved that if $\mathrm{P} \neq \mathrm{P}^{\#\mathrm{P}}$ then, for all $\epsilon < 1$, #SAT is not n^ϵ-enumerable, i.e., there is no polynomial algorithm that, with formula ψ as input, produces n^ϵ numbers one of which is $\#\mathrm{SAT}(\psi)$. We extend their definition of enumerability to superpolynomial functions. Using the theorems of this paper we can prove two powerful theorems that extend their results. Informally we show the following.

  (a) For many functions g $(\exists\delta \leq 1)(\forall\epsilon < \delta)[g\ \ n^\epsilon\text{-enumerable} \Rightarrow g \in \mathrm{PF}]$.

  (b) If #GA (counts the number of automorphisms of a graph) is k-enumerable then $\#\mathrm{GA} \in \mathrm{PF}$.

  (c) For many functions g $(\exists\delta)(\forall\epsilon < \delta)[g\ \ 2^{n^\epsilon}\text{-enumerable} \Rightarrow g \in \mathrm{PF}^{\Sigma_4^p \cap \Pi_4^p}]$. (If g is the function that computes the permanent of a 0-1 matrices then this theorem provides a lower bound that matches the upper bound.)

  (d) $(\forall\epsilon < 1)[\#\mathrm{SAT}\ \ 2^{n^\epsilon}\text{-enumerable} \Rightarrow \#\mathrm{SAT} \in \mathrm{PF}^{\Sigma_4^p \cap \Pi_4^p}]$.

  (e) $(\forall k)[\#\mathrm{SAT}\ \ n^k\text{-enumerable} \Rightarrow \#\mathrm{SAT} \in \mathrm{PF}]$. (Cai and Hemachandra [26] have obtained this result independently by different methods.)

The second theme of this paper is to look at the *class* of sets A such that F_k^A can be computed with $k-1$ queries to some X (and variants of this class). We examine closure properties, structural properties, and relations to other notions in complexity theory. A more detailed description of our results requires more definitions, hence it is deferred to the introduction of Section 8.

As an aid to the reader we provide a table of contents for this paper:

2. Definitions and Useful Facts

Section 2.1 reviews some standard definitions from complexity theory. Section 2.2 reviews some definitions and facts relevant to bounded query classes.

Throughout this paper we assume that $\Sigma = \{0,1\}$ and all sets are subsets of Σ^* unless otherwise noted.

2.1. Definitions from the Literature

Definition 2.1. PF is the class of functions that can be computed in polynomial time. If A is any set then PF^A is the class of functions that can be computed in polynomial time using oracle A.

Definition 2.2. A set A is *polynomial truth table reducible* to B, denoted $A \leq_{\mathrm{tt}}^{p} B$, if there exists $f \in \mathrm{PF}$ such that (i) for all x, $f(x) = \langle y_1, \ldots, y_m, \varphi\rangle$ where $y_i \in \Sigma^*$ and φ is an m-place Boolean formula, and (ii) $x \in A$ iff $\varphi(b_1, \ldots, b_m)$ is true, where b_i is the truth value of "$y_i \in B$." The number m is the *norm* of $f(x)$. Two sets A and B are *polynomial truth table equivalent* if $A \leq_{\mathrm{tt}}^{p} B$ and $B \leq_{\mathrm{tt}}^{p} A$.

Definition 2.3. Let k be a constant. $A \leq_{k\text{-tt}}^{p} B$ if $A \leq_{\mathrm{tt}}^{p} B$ via a function f such that, for all x, $f(x)$ has norm $\leq k$. $A \leq_{\mathrm{btt}}^{p} B$ if there exists a k such that $A \leq_{k\text{-tt}}^{p} B$.

Definition 2.4. If $\mathcal{C}$ is a class of sets then A is $\leq_{\mathrm{tt}}^{p}$-hard for $\mathcal{C}$ if for every $B \in \mathcal{C}$, $B \leq_{\mathrm{tt}}^{p} A$. The notions of $\leq_{\mathrm{btt}}^{p}$-hard and $\leq_{k\text{-tt}}^{p}$-hard are defined similarly.

Notation 2.5. If A and B are sets then $A \oplus B$ is the set $\{1x \mid x \in A\} \cup \{0x \mid x \in B\}$.

Pippenger [61] showed that the class of languages recognized by polynomial sized circuits is P/poly (though it was not quite defined yet). This inspired Karp and Lipton [46] to define general advice classes.

Definition 2.6. Let $\mathcal{C}$ be a class of functions. A function f is in $\mathcal{C}$/poly if there exists $g \in \mathcal{C}$ and a function h with polynomial size output such that $(\forall n)(\forall x)[|x| \leq n \Rightarrow f(x) = g(x, h(0^n))]$. The function h is called the *advice function* and $h(0^n)$ is called the *advice for strings of length* $\leq n$. (This definition is equivalent to asking for advice for strings of length exactly n. We will use both definitions.)

Cai and Hemachandra [27] defined $b(n)$-enumerability as follows.

Definition 2.7. Let $b(n)$ be a function with range N. A function f is $b(n)$-*enumerable* if there exists $e \in \mathrm{PF}$, $e : \Sigma^* \rightarrow (\Sigma \cup \{\%\})^*$, such that, for all x, $e(x)$ is a list of at most $b(|x|)$ elements of Σ^*, separated by %, at least one of which is $f(x)$.

This definition only makes sense if $b(|x|)$ is bounded by a polynomial. We define a more general notion of enumerability that allows superpolynomial b.

Definition 2.8. Let $b(n)$ be a function with range N. A function f is $b(n)$-*enumerable* if there exists $e \in \mathrm{PF}$, $e : \Sigma^* \times \mathsf{N} \rightarrow \Sigma^*$, such that, for all x, there exists an $i < b(|x|)$ such that $e(x, i) = f(x)$. (We need to have $i < b(|x|)$ instead of $i \leq b(|x|)$ since the natural numbers N include 0.) We assume the second input to e is written in binary.

Definition 2.9. The quantifier $\overset{\infty}{\forall} x$ means "for all but a finite number of $x \in \mathsf{N}$." The quantifier $\overset{\infty}{\exists} x$ means "for an infinite number of $x \in \mathsf{N}$."

Definition 2.10. Let $i \geq 1$. QBF_i is the set of true quantified Boolean formulas that have an $\exists$ as the leftmost quantifier and make at most $i-1$ alternations of quantifiers. Note that $\mathrm{QBF}_1 = \mathrm{SAT}$. It is well known that QBF_i is self-reducible and complete for Σ_i^p [68, 76]. Let $\mathrm{QBF} = \cup_{i=1}^{\infty} \mathrm{QBF}_i$. It is well known that QBF is self-reducible and complete for PSPACE [69].

Notation 2.11. The *size* of a circuit is the number of gates. The *depth* of a circuit is the length of the longest path from an input to an output. A circuit that has size $s(n)$ and depth $d(n)$ is called an $(s(n), d(n))$ *circuit family.* We will continue the tradition of abusing notation by calling a circuit family just a circuit.

Notation 2.12. We will assume that our tupling function $\langle\rangle$ from $\cup_{m=0}^{\infty} (\Sigma^*)^m$ to Σ^* has the following properties.

i. The function is polynomial time computable.

ii. $|\langle x_1, \ldots, x_m \rangle| = O(\sum_{i=1}^{m} |x_i|)$.

iii. $(\forall x_1, \ldots, x_m)(\forall i)[|\langle x_1, \ldots, x_m \rangle| = |\langle 0^{|x_1|}, \ldots, 0^{|x_m|} \rangle| > |x_i|]$.

2.2. Bounded Query Classes

Bounded query classes were defined in [6, 14] as follows.

Definition 2.13. If A is a set and $j(n)$ is a function from $\mathbb{N}$ to $\mathbb{N}$ then $\mathrm{PF}^{A}_{j(n)\text{-T}}$ is the class of functions that can be computed by a polynomial time oracle Turing machine that makes, on inputs of length n, at most $j(n)$ queries to oracle A.

The queries involved in the definition of $\mathrm{PF}^{A}_{k\text{-T}}$ are *serial* or, as they are called by Book and Ko [20], adaptive. We also define *parallel* queries (nonadaptive).

Definition 2.14. If A is a set and $j(n)$ is a function from $\mathbb{N}$ to $\mathbb{N}$ then $\mathrm{PF}^{A}_{j(n)\text{-tt}}$ is the class of functions that can be computed by a polynomial time oracle Turing machine that, on inputs of length n, prepares a list of the $j(n)$ queries it is going to make to A *before* actually making any of them.

If f is computable by making $j(n)$ oracle queries, then there are only $2^{j(n)}$ *possible* values for the result of f. Conversely, if there are only $2^{j(n)}$ possible values of the result of f, then $j(n)$ queries to some oracle suffice to distinguish the correct result. The informal notion of *possibility* is made precise by using the notion of enumerability as defined in Section 2.1.

The connection between bounded queries and enumerability is formalized by the following two facts from [11, Lemma 3.2]. We include a proof sketch for completeness.

Fact 2.15. *Let f be any function. Let $j(n) \in \mathrm{PF}$. The following are equivalent:*

i. There exists X such that $f \in \mathrm{PF}^{X}_{j(n)\text{-T}}$.

ii. f is $2^{j(n)}$-enumerable.

iii. There exists Y such that $f \in \mathrm{PF}^{Y}_{j(n)\text{-tt}}$. (If $2^{j(n)}$ is bounded by a polynomial then $Y \in \mathrm{P}^{f}_{1\text{-tt}}$.)

Proof sketch:

$(i) \Rightarrow (ii)$: If $f \in \mathrm{PF}^{X}_{j(n)\text{-T}}$ via M^X then let $e(x,i)$ be defined as follows. If $i \geq 2^{j(|x|)}$ then output 0, else run $M^{()}(x)$ using the b^{th} bit of the $(i+1)^{\text{st}}$ element of $\{0,1\}^{j(|x|)}$ to answer the b^{th} query. For all x, some sequence of $j(|x|)$ bits provides the correct query answers for the $M^X(x)$ computation, hence there is an $i < 2^{j(|x|)}$ such that $e(x,i) = M^X(x) = f(x)$.

$(ii) \Rightarrow (iii)$: If f is $2^{j(n)}$-enumerable then let e be the function such that, for all x, there exists $i < 2^{j(|x|)}$ such that $e(x,i) = f(x)$. Let Y be the set of all tuples $\langle x, a\rangle$ such that the following is true: if i is the least number such that $f(x) = e(x,i)$ then the a^{th} bit of i is 1. On input x, asking the queries "$\langle x, 1\rangle \in Y$?", "$\langle x, 2\rangle \in Y$?",..., "$\langle x, j(|x|)\rangle \in Y$?" allows you to find the appropriate i such that $e(x,i) = f(x)$.

$(iii) \Rightarrow (i)$ is obvious. ∎

By plugging in F_k^A for f and $j(n) = j$ (a constant) into Fact 2.15 we obtain the following.

Fact 2.16. *Let $j, k \in \mathsf{N}$. The following are equivalent:*

i. There exists X such that $F_k^A \in \mathrm{PF}^X_{j\text{-T}}$.

ii. F_k^A is 2^j-enumerable.

iii. There exists $B \in \mathrm{P}^{F_k^A}_{1\text{-tt}} \subseteq \mathrm{P}^A_{k\text{-tt}}$ such that $F_k^A \in \mathrm{PF}^B_{j\text{-tt}}$.

We are interested in finding out when the function F_k^A *requires* k queries to A (or X).

Definition 2.17. Let $k \geq 2$. A set A is *k-p-terse* if $F_k^A \notin \mathrm{PF}^A_{(k-1)\text{-T}}$. A is *p-terse* if A is k-p-terse for all k. A set A is *k-p-superterse* if, for all sets X, $F_k^A \notin \mathrm{PF}^X_{(k-1)\text{-T}}$. A is *p-superterse* if A is k-p-superterse for all k.

At the opposite extreme, we are also interested in determining when the function F_k^A can be computed with far fewer than k queries.

Definition 2.18. Let k be a constant. A set A is *k-cheatable* if there exists a set X such that $F_{2^k}^A \in \mathrm{PF}^X_{k\text{-T}}$. We say A is *cheatable* iff A is k-cheatable for some constant k. Note that by Fact 2.16 a set is k-cheatable iff $F_{2^k}^A$ is 2^k-enumerable.

We state some (known) useful consequences of a set being k-cheatable. We include proof sketches for completeness.

Fact 2.19. *Let $k \in \mathsf{N}$ and $A \subseteq \Sigma^*$.*

i. If A is k-cheatable then $(\forall m)[F_m^A \in \mathrm{PF}^A_{(2^k-1)\text{-tt}}$ via a machine that, on input $\{x_1, \ldots, x_m\}$, queries a subset of $\{x_1, \ldots, x_m\}]$. This reduction is uniform in m. (Theorem 5.4.i of [14].)

ii. If A is k-cheatable then $(\exists X)(\forall m)[F_m^A \in \mathrm{PF}^X_{k\text{-T}}]$. (Theorem 5.4.ii of [14].)

iii. A is k-cheatable iff $(\forall m)[F_m^A$ is 2^k-enumerable$]$.

Proof sketch: We show that (i) holds. We first show this for $m = 2^k$. By Fact 2.16, $F_{2^k}^A$ is 2^k-enumerable. Hence on input $(x_1, \ldots, x_{2^k})$ we can generate 2^k possibilities for $F_{2^k}^A(x_1, \ldots, x_{2^k})$. By a combinatorial lemma [14, 30] we can find a subset of $\{x_1, \ldots, x_{2^k}\}$ of size $2^k - 1$ such that the status of those elements in A completely determines which possibility is actually $F_{2^k}^A(x_1, \ldots, x_{2^k})$. Hence we have established the $m = 2^k$ case. For general m we use induction and the result for $m = 2^k$.

We show that (ii) holds. Since A is k-cheatable there exists an X such that $F_{2^k}^A \in \mathrm{PF}^X_{k\text{-T}}$. This is the X we seek. For any value of m we may compute F_m^A with k

queries to X as follows: by (i) the computation reduces to finding out the membership of some 2^k-1 elements in A. Since $F_{2^k}^A \in \mathrm{PF}_{k\text{-T}}^X$, this can be done with k queries to X.

Item (iii) holds by using (ii) and Fact 2.16. ∎

The following fact about p-superterse sets will point the way to a generalization of p-terseness that is used in Theorem 4.4.

Fact 2.20. *If A is not p-superterse then there exists a $k \in \mathsf{N}$ and $w \in \mathrm{PF}$, $w : (\Sigma^*)^k \to \{0,1\}^k$, such that for all $x_1,\ldots,x_k$, $F_k^A(x_1,\ldots,x_k) \neq w(x_1,\ldots,x_k)$.*

Proof: Let k be such that there exists X, $F_k^A \in \mathrm{PF}_{(k-1)\text{-T}}^X$. By Fact 2.16 F_k^A is 2^{k-1}-enumerable. We compute w as follows: on input $(x_1,\ldots,x_k)$ we compute all 2^{k-1} possibilities for $F_A^k(x_1,\ldots,x_k)$ and output the least element (using lexicographical ordering) of $\{0,1\}^k$ that is not one of them. ∎

The converse of Fact 2.20 is also known, i.e., if such a w exists then A is not p-superterse (see [11]).

3. Cheatable Sets

We pursue the question "Which sets A can be cheatable?" In Section 3.1 we prove a very general theorem about queries. From it we obtain that if A is cheatable then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$ (see Definition 3.1). In Section 3.2 we show that if A is p-selective (NPMV-selective, NPSV-selective) then $A \in \mathrm{P/poly} \cap \mathrm{EL}_2$ ($A \in \mathrm{NP/poly}$, $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$). In Section 3.3 we show that if A is cheatable and self-reducible then $A \in \mathrm{P}$. As a corollary we obtain that NP-hard sets are not cheatable (unless P = NP). We then extend the techniques to a variation on k-cheatability where k is not a constant. The results for non-constant k are used in Section 7.4 to obtain results about n^ϵ-enumerability.

3.1. Circuits and Lowness

In a recursion-theoretic setting (no time bounds) it can be shown that a set A is cheatable iff A is recursive [15]. The situation is not so simple in the polynomial-time framework, because in fact there exist arbitrarily complex sets that are cheatable [6]. Since the cheatable sets we constructed in [6] are super-sparse (defined in [4]), we pursue a connection between sparseness and cheatability.

It is known that cheatable sets need not be sparse [6]. On the other hand, we show that all cheatable sets are in P/poly (which is equal to $\mathrm{P}^{\mathrm{SPARSE}}$). We obtain a further result which requires a definition from the literature.

Schöning [63] defined the low and high hierarchies to classify NP sets. Balcázar, Book, and Schöning [9] defined the extended low and extended high hierarchies to classify sets in general. (All these notions are analogous to similar concepts in recursion theory. See [54].)

Definition 3.1. Let $k \geq 1$ and A be a set. The set A is in EL_k, the k^{th} level of the extended low hierarchy, if $\Sigma_k^{p,A} \subseteq \Sigma_{k-1}^{p,A\oplus\mathrm{SAT}}$. The set A is in EH_k, the k^{th} level of the extended high hierarchy, if $\Sigma_k^{p,A\oplus\mathrm{SAT}} \subseteq \Sigma_k^{p,A}$.

It is known that $\mathrm{P/poly} \subseteq \mathrm{EL}_3$. In addition to obtaining that cheatable sets are in P/poly we also obtain that they are in EL_2. In fact we show containment in EL_2 for a larger class of languages that includes the cheatable sets and the p-selective sets (see Definition 3.11) as special cases.

We need the following lemma. The techniques to prove it are standard but it does not seem to be in the literature.

Lemma 3.2. *Let $A \in \mathrm{P/poly}$. Let $p(n)$ be the length of the advice. We assume that p is 1-1. Let C be a set of strings that satisfy the following properties:*

i. $w \in C \Rightarrow (\exists n)[|w| = p(n)]$.

ii. $w \in C \Rightarrow w$ *could serve as advice for the* P/poly *algorithm for* $A^{\leq n}$.

iii. $(\forall n)(\exists w)[|w| = p(n) \wedge w \in C]$.

If $C \in \mathrm{P}^{\mathrm{NP},A}$ *then* $A \in \mathrm{EL}_2$.

Proof: We show $\Sigma_2^{p,A} \subseteq \Sigma_1^{p,A\oplus\mathrm{SAT}}$. Let $B \in \Sigma_2^{p,A}$. There exists $R^A \in \mathrm{P}^A$ such that $B = \{x \mid (\exists^p y)(\forall^p z)[R^A(x,y,z)]\}$.

Let $R'(w,x,y,z)$ denote the result of trying to compute $R^A(x,y,z)$ by assuming that w is advice for A, hence answering all queries to A by using the P/poly algorithm for A and advice w.

Note that

$$B = \{x \mid (\exists^p w)(\exists^p y)[w \in C \wedge (\forall^p z)[R'(w,x,y,z)]]\}.$$

Since $C \in \mathrm{P}^{\mathrm{NP},A}$ and $R' \in \mathrm{P}$, $B \in \Sigma_1^{p,A\oplus\mathrm{SAT}}$. ∎

We state the next theorem in rather general terms. We will need this generality for Corollary 3.13.

Definition 3.3. Let X and Y be two disjoint sets. If Z is such that $X \subseteq Z$ and $Y \subseteq \overline{Z}$ then Z *separates* X *from* Y.

Theorem 3.4. *Let A be a set. Assume there exist k and Z such that the following hold.*

i. Z *separates* A^k *from* $\{(x_1,\ldots,x_k) : (\forall i \neq j)[x_i \neq x_j] \wedge |A \cap \{x_1,\ldots,x_k\}| = k-1\}$.

ii. $Z \in \mathrm{P}^A_{(k-1)\text{-}\mathrm{T}}$ *via an algorithm* $\mathcal{A}$ *that queries only components of the input.*

Then $A \in \mathrm{P/poly}$. *If in addition* $\overline{A}$ *satisfies the same condition then* $A \in \mathrm{EL}_2$.

The proof of membership in P/poly is largely inspired by [48].

Proof: Without loss of generality, assume that algorithm $\mathcal{A}$ always makes $k-1$ distinct queries to A (assuming distinct inputs) independent of the order of the inputs. Let $g(\{x_1, \ldots, x_k\})$ be the set of oracle queries asked by $\mathcal{A}$ on input $(x_1, \ldots, x_k)$. Note that $g(\{x_1, \ldots, x_k\})$ is a $(k-1)$-element subset of $\{x_1, \ldots, x_k\}$.

For any large set $S \subseteq A \cap \Sigma^n$ we will show how to find a $(k-1)$-element subset $X \subseteq S$ such that knowing X allows us to verify that a constant fraction of the elements of S are in A. By iterating this procedure $O(\log |S|)$ times we generate polynomial advice that allows us to verify that each element of S is in A. We will generate such advice for $S = A \cap \Sigma^n$; then we can test strings in Σ^n for membership in A because all the strings that can not be verified as being in A will be in $\overline{A}$.

In general let $[S]^k$ denote the set of k-element subsets of S, and let s denote $|S|$. For $S \subseteq A$ and $X = \{x_1, \ldots, x_{k-1}\} \in [S]^{k-1}$, we define the set of strings for which $(x_1, \ldots, x_{k-1})$ is useful advice:

$$\textsc{advisees}(X) = \{b \in S - X : g(X \cup \{b\}) = X\}.$$

Note that if we know $(x_1, \ldots, x_{k-1})$ then we can use algorithm $\mathcal{A}$ to verify $x \in A$ for every element $x \in \textsc{advisees}(X)$.

In the following calculation we use the fact that every $Y \in [S]^k$ can be interpreted as an $X \in [S]^{k-1}$ and a $b \in \textsc{advisees}(X)$ via $g(Y) = X$ and $b \in Y - X$.

$$\begin{aligned}
\sum_{X \in [S]^{k-1}} |\textsc{advisees}(X)| &= \sum_{X \in [S]^{k-1}} \sum_{b \in \textsc{advisees}(X)} 1 \\
&= \sum_{X \cup \{b\} \in [S]^k} 1 \\
&= |[S]^k|.
\end{aligned}$$

Therefore, there exists $X \in [S]^{k-1}$ such that

$$|\textsc{advisees}(X)| \geq \frac{|[S]^k|}{|[S]^{k-1}|} = \frac{\binom{s}{k}}{\binom{s}{k-1}} = \frac{s-k+1}{k}.$$

For this choice of $X = \{x_1, \ldots, x_{k-1}\}$, the tuple $(x_1, \ldots, x_{k-1})$ is useful advice for $(s-k+1)/k$ strings in addition to the members of X; hence it is useful advice for $(s-k+1)/k + k - 1 = (s + k^2 - 2k + 1)/k \geq s/k$ strings.

In particular, let $S_0 = A \cap \Sigma^n$. As shown in the preceding paragraph, there is a $(k-1)$-tuple X_1 that is useful advice for at least $|S_0|/k$ strings in S_0. Let $S_1 = S_0 - (\textsc{advisees}(X_1) \cup X_1)$. Then $|S_1| \leq |S_0|/c$, where $c = \frac{k}{k-1}$. Repeat the argument using S_1 in place of S_0 to obtain X_2 and S_2. Repeat for p iterations, stopping when $|S_p| < k$. Then $p \leq \log_c(|\Sigma|^n) = O(n)$. Let d be such that $p \leq dn$. Note that d is independent of n and of what is happening at this stage.

Our advice for strings of length n consists of the sets $X_1, \ldots, X_p$ and S_p. This advice contains at most $(p+1)(k-1)(n)$ bits, which is $O(n^2)$ because k is a constant and $p = O(n)$.

If y is a string of length n, then we determine if $y \in A$ as follows.

Step 1: If $y \in X_1 \cup \cdots \cup X_p \cup S_p$ then output(YES) and halt.

Step 2: For $i = 1$ to p

(a) Let z_i be the k-tuple containing the elements of $X_i \cup \{y\}$ in lexicographic order.

(b) We simulate $\mathcal{A}$ on input z_i using the answer "yes" for each query; if $\mathcal{A}$ queries only elements of X_i and outputs YES, then output(YES) and halt. (Note that if $\mathcal{A}$ queries only elements of X_i then the queries are answered correctly above, so the simulation correctly distinguishes between whether all k components of z_i or only $k-1$ of them are in A. Since X_i is a $(k-1)$-element subset of A, this tells us whether $y \in A$.)

Step 3: Output(NO). (By the construction, every element of A will be recognized in the previous step. Hence if no such i exists then $y \notin A$.)

We now assume that $\overline{A}$ satisfies the condition of the theorem as well. We will conclude that $A \in \mathrm{EL}_2$. First, build advice $Y_1, \ldots, Y_{p'}, T_{p'} \subseteq \overline{A}$ similar to the procedure above. We now use the sets $X_1, \ldots, X_p, S_p, Y_1, \ldots, Y_{p'}, T_{p'}$ to show $A \in \mathrm{P/poly}$ via an algorithm that satisfies the property of Lemma 3.2; therefore, $A \in \mathrm{EL}_2$. Let $\mathcal{B}_A$ ($\mathcal{B}_{\overline{A}}$) be the P/poly algorithm for A ($\overline{A}$) obtained by using advice $X_1, \ldots, X_p, S_p$ ($Y_1, \ldots, Y_{p'}, T_{p'}$).

If y is a string of length n, then we determine if $y \in A$ as follows: If $\mathcal{B}_A(y) = YES$ then output YES. Otherwise run $\mathcal{B}_{\overline{A}}(y)$. If it outputs YES then output NO. By the construction, for every $x \in A$, some X_i advises it, and for every $x \in \overline{A}$ some Y_i advises it. Hence one of the two must happen. The algorithm is more complicated than is needed to recognize A, but the point is that the set of strings that can serve as advice will be relatively simple.

Let ADV_n be the set of all strings that can serve as advice for strings of length n using this algorithm. We give a $\mathrm{P}^{A,\mathrm{NP}}$ algorithm to decide a nontrivial subset of $\cup_{n=0}^{\infty} ADV_n$. By Lemma 3.2 this shows $A \in \mathrm{EL}_2$.

A string that claims to be advice is of the form $X_1, \ldots, X_p, S_p, Y_1, Y_2, \ldots, Y_{p'}, T_{p'}$ where $S_p \cup \bigcup_{i=1}^{p} X_i \subseteq A$ and $T_{p'} \cup \bigcup_{i=1}^{p'} Y_i \subseteq \overline{A}$. These inclusions can easily be verified with queries to A. It suffices to verify that for every $y \in \Sigma^n$ either (1) y belongs to the set $\bigcup_{i=1}^{p} X_i \cup \bigcup_{i=1}^{p} Y_i \cup S_p \cup T_p$, or (2) there exists an i such that $g(X_i \cup \{y\}) = X_i$ (we can test this since $X_i \subseteq A$) (3) there exists an i such that $g(Y_i \cup \{y\}) = Y_i$ (we can test this since $Y_i \subseteq \overline{A}$). This is a co-NP predicate and hence can be answered with a query to SAT.

∎

Corollary 3.5. *If there exists $k \in \mathsf{N}$ such that $F_k^A \in \mathrm{PF}^A_{(k-1)\text{-tt}}$ via an algorithm $\mathcal{A}$ that queries only components of the input, then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$.*

Corollary 3.6. *If A is cheatable then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$.*

Proof: By Fact 2.19 if A is k-cheatable then $F_{2^k}^A \in \mathrm{PF}^A_{(2^k-1)\text{-tt}}$ via an algorithm that queries only components of the input. Therefore Corollary 3.5 applies. ∎

In the proof of Theorem 3.4 we only used the fact that Z was computable in *polynomial time* (given the answers to certain queries) in step 2b of the algorithm. In this step we had the answers to all queries that would be asked and merely ran the algorithm with this information. If originally, given the answers to certain queries, Z is in a class $\mathcal{C}$ then perhaps we can obtain $A \in \mathcal{C}/\mathrm{poly}$. This train of thought leads to a very general corollary.

We need to define computation over classes other than P with a bounded number of queries. It is obvious how to count the number of queries when the base computation is deterministic. However, when the base computation is not deterministic we may consider either the maximum number of queries per path or the total number of queries over all paths.

Definition 3.7. A language Z is in $\mathrm{NP}^A_{k\text{-T}}$ if there exists a nondeterministic polynomial time oracle Turing machine $M^{()}$ such that M^A recognizes Z and on *each* computation path makes at most k queries.

Definition 3.8. Let $\mathcal{C} \in \{\Pi_1^p, \Sigma_2^p, \Pi_2^p, \Sigma_3^p, \Pi_3^p, \ldots\} \cup \{\mathrm{QP} : Q \in \mathrm{P}\}$. (See Example 7.7 for the definition of QP). A language Z is in $\mathcal{C}^A_{\text{ctree}}[k]$ if $Z \in \mathcal{C}^A$ via an algorithm $\mathcal{A}$ and the obvious deterministic simulation of $\mathcal{A}$ makes only k queries to A. Note that we may assume that $\mathcal{A}$ makes all of its queries before executing any nondeterministic steps. This notation was first used by Wagner [74] for the case $\mathcal{C} = \mathrm{NP}$. In this case the term "ctree" meant "computation tree".

Corollary 3.9. *Let A be a set, let $k \in \mathsf{N}$, and let Z separate A^k from $\{(x_1, \ldots, x_k) : (\forall i \neq j)[x_i \neq x_j] \wedge |A \cap \{x_1, \ldots, x_k\}| = k-1\}$. If*

- *$Z \in \mathcal{C}^A_{\text{ctree}}[k-1]$ where $\mathcal{C}$ is Σ_i^p or Π_i^p for some i or any complexity class in $\cup\{\mathrm{QP} : Q \in \mathrm{P}\}$ that is closed under union*
- *or $Z \in \mathcal{C}^A_{(k-1)\text{-}T}$ where $\mathcal{C}$ is NP or PSPACE,*

via an algorithm $\mathcal{A}$ that queries only components of the input, then $A \in \mathcal{C}/\mathrm{poly}$.

Proof:

First we obtain advice exactly as in the proof of Theorem 3.4. Let the advice be $X_1, \ldots, X_p, S_p$.

Case 0: $\mathcal{C} = \mathrm{PSPACE}$. Because PSPACE is a deterministic class, we proceed exactly as in the proof of Theorem 3.4.

Case 1: $\mathcal{C} = \mathrm{NP}$. Let $\mathcal{A}(x, X_i, d)$ be the result of running $\mathcal{A}$ on x, along the path specified by d, assuming that the queries are all in $X_i \subseteq A$. If a query is made that is not in X_i then the computation outputs NO and halts. From the nature of the advice we have

$$x \in A \text{ iff } (x \in S_p \text{ or } (\exists i)(\exists d)[x \in X_i \vee \mathcal{A}(x, X_i, d) = YES]).$$

Hence $A \in$ NP/poly.

Case 2: $\mathcal{C}$ is Σ_i^p, Π_i^p or QP. Recall that we can assume that all the queries that $\mathcal{A}$ makes are made before any nondeterministic step is executed. Let $\mathcal{A}(x, X_i)$ be the result of running $\mathcal{A}(x)$ and assuming that the queries are all in $X_i \subseteq A$. If a query is made that is not in X_i then the computation creates branches that cause x to be rejected (e.g., if $\mathcal{C}$ = PARITYP then create an even number of branches that all accept.)

From the algorithm we obtain

$$x \in A \text{ iff } (x \in S_p \text{ or } \bigvee_{i=1}^{p} [\mathcal{A}(x, X_i) = YES].)$$

Since $\mathcal{C}$ is closed under union, $A \in \mathcal{C}$/poly. ∎

3.2. Applications to Selective Sets

Inspired by the concept of a semi-recursive set from recursion theory [44], Selman [66] defined p-selective sets to distinguish several types of polynomial reducibilities. Hemaspaandra et al. generalized this concept to NPMV-selective ([39, 42]) and NPSV-selective ([40]).

Convention: Let M be a nondeterministic polynomial time Turing machine. We will think of M as computing a multivalued partial function by having its branches write strings. By convention all paths halt, though some will output $\perp$, which stands for not contributing to the answer.

Definition 3.10. [19] A partial multivalued function $f : \Sigma^* \to 2^{\Sigma^*}$ is in NPMV (nondeterministic polynomial time multi-valued) if there exists a nondeterministic polynomial time Turing machine M such that $y \in f(x)$ iff one of the branches in the computation of $M(x)$ outputs y. Note that if $f(x) = \emptyset$ then all branches will output $\perp$. A partial function $f : \Sigma^* \to \Sigma^*$ is in NPSV (nondeterministic polynomial time single valued) if it is in NPMV.

Definition 3.11. Let A be a language. An *A-selector* is a partial multivalued function f such that $A \cap \{x, y\} \neq \emptyset \Rightarrow \emptyset \subset f(x, y) \subseteq A \cap \{x, y\}$. An A-selector is *single-valued* if $A \cap \{x, y\} \neq \emptyset \Rightarrow |f(x, y)| = 1$.

i. A is *p-selective* if there is a single-valued A-selector $f \in$ PF.

ii. A is *NPMV-selective* if there is an A-selector $f \in$ NPMV.

iii. A is *NPSV-selective* if there is a single-valued A-selector $f \in \mathrm{NPSV}$.

Corollary 3.12 ([48]). *If A is (weakly) p-selective then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$.*

Proof: We show this result for p-selective sets. The proof for weakly p-selective sets (defined in [48]) is similar.

We show that if A is p-selective then both A and $\overline{A}$ satisfy the hypothesis of Theorem 3.4. Let f be a polynomial-time computable A-selector. Then $(x, y) \in \overline{A} \times \overline{A}$ iff $f(x, y) \in \overline{A}$, so $\overline{A}$ satisfies the hypothesis of Theorem 3.4. Since the class of p-selective sets is closed under complementation, A satisfies that hypothesis as well. ∎

It was already known that if A is weakly p-selective then A is in P/poly [48] (Ko constructs a small set of witnesses for membership in A) and in L_2 (i.e., $\Sigma_2^{p,A} \subseteq \Sigma_2^p$, see [49]).

We now apply Corollary 3.9 to NPMV-selective sets and NPSV-selective sets. The following result was originally proven by Hemaspaandra et al. [39, 41, 42]).

Corollary 3.13. *If A is NPMV-selective then $A \in \mathrm{NP/poly}$. If A is NPSV-selective then $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly} \cap \mathrm{EL}_3$.*

Proof: We show that if A is NPMV-selective via f then A satisfies the condition of Corollary 3.9 with $\mathcal{C} = \mathrm{NP}$ and $k = 2$. We define Z by the following $\mathrm{NP}^{A[1]}$ algorithm: On input (x, y) run the NP algorithm for f. On each branch do the following. If the output of f is $\perp$ then we output NO. If the output of f is x then make the query "$y \in A$?" Output the answer to this query. Note that $A^2 \subseteq Z$ and $\{(x, y) : x \neq y \wedge |A \cap \{x, y\}| = 1\} \subseteq \overline{Z}$. Also note that each path asked at most one question, though there may have been two questions asked overall.

If A is NPSV-selective via f then A is NPMV-selective hence $A \in \mathrm{NP/poly}$. We show that A satisfies the condition of Corollary 3.9 with $\mathcal{C} = \mathrm{co\text{-}NP}$ and $k = 2$. We define Z by the following $\mathrm{co\text{-}NP}^{A[1]}$ algorithm. On input (x, y) run the NP algorithm for f. On each branch do the following. If the output of f is $\perp$ then output YES. If the output of f is x then make the query "$y \in A$?" Output the answer to this query. Note that $A^2 \subseteq Z$ and $\{(x, y) : x \neq y \wedge |A \cap \{x, y\}| = 1\} \subseteq \overline{Z}$. Also note that since f is in NPSV the total number of different queries asked overall is at most 1.

By [40] any set in $\mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$ is in EL_3. ∎

3.3. Self-Reduction and NP-hardness

In this section we show that if a set is self-reducible and cheatable then it is in P. [2] We use this to show that, under a suitable hypothesis, certain sets A are not cheatable. We then prove a lemma that extends this to any set B such that $A \leq_{\mathrm{T}}^{p} B$.

[2]This result was first stated (without proof) in [10], crediting the current authors. We subsequently learned that [36] obtained the result independently.

Many natural sets, *e.g.*, all NP-complete sets in [31], are self-reducible, as defined by Schnorr in [62]. We use an alternative definition which is more general and is implicit in the literature. It was first introduced by [56] and a variation of it is in [48].

Definition 3.14. Let p be a function and $\prec$ be an ordering on Σ^*. The ordering $\prec$ has *p-bounded chains* if whenever $x_m \prec x_{m-1} \prec \cdots \prec x_1$, we have $m \leq p(|x_1|)$ and $(\forall i)[|x_i| \leq p(|x_1|)]$. The ordering $\prec$ has *polynomially-bounded chains* if there exists a polynomial p such that $\prec$ has p-bounded chains.

Definition 3.15. A set A is *polynomial Turing self-reducible* (henceforth self-reducible) if there exists a polynomial-time computable partial order $\prec$ such that

i. $\prec$ has polynomially bounded chains; and

ii. there exists a polynomial time bounded oracle Turing machine $M^{()}$ such that

   (a) all strings queried by $M^{()}$ precede the input string in the ordering $\prec$, and

   (b) the language accepted by M^A is A.

The definition of *polynomial truth-table self-reducible* (henceforth tt-self-reducible) is similar, just replace the polynomial Turing reduction with a polynomial tt-reduction.

Note that the set of elements of A that are minimal with respect to $\prec$ is in P.

We state a theorem that appeared in [14]. Theorems 3.20 and 3.31, and Lemma 3.29 are generalizations of it.

Proposition 3.16. *A is tt-self-reducible and cheatable iff A is in P.*

In order to improve Proposition 3.16 from tt-self-reducible to self-reducible we need the following lemma.

Definition 3.17. A Σ^**-labeled tree* is a tree where every node is mapped to an element of Σ^*. If the tree is binary then every node can be interpreted as an element of $\{0,1\}^*$ and a labeled tree is a mapping from $\{0,1\}^*$ to Σ^*.

Definition 3.18. Let $f \leq_{\mathrm{T}}^{p} B$ via an oracle Turing machine $M^{()}$ that runs in time bounded by $p(n)$ for all oracles. Let $x \in \Sigma^*$. The *oracle query tree for $M^{()}(x)$* is the labeled tree such that the node $b_1 b_2 \cdots b_{m-1}$ ($b_i \in \{0,1\}$) is mapped to the m^{th} question that would be asked if the first $m-1$ questions are answered $b_1, b_2, \ldots, b_{m-1}$. Note that the depth of the tree is at most $p(|x|)$.

Lemma 3.19. *If B is k-cheatable and $f \leq_{\mathrm{T}}^{p} B$ then $f \leq_{(2^k-1)\text{-tt}}^{p} B$ using queries that belong to the oracle query tree in the $f \leq_{\mathrm{T}}^{p} B$ Turing reduction.*

Proof: Let $f \leq_{\mathrm{T}}^{p} B$ via an oracle Turing machine $M^{()}$ that runs in time bounded by $p(n)$ for all oracles. Since B is k-cheatable we have, by Fact 2.19.*iii*, that $F_{2^{k+2}}^{B}$ is 2^k-enumerable. We use this later.

Given x, we show how to compute $f(x)$ with $2^k - 1$ parallel queries to B where those queries are on the oracle query tree. Assume $|x| = n$. We generate the oracle query tree for $M^{()}(x)$ and prune it so that it remains small. Let T_i be the pruned tree through level i (note that T_0 is the one-node tree that labels that one node with the first query). Let $a(i)$ be the number of nodes in T_i and $b(i)$ be the number of leaves in T_i. We describe the pruning process and derive the bounds $a(i) \leq 2^{k+1}$ and $b(i) \leq 2^k$. Note that $a(0) = b(0) = 1$ trivially.

Inductively assume that we have constructed T_i with $a(i) \leq 2^{k+1}$ and $b(i) \leq 2^k$. Find the queries asked on both a YES and NO answer to the leaf queries in T_i. The total number of queries is now $\leq a(i) + 2b(i) \leq 2^{k+2}$. Assume, without loss of generality, that there are exactly 2^{k+2} queries (can repeat queries to pad). Let the queries be $x_1, \ldots, x_{2^{k+2}}$. Since $F_{2^{k+2}}^{B}$ is 2^k-enumerable we can generate 2^k possibilities for $F_{2^{k+2}}^{B}(x_1, \ldots, x_{2^{k+2}})$. We prune from the tree all leaves that do not correspond to any of these possibilities. There are now $\leq 2^k$ leaves total, so $b(i+1) \leq 2^k$. This bound on the number of leaves implies that the total number of nodes is bounded by 2^{k+1}, hence $a(i+1) \leq 2^{k+1}$.

The tree $T_{p(n)}$ can be found in polynomial time and has at most 2^{k+1} queries. Let $x_1, \ldots, x_{2^{k+1}}$ be those queries. Note that all of the x_i are on the oracle query tree of the $f \leq_{\mathrm{T}}^{p} B$ Turing reduction. By Fact 2.19.*i* $F_{2^{k+1}}^{B}(x_1, \ldots, x_{2^{k+1}})$ can be determined by querying $2^k - 1$ of the elements of $\{x_1, \ldots, x_{2^{k+1}}\}$. Once this is done $f(x)$ can easily be computed. So $f \leq_{(2^k-1)\text{-tt}}^{p} B$ and all the queries came from the oracle query tree of the $f \leq_{\mathrm{T}}^{p} B$ Turing reduction. ∎

Theorem 3.20. *A is self-reducible and cheatable iff A is in P.*

Proof: Assume A is self-reducible and cheatable. By Proposition 3.16 we need only prove that A is tt-self-reducible. Let $M^{()}$ be such that A is self-reducible via M^{A}. Note that for all x, for all queries y made in the oracle query tree of $M^{()}(x)$, $y \prec x$. Let k be such that A is k-cheatable. Apply Lemma 3.19 to the set A and the characteristic function χ_A to obtain that $\chi_A \in \mathrm{PF}_{(2^k-1)\text{-tt}}^{A}$ using queries that belong to the oracle query tree of $M^{()}(x)$. Since all such queries are $\prec x$, A is tt-self-reducible.

The converse is trivial. ∎

We apply our Theorem 3.20 to particular sets.

Corollary 3.21.

i. If $\mathrm{P} \neq \mathrm{NP}$ *then* SAT *is not cheatable.*

ii. If $\mathrm{P} \neq \Sigma_i^p$ *then* QBF_i *is not cheatable.*

iii. If $\mathrm{P} \neq \mathrm{PSPACE}$ *then* QBF *is not cheatable.*

(See Definition 2.10 for a definition of QBF_i *and* QBF.*)*

To extend Corollary 3.21 to NP-hard sets (Σ_i^p-hard sets, PSPACE-hard sets) we need the following lemma.

Lemma 3.22. *If B is k-cheatable and $A \leq_{\mathrm{T}}^p B$ then A is k-cheatable.*

Proof: Since $A \leq_{\mathrm{T}}^p B$, $F_{2^k}^A \leq_{\mathrm{T}}^p B$. By Lemma 3.19 $F_{2^k}^A \in \mathrm{PF}_{(2^k-1)\text{-tt}}^B$. Since B is k-cheatable, the $2^k - 1 \leq 2^k$ parallel queries can be answered by using k queries to some set X. Hence $F_{2^k}^A$ can be computed with k queries to that set X, so A is k-cheatable. ▮

Corollary 3.23.

i. If $\mathrm{P} \neq \mathrm{NP}$ *then (Turing-)* NP*-hard sets are not cheatable.*

ii. If $\mathrm{P} \neq \Sigma_i^p$ *then (Turing-)* Σ_i^p*-hard sets are not cheatable.*

iii. If $\mathrm{P} \neq \mathrm{PSPACE}$ *then (Turing-)* PSPACE*-hard sets are not cheatable.*

Corollaries 3.21 and 3.23 have been superseded by [17, 57] who have shown that if $\mathrm{P} \neq \mathrm{NP}$ then SAT is p-superterse.

3.4. Non-constant number of queries

We extend the results of the last section to a variation on k-cheatability where k is a function instead of a constant. In particular we show that if A is self-reducible and $k(m)$-cheatable (where $2^{k(m)}$ is bounded by a polynomial) then $A \in \mathrm{P}$. This will be used to show that, under a suitable hypothesis, certains sets are not $k(m)$-cheatable. These results will be used in Section 7.4 to show that, under a suitable hypothesis, certain functions are not n^ϵ-enumerable.

In order to generalize k-cheatability we need a generalization of F_k^A. The next definition provides such.

Definition 3.24. Let A be a set and let q be a function. The function F_q^A is defined only on the domain $\cup_{m=0}^{\infty}(\Sigma^{\leq m})^{q(m)}$ as follows

$$(\forall m)(\forall x_1, \ldots, x_{q(m)} \in \Sigma^{\leq m})[F_q^A(x_1, \ldots, x_{q(m)}) = A(x_1) \cdots A(x_{q(m)})].$$

Notation 3.25. When we write $F_q^A(x_1, \ldots, x_{q(m)})$ we assume $(\forall i)[|x_i| \leq m]$.

In this section we avoid using the bounded query classes notation and the enumerability notation since the parameter of interest will be m, which is *not* the length of the input.

Definition 3.26. Let A be a set. Let $k(m)$ be a function such that $q(m) = 2^{k(m)}$ is polynomial bounded. A is *$k(m)$-cheatable* if there exists a set X and a polynomial time oracle Turing machine $M^{()}$ such that M^X computes $F_q^A(x_1, \ldots, x_{q(m)})$ with at most $k(m)$ queries to X.

Important Convention: Throughout this section we will assume that $k(m)$ is such that $2^{k(m)}$ is bounded by a polynomial, and we will let q denote the function $q(m) = 2^{k(m)}$.

The following two facts closely parallel Fact 2.16 and Fact 2.19.

Fact 3.27. *Let A be a set. The following are equivalent:*

i. A is $k(m)$-cheatable.

ii. There exists a function $e \in$ PF, $e : \cup_{m=0}^{\infty}(\Sigma^{\leq m})^{q(m)} \rightarrow \{0, 1, \%\}^*$ such that $(\forall m)(\forall x_1, \ldots, x_{q(m)} \in \Sigma^{\leq m})$

(a) $e(x_1, \ldots, x_{q(m)})$ is a list of at most $q(m)$ elements of $\{0, 1\}^*$, and

(b) one of the elements on the list is $F_q^A(x_1, \ldots, x_{q(m)})$.

Proof: Similar to the proof of Fact 2.16. ∎

Fact 3.28. *Let A be a set.*

i. If A is $k(m)$-cheatable then, for any polynomial $r(m)$, $F_r^A(x_1, \ldots, x_{r(m)})$ can be computed in polynomial time by querying a subset of at most $q(m) - 1$ elements of $\{x_1, \ldots, x_{r(m)}\}$.

ii. If A is $k(m)$-cheatable then there exists an X such that for all polynomials r, $F_r^A(x_1, \ldots, x_{r(m)})$ can be computed with $k(m)$ queries to X.

iii. A is $k(m)$-cheatable iff the following holds. For every polynomial r there exists a function $e \in$ PF, $e : \cup_{m=0}^{\infty}(\Sigma^{\leq m})^{r(m)} \rightarrow \{0, 1, \%\}^*$, such that $e(x_1, \ldots, x_{r(m)})$ is a list of at most $q(m)$ elements of $\{0, 1\}^*$, one of which is $F_r^A(x_1, \ldots, x_{r(m)})$.

Proof sketch:

We show that (i) holds. Assume that $r(m) > q(m)-1$ otherwise this is trivial. We compute $F_r^A(x_1, \ldots, x_{r(m)})$ as follows. Find $q(m)$ possibilities for $F_q^A(x_1, \ldots, x_{q(m)})$ by Fact 3.27. By a combinatorial lemma [14, 30] we can find a subset of $\{x_1, \ldots, x_{q(m)}\}$ of size $q(m)-1$ such that the membership of those elements in A completely determines which possibility is actually $F_{q(m)}^A(x_1, \ldots, x_{q(m)})$. Say that the query eliminated is x_i. Iterate this process with $F_q^A(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{q(m)}, x_{q(m)+1})$. After $r(m) - q(m)$ steps we are left with just $q(m) - 1$ queries.

The proofs of (ii) and (iii) are analogous to the proofs of (ii) and (iii) of Fact 2.19.
∎

The following two lemmas closely parallel Proposition 3.16 and Lemma 3.19.

Lemma 3.29. *A is tt-self-reducible and $k(m)$-cheatable iff $A \in$ P.*

Proof: Let $\prec$ be the ordering on A used in the tt-self-reduction and let p_1 be the polynomial such that $\prec$ has p_1-bounded chains. Let p_2 be a polynomial that bounds the running time of the tt-reduction. Let p be a polynomial such that $p(n) \geq \max\{p_1(n), p_2(n)\}$. Let r be a polynomial such that, for all m, $q(p(m)) \leq r(p(m))$ and $r(m)p(m) \leq r(p(m))$. Let r' be the polynomial such that, for all m, $r'(m) = r(p(m))$. By Fact 3.28 we have the following facts which we will use later. (1) If $(\forall i)[|y_i| \leq p(m)]$ then $F_r^A(y_1, \ldots, y_{r(p(m))})$ can be computed by querying a subset of $\{y_1, \ldots, y_{r(p(m))}\}$ of size $\leq q(p(m)) - 1$; and (2) if $(\forall i)[|z_i| \leq p(m)]$ then $F_{r'}^A(z_1, \ldots, z_{r'(p(m))})$ can be computed by querying a subset of $\{z_1, \ldots, z_{r'(p(m))}\}$ of size $\leq q(p(m)) - 1$.

We show that F_r^A is computable in polynomial time. Let the input be $(x_1, \ldots, x_{r(m)})$ (assume $(\forall i)[|x_i| \leq m]$). We intend to use the tt-self-reduction algorithm on each individual x_i, and then on the queries that come out from that, etc. Note that by the definition of self-reduction, for *any* query y encountered at *any* step of this process, $|y| \leq p(m)$.

Initially we tt-reduce every x_i to $\leq p(m)$ queries which are all $\prec x_i$ and of length $\leq p(m)$. By doing this for all i we have at most $r(m)p(m) \leq r(p(m))$ strings. Assume, without loss of generality, that we have exactly $r(p(m))$ strings (use 0's for padding). Call them $y_1, \ldots, y_{r(p(m))}$. Note that $\langle y_1, \ldots, y_{r(p(m))}\rangle$ is in the domain of F_r^A. Hence from $F_r^A(y_1, \ldots, y_{r(p(m))})$ we can easily find $F_r^A(x_1, \ldots, x_{r(m)})$.

We need to find $F_r^A(y_1, \ldots, y_{r(p(m))})$. We will reduce this to the problem of finding $F_r^A(z_1, \ldots, z_{r(p(m))})$ where

$$(\forall i)[|z_i| \leq p(m) \text{ and } (\exists j)[z_i \prec y_j]].$$

This procedure will be iterative so that it can be applied again to further reduce the problem. By the definition of p this procedure will be iterated at most $p(m)$ times.

The reduction proceeds as follows. We tt-reduce every y_i to $\leq p(m)$ queries which are all $\prec y_i$ and of length $\leq p(m)$ (this is because we initially noted that *any* query encountered will be of length $\leq p(m)$). By doing this for all i we have at most $r(p(m))p(m) \leq r(p(p(m))) = r'(p(m))$ strings. Assume, without loss of generality, that we have exactly $r'(p(m))$ strings (use 0's for padding). Call them $z_1, \ldots, z_{r'(p(m))}$.

Note that $(z_1, \ldots, z_{r'(p(m)})$ is in the domain of $F_{r'}^A$. Hence from $F_{r'}^A(z_1, \ldots, z_{r'(p(m))})$ we can easily find $F_r^A(y_1, \ldots, y_{r(p(m))})$.

By Fact 3.28.i applied to r' the value of $F_{r'}^A(z_1, \ldots, z_{r'(p(m))})$ can be determined by querying a subset of $q(p(m)) - 1 \leq r(p(m))$ of the elements $\{z_1, \ldots z_{r'(p(m))}\}$. Renumber so that those queries are $z_1, \ldots, z_{r(p(m))}$.

We have reduced $F_r^A(y_1, \ldots, y_{r(p(m))})$ to $F_r^A(z_1, \ldots, z_{r(p(m))})$ where every element of $\{z_1, \ldots, z_{r(p(m))}\}$ is $\prec$ some element of $\{y_1, \ldots, y_{r(p(m))}\}$. Repeat this procedure until all the elements are $\prec$-minimal. By the nature of p, (1) the number of iterations needed before hitting $\prec$-minimal elements is bounded by $p(m)$, and (2) the length of the intermediary queries is bounded by $p(m)$. Since testing membership in A for $\prec$-minimal elements is in polynomial time, we can find $F_r^A(y_1, \ldots, y_{r(p(m))})$ in polynomial time. With this information we can determine $F_r^A(x_1, \ldots, x_{r(m)})$ in polynomial time.

The converse is trivial. ∎

In order to improve Lemma 3.29 from tt-self-reducible to self-reducible we need the following lemma.

Lemma 3.30. *If B is $k(m)$-cheatable and $f \le_{\mathrm{T}}^p B$ then $f \le_{\mathrm{tt}}^p B$ using queries that belong to the oracle query tree in the $f \le_{\mathrm{T}}^p B$ Turing reduction.*

Proof: Let $f \le_{\mathrm{T}}^p B$ via an oracle Turing machine $M^{()}$ that runs in time bounded by $p(n)$ for all oracles. Fix a polynomial r such that $4q(p(n)) \le r(p(n))$. By Fact 3.27 there exists a function $e \in$ PF that, on input $\langle x_1, \ldots, x_{r(p(n))}\rangle$ (where $(\forall i)[|x_i| \le p(n)]$), lists at most $q(p(n)) \le r(p(n))$ possibilities for $F_r^A(x_1, \ldots, x_{r(p(n))})$, one of which is correct. We will use this later.

On input x, $|x| = n$, we generate the oracle query tree for $M^{()}(x)$ and prune it so that it remains small. Let T_i be the pruned tree through level i (note that T_0 is the one-node tree that has the first query on the one node). Let $a(i)$ be the number of nodes in T_i and $b(i)$ be the number of leaves in T_i. We describe the pruning process and derive the bounds $a(i) \le 2q(p(n))$ and $b(i) \le q(p(n))$. Note that $a(0) = b(0) = 1$ trivially.

Inductively assume that we have constructed T_i with $a(i) \le 2q(p(n))$ and $b(i) \le q(p(n))$. Find the queries asked on both a YES and NO answer to the leaf queries in T_i. The total number of queries is now $\le a(i) + 2b(i) \le 4q(p(n)) \le r(p(n))$. Assume, without loss of generality, that there are exactly $r(p(n))$ queries (you can repeat queries to pad). Let the queries be $x_1, \ldots, x_{r(p(n))}$. Note that $(\forall i)[|x_i| \le p(n)]$. Hence the value of $e(\langle x_1, \ldots, x_{r(p(n))}\rangle)$ is defined and is a list of $\le q(p(n))$ elements, one of which is $F_r^B(x_1, \ldots, x_{r(p(n))})$. Also note that $e(\langle x_1, \ldots, x_{r(p(n))}\rangle)$ can be found in time polynomial in $|\langle x_1, \ldots, x_{r(p(n)}\rangle|$.

We prune from the tree all leaves that do not correspond to any of these possibilities. There are now $\le q(p(n))$ leaves total, so $b(i+1) \le q(p(n))$. This bound on the number of leaves implies that the total number of nodes is bounded by $2q(p(n))$, hence $a(i+1) \le 2q(p(n))$.

The tree $T_{p(n)}$ can be found in polynomial time and has at most $2q(p(n))$ queries. These can be asked in parallel, so $f \le_{\mathrm{tt}}^p B$ and all the queries came from the oracle query tree of the $f \le_{\mathrm{T}}^p B$ Turing reduction. ∎

Theorem 3.31. *A is self-reducible and $k(m)$-cheatable iff A is in P.*

Proof: The proof of this theorem is similar to that of Theorem 3.20 except that we use Lemma 3.29 and Lemma 3.30 instead of Proposition 3.16 and Lemma 3.19. ∎

Corollary 3.32.

i. If P $\neq$ NP *then* SAT *is not $k(m)$-cheatable.*

ii. If $\mathrm{P} \neq \Sigma_i^p$ *then* QBF_i *is not* $k(m)$*-cheatable.*

iii. If $\mathrm{P} \neq \mathrm{PSPACE}$ *then* QBF *is not* $k(m)$*-cheatable.*

(See Definition 2.10 for a definition of QBF_i *and* QBF*.)*

To extend Corollary 3.32 to NP-hard sets (Σ_i^p-hard sets, PSPACE-hard sets) we need the following lemma.

Lemma 3.33. *If* B *is* $k(m)$*-cheatable and* $A \leq_{\mathrm{T}}^p B$ *then there exists a function* k' *with* $2^{k'(m)}$ *polynomial bounded such that* A *is* $k'(m)$*-cheatable.*

Proof: Let $k'(m)$ and $q'(m) = 2^{k'(m)}$ be functions that we will specify later. We would like to enumerate $\leq q'(m)$ possibilities for $F_{q'}^A(x_1, \ldots, x_{q'(m)})$, which will show that A is $k'(m)$-cheatable (by Fact 3.27).

Clearly $F_{q'}^A \leq_{\mathrm{tt}}^p A \leq_{\mathrm{T}}^p B$. By Lemma 3.30 $F_{q'}^A \leq_{\mathrm{tt}}^p B$ by a reduction that asks queries on the $F_{q'}^A \leq_{\mathrm{T}}^p B$ query tree. Let $t(m)$ bound the time the reduction of $A \leq_{\mathrm{T}}^p B$ takes on elements of length m. Let $p(q'(m))$ bound the number of queries made to B on input $x_1, \ldots, x_{q'(m)}$ (where $(\forall i)[|x_i| \leq m]$) in the $F_{q'}^A \leq_{\mathrm{tt}}^p B$ reduction.

Note that in the reduction of $F_{q'}^A$ to A we query elements of length $\leq m$ (which is not the length of the input). Hence in the reduction of $F_{q'}^A$ to B we ask (in parallel) $\leq p(q'(m))$ queries of length $\leq t(m)$. Let $r(m)$ be a polynomial such that $p(q'(m)) \leq r(t(m))$.

We now describe how to enumerate $\leq q'(m)$ possibilities for $F_{q'}^A(x_1, \ldots, x_{q'(m)})$, given $x_1, \ldots, x_{q'(m)}$ (where $(\forall i)[|x_i| \leq m]$). First prepare the $\leq p(q'(m))$ queries to B of length $\leq t(m)$ that are needed. Assume (by padding) that the queries to B are $y_1, \ldots, y_{r(t(m))}$. Hence every possibility for $F_r^B(y_1, \ldots, y_{r(t(m))})$ leads to a possibility for $F_{q'}^A(x_1, \ldots, x_{q'(m)})$.

Since B is $k(m)$-cheatable, by Fact 3.28 we can enumerate $q(t(m))$ possibilities for $F_r^B(y_1, \ldots, y_{r(t(m))})$. Hence we can enumerate $q(t(m))$ possibilities for $F_{q'}^A(x_1, \ldots, x_{q'(m)})$.

Choose q' such that $q(t(m)) \leq q'(m)$. ∎

Corollary 3.34.

i. If $\mathrm{P} \neq \mathrm{NP}$ *then (Turing-)* NP*-hard sets are not* $k(m)$*-cheatable.*

ii. If $\mathrm{P} \neq \Sigma_i^p$ *then (Turing-)* Σ_i^p*-hard sets are not* $k(m)$*-cheatable.*

iii. If $\mathrm{P} \neq \mathrm{PSPACE}$ *then (Turing-)* PSPACE*-hard sets are not* $k(m)$*-cheatable.*

4. Non P-Superterse Sets

We pursue the question "Which sets A can be non-p-superterse?" In Section 4.1 we show that if A is not p-superterse then $A \in \mathrm{P/poly}$. We then extend the techniques to obtain a generalization that involves a nonconstant number of queries. This generalization is applied in Sections 5,6, and 7 to obtain results about bounded query classes, circuits, and enumeration.

4.1. Circuits

We prove that if A is not p-superterse then A is in P/poly. Although the result resembles Theorem 3.4 the proof is substantially different.

We use the following notation in Theorems 4.2 and 4.4.

Notation 4.1.

- If t is a k-tuple of bits $(b_1, \ldots, b_k)$ then $t[i]$ denotes b_i and $t[i:j]$ denotes $(b_i, \ldots, b_j)$.
- If x is a string, y is a j-tuple of strings $(y_1, \ldots, y_j)$, and z is a k-tuple of strings $(z_1, \ldots, z_k)$ then (y, z) denotes $(y_1, \ldots, y_j, z_1, \ldots, z_k)$, (x, y) denotes $(x, y_1, \ldots, y_j)$ and (y, x) denotes $(y_1, \ldots, y_j, x)$.

Theorem 4.2. *If A is not p-superterse then* $A \in \mathrm{P/poly}$.

Proof: Let A be a non-p-superterse set. By Fact 2.20 there exist $k \in \mathsf{N}$ and $w \in \mathrm{PF}$, $w : (\Sigma^*)^k \to \{0,1\}^k$, such that for all tuples t in $(\Sigma^*)^k$, $w(t) \neq F_k^A(t)$. Hence we have the weaker statement, denoted by $S(k)$, that there exists $w \in \mathrm{PF/poly}$, $w : \bigcup_{n=1}^{\infty}(\Sigma^n)^k \to \{0,1\}^k$, such that for all $t \in (\Sigma^n)^k$, $w(t) \neq F_k^A(t)$. We show that for all $m \geq 2$, $S(m)$ implies $S(m-1)$. Since we have $S(k)$, this implies $S(1)$, which yields $A \in \mathrm{P/poly}$. Since we only need the implication $S(m) \Rightarrow S(m-1)$ for $m \leq k$ we can treat m as a constant.

Assume $S(m)$ is true via w. We show $S(m-1)$ by constructing $w' \in \mathrm{P/poly}$, $w' : \bigcup_{n=1}^{\infty}(\Sigma^n)^{m-1} \to \{0,1\}^{m-1}$, such that $w'(t) \neq F_{m-1}^A(t)$. We show how to construct the advice for computing $w'(t)$ where $t \in (\Sigma^n)^{m-1}$. The construction is an iterative process that, at each iteration, finds advice for computing w' for a fraction of its inputs (this uses the function w). Two things may happen in the construction. If the construction finds advice for every tuple in $(\Sigma^n)^{m-1}$ then we have advice for w' on $(\Sigma^n)^{m-1}$. If the construction is unable to find advice then the very reason that it cannot find advice yields a probabilistic algorithm for $A \cap \Sigma^n$. More advice removes the probability. (We will use Theorem 11.6 from the appendix, which is a variation of Schöning's proof that $\mathbf{BP} \cdot \mathcal{C} \subseteq \mathcal{C}/\mathrm{poly}$.)

Definition Let $n, m \in \mathsf{N}$. Assume $S(m)$ is true via w. Let $z \in \Sigma^n$. ADVISEES(z) is the set of all $(m-1)$-tuples $t \in (\Sigma^n)^{m-1}$ such that $w(z,t)[1] = A(z)$. (Since we know that $w(z,t) \neq F_m^A(z,t)$ we have $w(z,t)[2:m] \neq F_{m-1}^A(t)$.) We say that a string z is *advice for a set* T of $(m-1)$-tuples if

$$|\text{ADVISEES}(z) \cap T| > \frac{1}{4}|T|.$$

CONSTRUCTION OF ADVICE FOR Σ^n

BEGIN

$T_n := (\Sigma^n)^{m-1}$
$Z_n := \emptyset$
While there exists a string z in Σ^n that is advice for T_n
  choose such a z
  $T_n := T_n - \text{ADVISEES}(z)$
  $Z_n := Z_n \cup \{z\}$
END

END OF CONSTRUCTION

Note that after the ith iteration $|T_n| \leq (\frac{3}{4})^i * |(\Sigma^n)^{m-1}|$. Hence there are at most $O(\log |(\Sigma^n)^{m-1}|) = O(n)$ iterations. Since the number of elements in Z_n is the number of iterations, $|Z_n| = O(n)$.

Let I be the set of all n such that $T_n \neq \emptyset$. We show the following.

i. There exists $w'_1 \in$ P/poly, $w'_1 : \bigcup_{n\notin I}(\Sigma^n)^{m-1} \rightarrow \{0,1\}^{m-1}$, such that $w'_1(t) \neq F^A_{m-1}(t)$.

ii. There exists $w'_2 \in$ P/poly, $w'_2 : \bigcup_{n\in I}(\Sigma^n)^{m-1} \rightarrow \{0,1\}^{m-1}$, such that $w'_2(t) \neq F^A_{m-1}(t)$.

These two easily yield the desired w'.

i. If $n \notin I$ then $T_n = \emptyset$. Let the advice be the union of $\{(z, A(z)) \mid z \in Z_n\}$ and the advice needed to compute w on $(\Sigma^n)^m$ (we use the induction hypothesis here). For $t \in \bigcup_{n\notin I}(\Sigma^n)^{m-1}$ we define $w'_1(t)$ by

$$\begin{aligned} z = &\quad \min\{y : y \in Z_n \text{ and } t \in \text{ADVISEES}(y)\} \\ w'_1(t) = &\quad w(z,t)[2:m] \end{aligned}$$

(The minimum in the definition of z is with respect to lexicographic ordering.) Thus,

$$(\forall n \notin I)(\forall t \in (\Sigma^n)^{m-1})[w'_1(t) \neq F^A_{m-1}(t)].$$

w'_1 can be computed efficiently using the advice.

ii. If $n \in I$ then $T_n \neq \emptyset$ and for all $z \in \Sigma^n$ at least 3/4 of the elements $t \in T_n$ satisfy $w(z,t)[1] \neq A(z)$. Let $B_n = \{\langle x,t\rangle \mid w(x,t)[1] = 0\}$. Note that for all $n \in I$, for all $x \in \Sigma^n$

$$\begin{aligned} x \in A &\Rightarrow \text{Prob}(\langle x,t\rangle \in B_n \mid t \in T_n) \geq \tfrac{3}{4} \\ x \notin A &\Rightarrow \text{Prob}(\langle x,t\rangle \notin B_n \mid t \in T_n) \geq \tfrac{3}{4} \end{aligned}$$

Since $w \in$ P/poly we have $\bigcup_{n\in I} B_n \in$ P/poly. By Theorem 11.6 (with $Y = \cup_{n\in I} T_n$ and $B = \cup_{n\in I} B_n$) $A \cap \bigcup_{n\in I} \Sigma^n \in$ (P/poly)/poly = P/poly. Hence the desired w'_2 can easily be constructed. ∎

Because all sets in P/poly belong to EL_3, it follows that all non-p-superterse sets belong to EL_3. Hoene and Nickelsen [43, Theorem 6, Corollary 7] have shown this is optimal.

Corollary 4.3.

i. *If* $\Sigma_2^p \neq \Pi_2^p$ *then all (Turing-) NP-hard sets are p-superterse.*

ii. *If* $\Sigma_2^p \neq \mathrm{PSPACE}$ *then all (Turing-)* PSPACE*-hard sets are p-superterse.*

iii. *If* $\mathrm{P} \neq \mathrm{PSPACE}$ *then all* $\leq_{\mathrm{tt}}^p$*-hard sets for* PSPACE *are p-superterse.*

iv. *Every* $\leq_{\mathrm{tt}}^p$*-hard set for* EXPTIME *is p-superterse.*

Proof:

i. If $\mathrm{NP} \subseteq \mathrm{P/poly}$ then $\Sigma_2^p = \Pi_2^p$ [46].

ii. If $\mathrm{PSPACE} \subseteq \mathrm{P/poly}$ then $\Sigma_2^p = \mathrm{PSPACE}$ [46].

iii. Assume that A is $\leq_{\mathrm{tt}}^p$-hard for PSPACE and A is not p-superterse. Then $\Sigma_2^p =$ PSPACE by (ii). Hence A is also $\leq_{\mathrm{tt}}^p$-hard for Δ_2^p. In [11] we showed that if a non-p-superterse set is $\leq_{\mathrm{tt}}^p$-hard for Δ_2^p then $\mathrm{P} = \mathrm{NP}$. By [68] this implies $\mathrm{P} = \Sigma_2^p$, hence we have $\mathrm{P} = \Sigma_2^p = \mathrm{PSPACE}$.

iv. Assume that A is $\leq_{\mathrm{tt}}^p$-hard for EXPTIME and that A is not p-superterse. Since every set in P/poly is tt-reducible to a tally set there exists a tally set T with $A \leq_{\mathrm{tt}}^p T$, so T is $\leq_{\mathrm{tt}}^p$-hard for EXPTIME. Book and Ko [20, Theorem 6.1] proved that no tally set can be $\leq_{\mathrm{tt}}^p$-hard for EXPTIME.

∎

Corollary 4.3 has been superseded by [17, 57], who have shown that if $\mathrm{P} \neq \mathrm{NP}$ then SAT is p-superterse.

4.2. Non-constant Number of Queries

We extend Theorem 4.2 to a nonconstant number of queries. In the proof of Theorem 4.2 we used Fact 2.20: if A is not p-superterse then there is $k \in \mathsf{N}$ and $w \in \mathrm{PF}$, $w : (\Sigma^*)^k \to \{0,1\}^k$, such that for all $x_1, \ldots, x_k$, $F_k^A(x_1, \ldots, x_k) \neq w(x_1, \ldots, x_k)$. Theorem 4.4 can be seen as an extension of Theorem 4.2 by replacing the function w with a relation W.[3]

We will use the results of this section in Sections 5, 6, and 7.

Theorem 4.4. *Let* $k(n)$ *be polynomial-bounded and let* A *be a language. Assume there exists a set* $W \subseteq \bigcup_{n=0}^{\infty}[(\Sigma^n)^{k(n)} \times \{0,1\}^{k(n)}]$ *such that for every* $k(n)$*-tuple* t *of length-*n *strings*

- *there exists* $\vec{b} \in \{0,1\}^{k(n)}$ *such that* $(t, \vec{b}) \in W$*, and*

[3] We originally proved Theorem 4.4 for a function w. Pankaj Rohatgi pointed out to us that the same proof works for relations.

- *for every* $\vec{b} \in \{0,1\}^{k(n)}$*, if* $(t, \vec{b}) \in W$ *then* $F^A_{k(n)}(t) \neq \vec{b}$*.*

Then $A \in \mathrm{NP}^W/\mathrm{poly} \cap \text{co-NP}^W/\mathrm{poly}$*.*

Proof: We show that $A \in \mathrm{NP}^W/\mathrm{poly}$. We later show how a simple trick also gets $A \in \text{co-NP}^W/\mathrm{poly}$.

We generate advice for determining membership in A for length-n strings by running the construction below. The construction generates sets $Z_{n,k(n)}$, $Z_{n,k(n)-1}$, ..., Z_{n,k_0} where $k_0 \geq 2$. All the sets constructed are sets of length-n strings and have cardinality bounded by a polynomial in n.

Two things may happen in the construction. If $k_0 = 2$ then the sets $Z_{n,2}, \ldots, Z_{n,k(n)}$ and the value of $A(z)$ for every $z \in Z_{n,2} \cup \cdots \cup Z_{n,k(n)}$ is advice for an $\mathrm{NP}^W/\mathrm{poly}$ algorithm for $A \cap \Sigma^n$. If $k_0 > 2$ then the very reason that the construction stopped before constructing $Z_{n,2}$ will yield a probabilistic $\mathrm{NP}^W/\mathrm{poly}$ algorithm for $A \cap \Sigma^n$ using the sets $Z_{n,k_0}, \ldots, Z_{n,k(n)}$ for advice. More advice removes the randomness. (We will use Theorem 11.6 from the appendix which is a variation of Schöning's proof that $\mathbf{BP} \cdot \mathcal{C} \subseteq \mathcal{C}/\mathrm{poly}$.)

Definition: Let $n, k \in \mathsf{N}$. Assume $Z_{n,k(n)}, \ldots, Z_{n,k+1}$ have already been constructed. Let $z \in \Sigma^n$, and t be a $(k-1)$-tuple of elements of Σ^n. z is *advice* for t if there exists $s \in Z_{n,k+1} \times \cdots \times Z_{n,k(n)}$ and $\vec{b} \in \{0,1\}^{k(n)}$ such that

$$((t,z,s),\vec{b}) \in W \text{ and } \vec{b}[k : k(n)] = F^A_{k(n)-k+1}(z,s).$$

Since $((t,z,s),\vec{b}) \in W$ we have $F^A_{k(n)}(t,z,s) \neq \vec{b}$. Hence, since $\vec{b}[k : k(n)] = F^A_{k(n)-k+1}(z,s)$, we have $\vec{b}[1 : k-1] \neq F^A_{k(n)}(t,z,s)[1 : k-1] = F^A_{k-1}(t)$. Let $\textsc{advisees}(z)$ be the set of $(k-1)$-tuples that z is advice for. We say that z is *advice* for a set T of $(k-1)$-tuples iff

$$|\textsc{advisees}(z) \cap T| \geq \frac{1}{4}|T|.$$

CONSTRUCTION OF ADVICE FOR Σ^n

BEGIN
$k := k(n)$
$STOP := FALSE$
While $k \geq 2$ and $STOP = FALSE$
  $T_{n,k} := (\Sigma^n)^{k-1}$
  $Z_{n,k} := \emptyset$
  While there exists a string z in Σ^n that is advice for $T_{n,k}$
    choose such a z
    $T_{n,k} := T_{n,k} - \textsc{advisees}(z)$
    $Z_{n,k} := Z_{n,k} \cup \{z\}$
  If $T_{n,k} = \emptyset$ then $k := k-1$ else $STOP := TRUE$
END

END OF CONSTRUCTION

Note that after the ith iteration $|T_{n,k}| \leq (\frac{3}{4})^i * |(\Sigma^n)^{k-1}|$. Hence there are at most $O(\log |(\Sigma^n)^{k-1}|) = O(kn)$ iterations. Since the number of elements in $Z_{n,k}$ is the number of iterations $|Z_{n,k}| = O(kn)$.

Let J be the set of n such that either $T_{n,2}$ is never defined or $T_{n,2} \neq \emptyset$. We show the following.

I. $\bigcup_{n \notin J} A \cap \Sigma^n \in \mathrm{NP}^W/\mathrm{poly}$.

II. $\bigcup_{n \in J} A \cap \Sigma^n \in \mathrm{NP}^W/\mathrm{poly}$.

These two statements together establish $A \in \mathrm{NP}^W/\mathrm{poly}$.

I. If $n \notin J$ then $T_{n,2} = \emptyset$. Hence, for every $x \in \Sigma^n$, advice was found. Therefore, for every $x \in \Sigma^n$,

$$(1) \qquad (\exists z \in Z_{n,2})(\exists s \in Z_{n,3} \times \cdots \times Z_{n,k(n)})(\exists \vec{b} \in \{0,1\}^{k(n)})$$
$$[((x,z,s),\vec{b}) \in W \text{ and } \vec{b}[2:k(n)] = F^A_{k(n)-1}(z,s)].$$

Let the advice encode the sets $Z_{n,2}, \ldots, Z_{n,k(n)}$ and the value of $A(z)$ for every $z \in Z_{n,2} \cup \cdots \cup Z_{n,k(n)}$. If $\vec{b}$ is as in equation (1) then $\vec{b}[1] \neq A(x)$. Note that for all $(t,\vec{b}) \in W$, $F^A_{k(n)}(t) \neq \vec{b}$. Hence, for all $n \notin J$, for all $x \in \Sigma^n$,

$$(2) \qquad x \in A \text{ iff}$$
$$(\exists z \in Z_{n,2})(\exists s \in Z_{n,3} \times \cdots \times Z_{n,k(n)})(\exists \vec{b} \in \{0,1\}^{k(n)})$$
$$[((x,z,s),\vec{b}) \in W \text{ and } \vec{b}[2:k(n)] = F^A_{k(n)-1}(z,s) \text{ and } \vec{b}[1] = 0].$$

Thus we can determine if $x \in \Sigma^n \cap A$ nondeterministically with oracle W, in polynomial time with polynomial advice.

II. If $n \in J$ then let k be such that $T_{n,k+1} = \cdots = T_{n,k(n)} = \emptyset$ but $T_{n,k} \neq \emptyset$. Note that $Z_{n,k}, Z_{n,k+1}, \ldots, Z_{n,k(n)}$ were all constructed. Fix $n \in J$ and $x \in \Sigma^n$.

Since $T_{n,k+1} = \emptyset$ every k-tuple of elements of Σ^n has advice in $Z_{n,k+1}$. In particular the k-tuples of the form (u,x) where $u \in (\Sigma^n)^{k-1}$ have advice in $Z_{n,k+1}$. Hence

$$(\exists z \in Z_{n,k+1})(\exists s \in Z_{n,k+2} \times \cdots \times Z_{n,k(n)})(\exists \vec{b} \in \{0,1\}^{k(n)})$$
$$[((u,x,z,s),\vec{b}) \in W \text{ and } \vec{b}[k+1:k(n)] = F^A_{k(n)-k}(z,s)].$$

We rewrite this as

$$(\exists s \in Z_{n,k+1} \times Z_{n,k+2} \times \cdots \times Z_{n,k(n)})(\exists \vec{b} \in \{0,1\}^{k(n)})$$
$$[((u,x,s),\vec{b}) \in W \text{ and } \vec{b}[k+1:k(n)] = F^A_{k(n)-k}(s)].$$

Let s_0 and $\vec{b}_0$ satisfy the above equation. Hence

(3) $\quad [((u,x,s_0),\vec{b}_0) \in W \text{ and } \vec{b}_0[k+1:k(n)] = F^A_{k(n)-k}(s_0)].$

Since $T_{n,k} \neq \emptyset$ the string x cannot serve as advice for $T_{n,k}$. Hence for $\frac{3}{4} * |T_{n,k}|$ elements $u \in T_{n,k}$

$(\forall s \in Z_{n,k+1} \times \cdots \times Z_{n,k(n)})(\forall \vec{b} \in \{0,1\}^{k(n)})$

$$[((u,x,s),\vec{b}) \notin W \text{ or } \vec{b}[k:k(n)] \neq F^A_{k(n)-k+1}(s)].$$

Since this equation holds for all s and for all $\vec{b}$ it holds for s_0 and $\vec{b}_0$. Hence for $\frac{3}{4} * |T_{n,k}|$ elements $u \in T_{n,k}$ we have

(4) $\quad [((u,x,s_0),\vec{b}_0) \notin W \text{ or } \vec{b}_0[k:k(n)] \neq F^A_{k(n)-k+1}(x,s_0)].$

Combining equations (3) and (4) we obtain that for $\frac{3}{4} * |T_{n,k}|$ elements $u \in T_{n,k}$ both $\vec{b}_0[k+1:k(n)] = F^A_{k(n)-k}(s_0)$ and $\vec{b}_0[k:k(n)] \neq F^A_{k(n)-k+1}(x,s_0)]$. Hence $A(x) = 1 \dot{-} \vec{b}_0[k]$.

To recap we have

$(\forall n \in J)(\forall x \in \Sigma^n)(\exists s_0 \in Z_{n,k+1} \times \cdots \times Z_{n,k(n)})(\exists \vec{b}_0 \in \{0,1\}^{k(n)})(\text{ for } \frac{3}{4} * |T_{n,k}|\ u \in T_{n,k})$

$$[\text{equations (3) and (4) hold hence } A(x) = 1 \dot{-} \vec{b}_0[k]].$$

Let B_n be all $\langle x,u \rangle$ such that

(5) $\quad (\exists s \in Z_{n,k+1} \times \cdots \times Z_{n,k(n)})(\exists \vec{b} \in \{0,1\}^{k(n)})$

$$[((u,x,s),\vec{b}) \in W \text{ and } \vec{b}[k+1:k(n)] = F^A_{k(n)-k}(s)] \text{ and } \vec{b}[k] = 0].$$

We have that for all $n \in J$, for all $x \in \Sigma^n$,

(6) $x \in A \Rightarrow \text{Prob}(\langle x,u \rangle \in B_n \mid u \in T_{n,k}) \geq \frac{3}{4}$;

(7) $x \notin A \Rightarrow \text{Prob}(\langle x,u \rangle \notin B_n \mid u \in T_{n,k}) \geq \frac{3}{4}$.

Let $B = \bigcup_{n \in J} B_n$. Clearly $B \in \text{NP}^W/\text{poly}$ by using for advice the sets $Z_{n,k+1}, \ldots, Z_{n,k(n)}$, and the value of $A(z)$ for every $z \in Z_{n,k+1} \cup \cdots \cup Z_{n,k(n)}$. Hence by Theorem 11.6, with $Y = \cup_{n \in J} T_{n,k}$, $A \in (\text{NP}^W/\text{poly})/\text{poly} = \text{NP}^W/\text{poly}$.

We now show that $A \in \text{co-NP}^W/\text{poly}$. We show that if the premise holds for A then it holds for $\overline{A}$. For $\vec{b} \in \{0,1\}^*$ let $\vec{b}'$ be the complement of $\vec{b}$ (i.e. if the ith place was 0 it is now 1, and if it was 1 it is now 0). Let $W' = \{(t,\vec{b}') \mid (t,\vec{b}) \in W\}$. Note that $\overline{A}$ satisfies the premise with W'. Hence $\overline{A} \in \text{NP}^{W'}/\text{poly} = \text{NP}^W/\text{poly}$. Therefore $A \in \text{co-}(\text{NP}^W/\text{poly}) = \text{co-NP}^W/\text{poly}$. ∎

A close examination of the proof yields results about what happens if W is restricted. We need the following special case of a relativized version of a lemma from Book, Balcázar, and Schöning ([8, Theorem 4.3] or [64, Theorem 4.27]).

Lemma 4.5. *If A is self-reducible and $A \in \mathrm{NP}^W/\mathrm{poly}$ then $\Sigma_2^{p,A} \subseteq \Sigma_3^{p,W}$, hence $A \in \Sigma_3^{p,W} \cap \Pi_3^{p,W}$ and $\mathrm{P}^A \subseteq \Sigma_3^{p,W} \cap \Pi_3^{p,W}$.*

Corollary 4.6. *Let $A, k(n)$ and W be as in Theorem 4.4.*

a) If $W \in \mathrm{DTIME}(T(n))$ then $A \in \mathrm{DTIME}(n^{O(k(n))}T(n))/\mathrm{poly}$.

b) If W is accepted by an $(s(n), d(n))$ circuit (the nth circuit operates on inputs $t \in (\Sigma^n)^{k(n)} \times \{0,1\}^{k(n)}$) then A is accepted by a $(n^{O(k(n))}s(n), d(n) + O(1))$ circuit (the nth circuit operates on Σ^n). The circuit for W may include MOD_a (any a) and MAJ gates.

c) If A is self-reducible then $A \in \Sigma_3^{p,W} \cap \Pi_3^{p,W}$ and $\mathrm{P}^A \subseteq \Sigma_3^{p,W} \cap \Pi_3^{p,W}$.

Proof: We first prove a useful bound. Recall that $(\forall n, k)[|Z_{n,k}| = O(kn)]$. Let d be such that $(\forall n, k)[|Z_{n,k}| \le dkn]$. Note that

$$(\forall n)[|Z_{n,k_0} \times \cdots \times Z_{n,k(n)} \times \{0,1\}^{k(n)}| \le (\prod_{i=k_0}^{k(n)} din) * 2^{k(n)} \le (2nd)^{k(n)}(k(n))!].$$

Using Stirling's formula and the fact that $k(n)$ is polynomial bounded one can show $k(n)! = n^{O(k(n))}$. Hence

$$(\forall n)[|Z_{n,k_0} \times \cdots \times Z_{n,k(n)} \times \{0,1\}^{k(n)}| \le n^{O(k(n))}].$$

The proof of Theorem 4.4 exhibited two $\mathrm{NP}^W/\mathrm{poly}$ algorithms. We show that in both cases we can obtain a $\mathrm{DTIME}(n^{O(k(n))}T(n))/\mathrm{poly}$ algorithm, and a $(n^{O(k(n)}s(n), d(n) + O(1))$ circuit .

a) Consider the algorithm in the $n \notin J$ case. In this algorithm the nondeterministic search is over $Z_{n,2} \times \cdots \times Z_{n,k(n)} \times \{0,1\}^{k(n)}$. For each possibility we make one query to W which takes $T(n)$ steps. Hence the total time a deterministic algorithm needs to simulate (given the advice) is $n^{O(k(n))}T(n)$.

Consider the algorithm in the $n \in J$ case. Note that $B \in \mathrm{NP}^W/\mathrm{poly}$ by an algorithm that nondeterministically searched $Z_{n,k+1} \times \cdots \times Z_{n,k(n)} \times \{0,1\}^{k(n)}$. For each possibility we make one query to W which takes $T(n)$ steps. Hence $B \in \mathrm{DTIME}(n^{O(k(n))}T(n))/\mathrm{poly}$. By Theorem 11.6 $A \in (\mathrm{DTIME}(n^{O(k(n))}T(n))/\mathrm{poly})/\mathrm{poly} = \mathrm{DTIME}(n^{O(k(n))}T(n))/\mathrm{poly}$.

b) If $n \notin J$ then equation 2 tells us exactly when $x \in A$. A circuit using this equation to test if $x \in A$ can be built as follows. For every

$$(z, s, \vec{b}) \in Z_{n,2} \times (Z_{n,3} \times \cdots \times Z_{n,k(n)}) \times \{0,1\}^{k(n)}$$

build a circuit that tests if

$$[((x,z,s),\vec{b}) \in W \text{ and } \vec{b}[2:k(n)] = F^A_{k(n)-1}(z,s) \text{ and } \vec{b}[1]=0].$$

(Each of these circuits is size $s(n)+O(1)$ depth $d(n)+O(1)$.) Then take the results of these circuits and OR them all together. The final circuit will have size $n^{O(k(n))}s(n)$ and depth $d(n)+O(1)$.

If $n \in J$ then equation 5 defines B_n and equations 6 and 7 tell us (using B_n) when $x \in A$ and when $x \notin A$. By reasoning similar to that above B_n can be recognized in size $n^{O(k(n))}s(n)$ and depth $d(n)+O(1)$.

Using equations (6) and (7) we can obtain a sort of probabilistic circuit for A. Usually a probabilistic circuit has the random variables going over $\{0,1\}^m$ for some m; by contrast we have that the random variables go over $T_{n,k}$. We discuss such circuits in the appendix (Section 11.2). Ajtai and Ben-Or [1] have shown how to convert (standard) probabilistic circuits to deterministic ones while only increasing the size and depth by very little. We have modified their proofs for our probabilistic circuits. Using Lemma 11.10 on the probabilistic circuit for A obtained above we can deduce that A can be recognized by a size $n^{O(k(n))}s(n)$, depth $d(n)+O(1)$ deterministic circuit. If the circuit for W had MOD_a or MAJ gates then use Lemma 11.11.

A more careful analysis actually yields circuits of size $(4n^6 \log^2(n))s(n)$ and depth $d(n)+4$.

c) This follows from Theorem 4.4 and Lemma 4.5. ∎

5. Applications: Bounded Query Classes

We apply Theorem 4.4 to obtain results about bounded query classes. In particular we show that under complexity-theoretic assumptions (e.g. $\Sigma_3^p \neq \Pi_3^p$) there are separations between bounded query classes. We actually prove stronger results from which separations will be clear. For example, we show that if $\Sigma_3^p \neq \Pi_3^p$ and $f(n) = O(\log n)$ then, for all X, $\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}$; hence, under these hypotheses, $\mathrm{PF}^{\mathrm{NP}}_{(f(n)-1)\text{-tt}} \subset \mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}}$ and $\mathrm{PF}^{\mathrm{NP}}_{(f(n)-1)\text{-T}} \subset \mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-T}}$.

Theorem 5.1. *Let A be a set. Let $f,k \in \mathrm{PF}$ be such that the following hold.*

i. $(\overset{\infty}{\forall} m)[f(mk(m)) \leq k(m)]$.

ii. $(\exists X)[\mathrm{PF}^{A}_{f(n)\text{-tt}} \subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}]$.

Then there exists W such that the following occur.

a) $W \in \text{co-NP} \cap \mathrm{DTIME}(n^{O(1)}2^{f(n)})$.

b) The set W, together with the function $k(m)$, satisfy the premise of Theorem 4.4.

c) $A \in \mathrm{NP}^W/\mathrm{poly} \cap \text{co-NP}^W/\mathrm{poly} \subseteq \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$. *If A is self-reducible then $A \in \Sigma_4^p \cap \Pi_4^p$.*

d) $A \in \mathrm{DTIME}(n^{O(k(n))}2^{f(n)})/\mathrm{poly}$.

e) *If* f *is* $O(\log n)$ *then* $W \in \mathrm{P}$ *so* $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$. *If* A *is self-reducible then* $A \in \Sigma_3^p \cap \Pi_3^p$.

Proof: We define W so that a, b hold. Items b, c, d, e will follow from a, b, Theorem 4.4, and Corollary 4.6.

We define an auxiliary function q as follows. On input $z \in \Sigma^*$, find m such that $z = x_1 x_2 \cdots x_{k(m)}$ and, for all i, $|x_i| = m$ (if no such m exists then $q(z) = 0$). Let $q(z) = F^A_{f(mk(m))}(x_1, \ldots, x_{f(mk(m))})$ (this is where we use $f(mk(m)) \leq k(m)$). Note that on input of length $mk(m)$ (any m), q can be computed with $\leq f(mk(m))$ parallel queries to A, and on inputs of any other length no queries are needed. Hence $q \in \mathrm{PF}^A_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}$. By Fact 2.15 q is $2^{f(n)-1}$ enumerable. Let $e(z, -) \in \mathrm{PF}$ be the function that enumerates (in the sense of Definition 2.8) at most $2^{f(|z|)-1} < 2^{f(|z|)}$ possibilities for $q(z)$.

We define W to be the union over m of the set of ordered pairs $(t, \vec{b})$ such that the following hold.

i. $t = \langle x_1, \ldots, x_{k(m)} \rangle$, $\vec{b} = b_1 b_2 \cdots b_{k(m)}$ where $(\forall i)[|x_i| = m$ and $b_i \in \{0, 1\}]$.

ii. Let $z = x_1 \cdots x_{k(m)}$. $(\forall i < 2^{f(mk(m))})[e(z, i) \neq b_1 b_2 \cdots b_{f(mk(m))}]$.

We show that $k(m)$ and W satisfy the premise of Theorem 4.4. Let $t = \langle x_1, \ldots, x_{k(m)} \rangle$ and $z = x_1 \cdots x_{k(m)}$. Since $e(z, -)$ enumerates at most $2^{f(mk(m))-1} < 2^{f(mk(m))}$ possibilities, there exists $\vec{b}$ such that $(t, \vec{b}) \in W$. Hence the first premise of Theorem 4.4 is satisfied. If $(t, \vec{b}) \in W$ then, by the definition of W, the first $f(mk(m))$ bits of $\vec{b}$ differ from the first $f(mk(m))$ bits of $F^A_{k(m)}$. Hence $F^A_{k(m)}(t) \neq \vec{b}$, so the second premise of Theorem 4.4 is satisfied.

It is easy to see that $W \in \mathrm{DTIME}(n^{O(1)}2^{f(n)})$ and $W \in \mathrm{co\text{-}NP}$. ∎

Krentel [51] proved (assuming $\mathrm{P} \neq \mathrm{NP}$) that if $f \in \mathrm{PF}$, f nondecreasing, such that $f(n) \leq (1-\epsilon)\log n$, then $\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-T}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}$ for any X. [4]

Corollary 5.2.*ii* extends his result.

Corollary 5.2. *Let* $f \in \mathrm{PF}$, f *nondecreasing, such that* $f(n) = O(\log n)$.

i. *If* $(\exists X)[\mathrm{PF}^A_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$, *then* $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$ *and* $A \in \mathrm{DTIME}(n^{O(\log n)})/\mathrm{poly}$. *If* A *is self-reducible then* $A \in \Sigma_3^p \cap \Pi_3^p$.

ii. *Let* $i \geq 1$. *If* $\Sigma_3^p \neq \Pi_3^p$ *then* $(\forall X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$. *(In particular* $(\forall X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$.*)*

iii. *If* $\mathrm{SAT} \notin \mathrm{DTIME}(n^{O(\log n)})/\mathrm{poly}$ *then* $(\forall X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$.

---

[4] As noted before, Krentel actually showed this just for $f(n) \leq \frac{1}{2}\log n$, but his proof can be modified to $f(n) \leq (1-\epsilon)\log n$. See [12].

iv. *If* $\Sigma_3^p \neq \mathrm{PSPACE}$ *then* $(\forall X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$.

Proof:
i. Let $k(m) = \lfloor d \log m \rfloor$ where d is large enough so that $f(mk(m)) \leq k(m)$ (such exists since $k(m) = O(\log m)$). By Theorem 5.1 $A \in \mathrm{NP/poly} \cap \text{co-NP/poly}$, $A \in \mathrm{DTIME}(n^{O(\log n)})/\mathrm{poly}$, and if A is self-reducible $A \in \Sigma_3^p \cap \Pi_3^p$.
ii. If $(\exists X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$ then, by part i, $\mathrm{NP} \subseteq \text{co-NP/poly}$. Hence [78] $\Sigma_3^p = \Pi_3^p$.
iii. If $(\exists X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$ then, by part i, $\mathrm{SAT} \in \mathrm{DTIME}(n^{O(\log n)})/\mathrm{poly}$.
iv. If $(\exists X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$ then, by part (i), $\mathrm{QBF} \in \Sigma_3^p \cap \Pi_3^p$ hence $\Sigma_3^p = \mathrm{PSPACE}$. ∎

We are currently unable to extend Corollary 5.2 to $f(n) \neq O(\log n)$. However we can prove a similar result about $f(n) = n^\epsilon$ queries to a Σ_i^p-complete oracle $(i \geq 3)$ or a PSPACE-complete oracle.

Corollary 5.3. *Let* ϵ *be a positive real number such that* $0 \leq \epsilon < 1$. *Let* f *be a function such that* $f \in \mathrm{PF}$, f *is nondecreasing, and* $f(n) \leq n^\epsilon$.

- *i.* *If* $(\exists X)[\mathrm{PF}^A_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$, *then* $A \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$. *If* A *is self-reducible then* $A \in \Sigma_4^p \cap \Pi_4^p$.
- *ii.* *Let* $i \geq 3$. *If* $\Sigma_4^p \neq \Pi_4^p$ *then* $(\forall X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$.
- *iii.* *If* $\Sigma_4^p \neq \mathrm{PSPACE}$ *then* $(\forall X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$.

Proof:
i. Let $k(m) = m^{\frac{\epsilon}{1-\epsilon}}$. Then $f(mk(m)) \leq k(m)$. By Theorem 5.1 $A \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$ and if A is self-reducible then $A \in \Sigma_4^p \cap \Pi_4^p$.
ii. If $(\exists X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$ then, by part (i), $\Sigma_i^p \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$. By Lemma 4.5 we obtain $\Sigma_2^{p,\Sigma_3^p} \subseteq \Sigma_4^p$, hence $\Sigma_4^p = \Pi_4^p$.
iii. If $(\exists X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$. then, by part (i), $\mathrm{PSPACE} \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$. By [8, Theorem 4.3] $\Sigma_2^{p,\mathrm{PSPACE}} \subseteq \Sigma_4^p$. Hence $\mathrm{PSPACE} \subseteq \Sigma_4^p$. ∎

In [12], using special properties of SAT we showed that $\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}$ for any X unless $\mathrm{NTIME}(n^{f(n^{O(1)})}) = \mathrm{RTIME}(n^{f(n^{O(1)})})$. As a Corollary to Theorem 5.1 we derive a result with a similar flavor, but without using any special properties of the oracle.

Corollary 5.4. *Let* $f(n) = \log^{O(1)} n$. *If* $(\exists X)[\mathrm{PF}^A_{f(n)\text{-tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)\text{-T}}]$ *then* $A \in \mathrm{DTIME}(n^{\mathrm{polylog}\, n})/\mathrm{poly}$.

Proof: By assumption, $f(n) = O(\log^i n)$ for some i. Let d be such that if $k(m) = d(\log^{i+1} m)$ then $(\forall m)[f(mk(m)) \leq k(m)]$. By Theorem 5.1 $A \in \mathrm{DTIME}(n^{O(k(n))}2^{f(n)}) \subseteq \mathrm{DTIME}(n^{\mathrm{polylog}\, n})/\mathrm{poly}$. ∎

6. Applications: Circuit Complexity

We consider circuits that make oracle queries. Our circuit model allows negation, unbounded fan-in and-gates, and unbounded fan-in or-gates. Following Wilson [75], we allow a circuit C to query an oracle X by permitting the circuit to contain an X-gate (which computes membership in X). The number of queries to X is the number of X-gates the circuit contains.

Notation 6.1. We have several conventions about how to interpret a circuit. If a circuit is intended to recognize a set then we assume that the nth circuit takes elements of Σ^n as input. If a circuit is intended to compute $F_{k(n)}^{A^{=n}}$ then we assume that the nth circuit takes elements of $(\Sigma^n)^{k(n)}$ as input. If a circuit is intended to compute a set of type W from Theorem 4.4 then we assume that the nth circuit takes as input elements of $(\Sigma^n)^{k(n)} \times \{0,1\}^{k(n)}$.

Let $k(n) \in \mathrm{PF}$. Let $A \subseteq \Sigma^*$. Clearly there exists a polynomial sized constant depth circuit $\{C_n\}_{n=1}^{\infty}$ such that C_n makes $k(n)$ queries to A and

$$(\forall n)(\forall t \in (\Sigma^n)^{k(n)})[C_n(t) = F_{k(n)}^{A^{=n}}(t)].$$

For which sets A can we compute $F_{k(n)}^{A^{=n}}$ with a small oracle circuit with $k(n)-1$ queries to some X? We show that such an A must be easy to compute with a (non-oracle) circuit. As a corollary we obtain an extension of a result by Cai [23] on PARITY.

Theorem 6.2. *Let $k(n) \in \mathrm{PF}$. Let $A \subseteq \Sigma^*$. If there is an $(s(n), d(n))$ oracle circuit family $\{C_n\}_{n=1}^{\infty}$ such that C_n computes $F_{k(n)}^{A^{=n}}$ while making only $k(n)-1$ oracle queries to some oracle X then A is recognized by an $(n^{O(k(n))}s(n)^2, d(n)+O(1))$ circuit family. The original circuit family may use MOD_a and MAJ gates.*

Proof: Assume the hypothesis. Equivalently, there exist, for each n, $2^{k(n)-1}$ circuits $C_n^1, \ldots, C_n^{2^{k(n)-1}}$ each having size $s(n)$ and depth $d(n)$ such that for every $t \in (\Sigma^n)^{k(n)}$ there exists $i \leq 2^{k(n)-1}$ such that $F_{k(n)}^{A}(t) = C_n^i(t)$. Let

$$W = \bigcup_{n=1}^{\infty} \{(t, \vec{b}) \mid t \in (\Sigma^n)^{k(n)}, \vec{b} \in \{0,1\}^{k(n)} \text{ and } (\forall i \leq 2^{k(n)-1})[\vec{b} \neq C_n^i(t)]\}.$$

It is easy to see that W can be recognized by a $(2^{k(n)}s(n), d(n) + O(1))$ circuit family. By Corollary 4.6 A can be recognized by an $(n^{O(k(n))}s(n)^2, d(n)+O(1))$ circuit family. ∎

The following corollary was originally proven by Cai [23].

Corollary 6.3. *Let $k(n) = n^{o(1)}$. Assume there is an $(s(n), d)$ oracle circuit family $\{C\}_{n=1}^{\infty}$ such that C_n computes $F_{k(n)}^{\mathrm{PARITY}^{=n}}$ with $k(n)-1$ oracle queries to some X. Then $s(n) = 2^{n^{\Omega(1)}}$. (The original circuit family is allowed to use MOD_a gates where $a \geq 3$ is a fixed prime.)*

Proof: By Theorem 6.2 PARITY can be recognized by an $(n^{O(k(n))}s(n)^2, d+O(1))$ circuit. Since constant depth circuits for parity require size $2^{n^{\Omega(1)}}$ [21, 38, 77] we have that $n^{k(n)}s(n)^2 = 2^{n^{\Omega(1)}}$. Since $k(n) = n^{o(1)}$ we obtain $s(n) = 2^{n^{\Omega(1)}}$. (The lower bound for parity holds even when the circuit is allowed to use MOD_a gates where $a \geq 3$ is some fixed prime [21, 67].) ∎

7. Applications: Enumerability

Cai and Hemachandra [26] proved that #SAT is not n^k-enumerable unless $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. We use Theorems 4.4 and 3.31 to prove many functions (under suitable assumptions). We also obtain their result as a consequence of our Theorems and [71, Theorem 4.1].

One of the functions we study will be PERM01.

Definition 7.1. Let PERM01 denote the function that, given a 0-1 matrix, returns the permanent of that matrix. PERM01 is known to be #P complete [72].

We will compare the complexity of PERM01 and #SAT. Even though they are both #P-complete they seem to behave differently with respect to enumeration.

Definition 7.2. We say that $g \leq_{\mathrm{m}}^{p} f$ if there exists $S, T \in \mathrm{PF}$ such that $g(x) = S(x, f(T(x)))$. Intuitively, T maps x to an element $T(x)$ such that $f(T(x))$ has information that allows one to compute $g(x)$, and S extracts that information. Hardness and completeness are defined accordingly. In this paper we will refer to T as the reduction and supress the role of S.

Definition 7.3. Let $g : \Sigma^* \to \mathsf{N}$. Then let

$$\mathrm{bit}_g = \{\langle x, 0^i\rangle : \text{the } i\text{-th bit of the binary expansion of } g(x) \text{ is } 1\}.$$

Notation 7.4. A *quantified Boolean formula* is a Boolean formula where some of the variables (though not necessarily all) are quantified over. When writing down a quantified Boolean formula with some variables free we will write $\phi(x_1, \ldots, x_m)$ to denote that $x_1, \ldots, x_m$ are the free variables. Note that for any $\vec{b} \in \{0,1\}^m$, $\phi(\vec{b})$ is either true or false. We denote that $\phi(\vec{b})$ is true (false) by $\phi(\vec{b}) = 1$ ($\phi(\vec{b}) = 0$).

We will be concerned with computing many queries to a function g by making just one query to g. The next definition defines a function g_q^+ that reports the answers to many queries to g. The next two sections present lemmas which allow you to show that, for some functions g, g_q^+ can be computed with one query to g.

Definition 7.5. Let g be any function and q be any polynomial. The function g_q^+ is defined on the set $\cup_{m=0}^{\infty}(\Sigma^{\leq m})^{q(m)}$ as $g_q^+(x_1, \ldots, x_{q(m)}) = \langle g(x_1), \ldots, g(x_{q(m)})\rangle$.

7.1. Lemmas on Functions associated to #C

Definition 7.6. Let C be a class of sets. Then

$$\#\mathrm{C} = \{f : (\exists R \in \mathrm{C})(\exists k)[f(x) = |\{z : |z| = |x|^k \text{ and } R(x,z)\}|\,]\}.$$

Example 7.7. The following are examples of classes C and functions f such that f (when modified slightly to fit the definition of #C exactly) is complete for #C.

i. $\mathrm{C} = \mathrm{P}$ and $f = \#\mathrm{SAT}$, the function that, given a Boolean formula $\phi(x_1, \ldots, x_n)$, returns $|\{\vec{b} \in \{0,1\}^n \ : \ \phi(\vec{b}) = 1\}|$.

ii. $\mathrm{C} = \Sigma_i^p$ and $f = \#\mathrm{QBF}_i$, the function that, given a quantified Boolean formula $\phi(x_1, \ldots, x_n)$ which (1) has n free variables, (2) starts with an $\exists$, and (3) has at most $i-1$ alternations of quantifiers, returns $|\{\vec{b} \in \{0,1\}^n \ : \ \phi(\vec{b}) = 1\}|$.

iii. $\mathrm{C} = \mathrm{PSPACE}$ and $f = \#\mathrm{QBF}$, the function that, given a quantified Boolean formula $\phi(x_1, \ldots, x_n)$, returns $|\{\vec{b} \in \{0,1\}^n \ : \ \phi(\vec{b}) = 1\}|$.

iv. Let $\mathrm{C} = \mathrm{US}$, (defined by [18]) which is the class of all sets of the form

$$\{x \mid \text{ there is exactly one path such that } N(x) \text{ accepts } \}$$

where N is a nondeterministic polynomial time Turing machine. Let f be the function that, when given a Boolean formula $\phi(x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2})$, returns $|\{\vec{b} \in \{0,1\}^{n_2} \ : \ \phi(x_1, \ldots, x_{n_1}, \vec{b}) \in \mathrm{USAT}\}|$. (Recall that USAT is the set of formulas that have exactly one satisfying assignment.)

v. This is a generalization of the previous example. It is due to Lozano and Ogiwara [58]. If N is a nondeterministic polynomial time machine and $x \in \Sigma^*$ then let $\#acc_N(x)$ be the number of accepting paths of N on x, and $\#rej_N(x)$ be the number of rejecting paths of N. Let Q be a polynomial time decidable predicate on $\mathsf{N} \times \mathsf{N}$. A set A is in the class $\mathrm{C} = \mathrm{QP}$ if there exists a nondeterministic time Turing machine N such that $A = \{x \mid Q(\#acc_N(x), \#rej_N(x))\}$. Let f be the function that, when given a Boolean formula $\phi(x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2})$, returns

$$|\{\vec{b} \in \{0,1\}^{n_2} \ : \ Q(a, 2^{n_1} - a) \ \text{ where } a = \#\mathrm{SAT}(\phi(x_1, \ldots, x_{n_1}, \vec{b}))\}|.$$

vi. Note that the following classes are of the type specified in the above item: $\oplus$P (defined in [60]), $\mathrm{MOD_kP}$ (defined in [16, 25]), co-$\mathrm{MOD_kP}$, $\mathrm{C_{=}P}$ (defined in [73]), and PP (defined in [34]).

The following lemma translates self-reducibility from classes of sets to the associated counting classes.

Notation 7.8. Let M_e denote the e^{th} Turing machine. Let $M_{e,s}(x)$ denote the output of machine M_e after running for s steps on input x.

Lemma 7.9. *Assume that* C *is closed under* $\leq_{\rm m}^{p}$*-reductions and that* C *has a self-reducible* $\leq_{\rm m}^{p}$*-complete set* A*. Then* $\#$C *has a* $\leq_{\rm m}^{p}$*-complete function* g *such that* bit_g *is self-reducible.*

Proof: Let

$$g(e,x,y,0^c,0^s) = |\{z : |z| = c \text{ and } M_{e,s}(x,yz) \in A\}|.$$

We show that $g \in \#$C. Since C is closed under $\leq_{\rm m}^{p}$-reductions and $A \in$ C the set

$$\{\langle\langle e,x,y,0^c,0^s\rangle, z0^{|\langle e,x,y,0^c,0^s\rangle|-c}\rangle \mid M_{e,s}(x,yz) \in A\}$$

is in C. Hence $g \in \#$C.

We show that g is $\#$C-hard. If $f \in \#$C then let $R \in$ C, $k \in \mathsf{N}$ be as in the definition of $\#$C. Since A is $\leq_{\rm m}^{p}$-complete for C, $R \leq_{\rm m}^{p} A$. Let e be such that M_e is the reduction. Let q_e be the polynomial that bounds the run time of M_e. Note that $f(x) = g(e,x,\lambda,0^{|x|^k},0^{p_e(|x|)})$. Hence g is $\leq_{\rm m}^{p}$-complete for $\#$C.

We show that bit_g is self-reducible. When $c > 0$

$$\begin{aligned}&|\{z : |z| = c \text{ and } M_{e,s}(x,yz) \in A\}| = \\ &\quad |\{z : |z| = c-1 \text{ and } M_{e,s}(x,y0z) \in A\}| + |\{z : |z| = c-1 \text{ and } M_{e,s}(x,y1z) \in A\}|.\end{aligned}$$

When $c = 0$

$$|\{z : |z| = c \text{ and } M_{e,s}(x,yz) \in A\}| = A(M_{e,s}(x,y)).$$

Hence

$$g(e,x,y,0^c,0^s) = \begin{cases} g(e,x,y0,0^{c-1},0^s) + g(e,x,y1,0^{c-1},0^s) & \text{if } c > 0; \\ A(M_{e,s}(x,y)) & \text{if } c = 0. \end{cases}$$

The self-reducible algorithm for bit_g operates as follows. If $c > 0$ then $\mathrm{bit}_g(\langle\langle e,x,y,0^c,0^s\rangle,0^i\rangle)$ can be computed from

$$\mathrm{bit}_g(\langle\langle e,x,y0,0^{c-1},0^s\rangle,0^1\rangle),\ldots,\mathrm{bit}_g(\langle\langle e,x,y0,0^{c-1},0^s\rangle,0^i\rangle),$$

$$\mathrm{bit}_g(\langle\langle e,x,y1,0^{c-1},0^s\rangle,0^1\rangle),\ldots,\mathrm{bit}_g(\langle\langle e,x,y1,0^{c-1},0^s\rangle,0^i\rangle)$$

by using the above equation for g. If $c = 0$ then $\mathrm{bit}_g(\langle\langle e,x,y,0^0,0^s\rangle,0^i\rangle)$ can be computed from $A(M_{e,s}(x,y))$. Since A is self-reducible this can be calculated by a polynomial time Turing machine that makes queries $q \prec_A M_{e,s}(x,y)$ (where $\prec_A$ is the ordering used to make A self-reducible). We can transform these queries into queries to bit_g as follows. Let M_j be a fixed Turing machine such that $M_j(x,y) = x$ and M_j runs in $s(|\langle x,y\rangle|)$ steps (s is some polynomial that depends on details of the model of computation). Then $A(q)$ iff $\mathrm{bit}_g(\langle\langle j,q,\lambda,0^0,0^{s(|\langle q,\lambda\rangle|)}\rangle,0^1\rangle)$.

The algorithm given above is a self-reduction for bit_g using the following ordering.

$$\langle\langle e',x',y',0^{c'},0^{s'}\rangle,0^{i'}\rangle \prec \langle\langle e,x,y,0^c,0^s\rangle,0^i\rangle$$

iff one of the following holds.

i. $e' = e$, $x' = x$, $|y| \le |y'| + c'$, $c' < c$, $s' = s$, $i' \le i$. (This is used when we are decreasing c and increasing y.)

ii. $e' = j$, $e \ne j$, $x' \preceq_A M_{e,s}(x,y)$, $y' = \lambda$, $c' = c = 0$, $s' = s(|\langle x', \lambda\rangle|)$, $i' \le i$. (When $x' = M_{e,s}(x,y)$ this is used when we've just made $c = 0$ and need to go to the algorithm for A. We need to include the $x' \preceq_A M_{e,s}(x,y)$ so that the order is transitive.)

iii. $e' = e = j$, $x' \prec_A x$, $y = y' = \lambda$, $c = c' = 0$, $s' = s(|\langle x, \lambda\rangle|)$, $i' \le i$. (We use this when we are using that A is self-reducible.)

The number of elements in a chain with top element $\langle\langle e, x, y, 0^c, 0^s\rangle, 0^i\rangle$ is at most $(i+1)(c + |\{x' : x' \prec_A x\}|)$. Since $\prec_A$ has polynomially bounded chains, so does $\prec$. ∎

For the next lemma we use the following conventions to make the notation easier. *Convention:* We modify our Turing machines such that if M_e is any Turing machine and $x \in \Sigma^*$ then the following hold.

i. If M_e does not halt within s steps then we define $M_{e,s}(x) = 0$.

ii. If $z \in \Sigma^* 0$ then $M_e(x, z) = 0$.

Lemma 7.10. *Assume that* C *is closed under* $\le_{\mathrm{m}}^p$*-reductions and that* C *has a self-reducible* $\le_{\mathrm{m}}^p$*-complete set* A*. Then* #C *has a* $\le_{\mathrm{m}}^p$*-complete function* g *such that the following hold.*

i. bit_g *is self-reducible.*

ii. For all polynomials q, $g_q^+ \le_{\mathrm{m}}^p g$ *via some reduction* T_q*. Furthermore, there is a degree* a *such that for all polynomials* q *of degree* $\ge a$ $(\forall z)[|T_q(z)| \le O(|z|^{2+\frac{1}{deg(q)}})]$.

Proof: Let g be the function from Lemma 7.9. By Lemma 7.9 g is #C-complete and bit_g is self-reducible. We show property (ii).

Let q be some fixed polynomial. We define a set A' and Turing machines $M_{e''}$ and $M_{e'}$.

Let

$$A' = \bigcup_{m=0}^{\infty} \{\langle x_1, \ldots, x_m\rangle \ : \ (\exists i)[x_i \in A - \{0\} \text{ and } (\forall j \ne i)[x_j = 0]]\}.$$

Clearly $A' \le_{\mathrm{m}}^p A$ via a linear reduction T'. Since C is closed under $\le_{\mathrm{m}}^p$-reductions $A' \in$ C.

We define $M_{e''}$ on strings of a certain form. On strings not of that form $M_{e''}$ returns λ. Let $FORM_i$ be the union over m, as $q(m) \ge i$, of strings of the form

$$\langle \ \langle\langle e_1, x_1, y_1, 0^{c_1}, 0^{s_1}\rangle, \ \ldots, \langle e_{q(m)}, x_{q(m)}, y_{q(m)}, 0^{c_{q(m)}}, 0^{s_{q(m)}}\rangle\rangle, \\ \langle 0^{m+c_1}, 0^{2m+c_2}, \ldots, 0^{(i-1)m+c_{i-1}}, z_i w_i, 0^{(i+1)m+c_{i+1}}, \ldots, 0^{mq(m)+c_{q(m)}}\rangle \ \rangle$$

where $|z_i| = c_i$, $z_i \notin 0^*$, and $|w_i| = im$. (We need $z_i \notin 0^*$ so that all the $FORM_i$ sets are disjoint.)

Let $FORM = \cup_i FORM_i$. Note that testing if a string t is in $FORM$ can be done by scanning t, finding $q(m)$, and inverting it to find m. This can be done in $O(|t|) + Inv_q(q(m))$ where Inv_q is the time it takes to invert $q(m)$. Note that Inv_q is a polynomial whose degree is independent of q.

For all $t \in FORM$ let

$$M_{e''}(t) = \langle 0, 0, \ldots, 0, M_{e_i,s_i}(x_i, y_i z_i), 0 \ldots, 0\rangle.$$

(I.e., $i-1$ 0's followed by $M_{e_i,s_i}(x_i, y_i z_i)$ followed by $q(m) - i - 1$ 0's.)

We are concerned with the run time of $M_{e''}$. Let $SIM(e, s)$ bound the number of steps it takes to simulate M_e for s steps (this is independent of the input). We can assume that $SIM(e, s)$ is a polynomial in $LENGTH(e), s$. To compute $M_{e''}(t)$ we first read the input and check if it is in the correct form, and then carry out the indicated simulation. Hence the runtime of $M_{e''}(t)$ is

$$O(|t|) + Inv_p(q(m)) + SIM(e_i, s_i) = O(|t|) + Inv_p(q(m)) + O(SIM(m, m)).$$

We denote this runtime by $q_{e''}(|t|)$. Note that $|t| = \Omega(m(q(m))^2)$. Hence

$$q_{e''}(|t|) = \begin{cases} O(SIM(|t|, |t|)) + Inv_p(p(|t|)) & \text{if } mq(m)^2 < SIM(m, m) + Inv_p(q(m)); \\ O(|t|) & \text{otherwise.} \end{cases}$$

Hence $\deg(q_{e''})$ is independent of $deg(q)$, and when q is of large enough degree $q_{e''}$ is linear.

Let $M_{e'}$ be defined as $M_{e'}(t) = T'(M_{e''}(t))$. Since T' is linear the runtime of $M_{e'}(t)$ is $O(p_{e''}(|t|))$. We denote this runtime by $q_{e'}(|t|)$. The degree of $q_{e'}$ is independent of $deg(q)$, and when q is of large enough $q_{e'}$ is linear.

We now show that $g_q^+ \leq_{\mathrm{m}}^p g$. Let the input be

$$\langle\ \langle e_1, x_1, y_1, 0^{c_1}, 0^{s_1}\rangle, \ldots, \langle e_{q(m)}, x_{q(m)}, y_{q(m)}, 0^{c_{q(m)}}, 0^{s_{q(m)}}\rangle\ \rangle.$$

where $(\forall i)[|\langle e_i, x_i, y_i, 0^{c_i}, 0^{s_i}\rangle| \leq m]$.

We formulate a query to ask g. Let

$$\begin{aligned} x = &\ \ \langle\ \langle e_1, x_1, y_1, 0^{c_1}, 0^{s_1}\rangle, \ldots, \langle e_{q(m)}, x_{q(m)}, y_{q(m)}, 0^{c_{q(m)}}, 0^{s_{q(m)}}\rangle\ \rangle; \\ c = &\ \ |\langle 0^{m+c_1}, 0^{2m+c_2}, \ldots, 0^{mq(m)+c_{q(m)}}\rangle|; \\ s = &\ \ p_{e'}(|\langle x, 0^c\rangle|); \\ u = &\ \ \langle e', x, \lambda, 0^c, 0^s\rangle. \end{aligned}$$

Note that

$$g(u) = |\{z \ : \ |z| = c \text{ and } M_{e',s}(x, z) \in A\}|.$$

Since s is larger than the runtime of $M_{e'}$ on (x, z) for any $|z| = c$ we can replace $M_{e',s}(x, z)$ with $M_{e'}(x, z)$. Hence

$$
\begin{aligned}
g(u) = \; & |\{z \;:\; |z| = c \text{ and } M_{e'}(x,z) \in A\}| \\
= \; & |\{z \;:\; |z| = c \text{ and } T'(M_{e''}(x,z)) \in A\}| \\
= \; & |\{z \;:\; |z| = c \text{ and } M_{e''}(x,z) \in A'\}|.
\end{aligned}
$$

In order for $M_{e''}(x,z) \in A'$ we first need that $\langle x,z\rangle \in FORM$. That requires that there is an i such that $\langle x,z\rangle \in FORM_i$. Note that the i is unique. Hence

$$
|\{z \;:\; |z| = c \text{ and } M_{e''}(x,z) \in A'\}| = \sum_{i=1}^{q(m)} |\{z \;:\; \langle x,z\rangle \in FORM_i \text{ and } M_{e''}(x,z) \in A'\}|.
$$

If $\langle x,z\rangle \in FORM_i$ then

$$
\begin{aligned}
z = \; & \langle 0^{m+c_1}, 0^{2m+c_2}, \ldots, 0^{(i-1)m+c_{i-1}}, z_i w_i, 0^{(i+1)m+c_{i+1}}, \ldots, 0^{q(m)m+c_{q(m)}}\rangle \\
& (\text{where } |z_i| = c_i, z_i \notin 0^*, \text{ and } |w_i| = im); \\
M_{e''}(x,z) = \; & \langle 0, 0, \ldots, 0, M_{e_i,s_i}(x_i, y_i z_i), 0, \ldots, 0\rangle.
\end{aligned}
$$

By the definition of A' we have $M_{e''}(x,z) \in A'$ iff $M_{e_i,s_i}(x_i, y_i z_i) \in A - \{0\}$. By our convention on Turing machines, there are no $z \in 0^*$ such that $M_{e,s}(x, yz) \in A - \{0\}$. Using both of these facts we obtain the following:

$$
\begin{aligned}
|\{z \;:\; \langle x,z\rangle \in \; & FORM_i \text{ and } M_{e''}(x,z) \in A'\}| \\
= \; & |\{z_i \;:\; |z_i| = c_i, z_i \notin 0^* \text{ and } M_{e_i,s_i}(x_i, y_i z_i) \in A\}| * |\{w_i \in \Sigma^{im}\}| \\
= \; & |\{z_i \;:\; |z_i| = c_i \text{ and } M_{e_i,s_i}(x_i, y_i z_i) \in A\}| * |\{w_i \in \Sigma^{im}\}| \\
= \; & g(\langle e_i, x_i, y_i, 0^{c_i}, 0^{s_i}\rangle) * (|\Sigma|)^{im}.
\end{aligned}
$$

Putting this all together we obtain

$$
\begin{aligned}
g(u) = \; & |\{z \;:\; |z| = c \text{ and } M_{e',s}(x,z) \in A\}|; \\
= \; & |\{z \;:\; |z| = c \text{ and } M_{e'}(x,z) \in A\}|; \\
= \; & |\{z \;:\; |z| = c \text{ and } T'(M_{e''}(x,z)) \in A\}|; \\
= \; & |\{z \;:\; |z| = c \text{ and } M_{e''}(x,z) \in A'\}|; \\
= \; & \textstyle\sum_{i=1}^{q(m)} |\{z \;:\; \langle x,z\rangle \in FORM_i \text{ and } M_{e''}(x,z) \in A'\}|; \\
= \; & \textstyle\sum_{i=1}^{q(m)} |\{z_i \;:\; |z_i| = c_i, \; z_i \notin 0^*, \text{ and } M_{e_i,s_i}(x_i, y_i z_i) \in A\}| * |\{w_i \in \Sigma^{im}\}|; \\
= \; & \textstyle\sum_{i=1}^{q(m)} |\{z_i \;:\; |z_i| = c_i \text{ and } M_{e_i,s_i}(x_i, y_i z_i) \in A\}| * |\{w_i \in \Sigma^{im}\}|; \\
= \; & g(\langle e_i, x_i, y_i, 0^{c_i}, 0^{s_i}\rangle) * (|\Sigma|)^{im}.
\end{aligned}
$$

Since $(\forall i)[g(\langle e_i, x_i, y_i, 0^{c_i}, 0^{s_i}\rangle) \leq 2^{c_i} < (|\Sigma|)^m]$, the values of $g(\langle e_i, x_i, y_i, 0^{c_i}, 0^{s_i}\rangle)$ can be deduced from $g(u)$.

We assume that q is of high enough degree so that $q_{e'}$ is linear. We bound $|u|$ in terms of n, the length of the original input. Note that

$$
\begin{aligned}
|x| = \; & n; \\
c = \; & O(mq(m)^2) + \textstyle\sum_{i=1}^{q(m)} c_i = O(n + mq(m)^2); \\
s = \; & p_{e'}(O(|x| + c)) = O(p_{e'}(n + mq(m)^2)) = O(n + mq(m)^2); \\
|u| = \; & |\langle e', x, \lambda, 0^c, 0^s\rangle| = O(n + mq(m)^2).
\end{aligned}
$$

Since $n = \Omega(q(m))$ the runtime can be bounded by $O(n^{2+\frac{1}{deg(q)}})$. ∎

7.2. Lemmas on Functions associated to Formulas

We now show that for certain functions g we have, for any polynomial q, $g_q^+ \leq_{\mathrm{m}}^p g$ with very little blowup in size. In particular we will be looking at #SAT, $\#\mathrm{QBF}_i$, and #QBF as defined in Example 7.7.

Cai and Hemachandra [27] proved that $\#\mathrm{SAT}_q^+ \leq_{\mathrm{m}}^p \#\mathrm{SAT}$ though the idea is essentially due to Papadimitriou and Zachos [60]. Their reduction causes a polynomial blowup of size. We show that $\#\mathrm{QBF}_q^+ \leq_{\mathrm{m}}^p \#\mathrm{QBF}$ with very little blowup in size.

Lemma 7.11. *There exists $T \in \mathrm{PF}$ such that the following hold.*

i. T takes as input a finite sequence $\phi_1, \ldots, \phi_q$ of quantified Boolean formulas that have the same number of free variables.

ii. T outputs a formula ϕ such that the following hold.

(a) Knowing $\#\mathrm{QBF}(\phi)$ yields $\langle \#\mathrm{QBF}(\phi_1), \ldots, \#\mathrm{QBF}(\phi_q)\rangle$.

(b) $|\phi| \leq O(|\langle \phi_1, \ldots, \phi_q \rangle|)$.

Proof: Given $\phi_1, \ldots, \phi_q$ we describe how to construct ϕ. Let m be the number of variables in each ϕ_i. Let the variables of ϕ_i be $x_{i1}, \ldots, x_{im}$. We assume without loss of generality that q is a power of 2.

We will first construct a formula ϕ' that satisfies *ii.a* but is too long to satisfy *ii.b*. We then show how to obtain an equivalent formula that is shorter.

Let $i \rightarrow v_i$ be a bijection from $\{1, \ldots, q\}$ to $\{0,1\}^{\log q}$. Let $z_1, \ldots, z_{\log q}$ be new (free) variables. We define the expression $(\vec{z} = v_i)$ to be the monomial that is 1 iff we set z_j to the jth bit of v_i. Formally

$$(\vec{z} = v_i) = (\bigwedge_{v_i[j]=1} z_j) \wedge (\bigwedge_{v_i[j]=0} \neg z_j).$$

Let $y_{10}, \ldots, y_{1m}$, $y_{20}, \ldots, y_{2m}$, $\ldots$, $y_{q0}, \ldots, y_{qm}$ be new variables.

Let

$$\phi' = \bigvee_{i=1}^{q} [\phi_i \wedge (\vec{z} = v_i) \wedge (\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab})].$$

Note that it is impossible for two disjuncts of ϕ' to be true at the same time. Hence

$$\#\mathrm{QBF}(\phi') = \sum_{i=1}^{q} \#\mathrm{QBF}(\phi_i \wedge (\vec{z} = v_i) \wedge (\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab})).$$

Note that ϕ_i, $(\vec{z} = v_i)$, and $\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab}$ use disjoint sets of variables. Also note that the $(q-i)(m+1)$ variables in $\{y_{ab} \mid i+1 \leq a \leq q, 0 \leq b \leq m\}$ are not constrained. Hence

$$\begin{aligned}\#\mathrm{QBF}(\phi_i \wedge (\vec{z} = v_i) \wedge (\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab})) = & \ \#\mathrm{QBF}(\phi_i) \cdot \#\mathrm{QBF}(\vec{z} = v_i) \cdot \\ & \ \#\mathrm{QBF}(\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab}); \\ = & \ \#\mathrm{QBF}(\phi_i) \cdot 2^{(q-i)(m+1)}.\end{aligned}$$

Putting this all together we obtain

$$\#\mathrm{QBF}(\phi') = \sum_{i=1}^{q} 2^{(q-i)(m+1)} \#\mathrm{QBF}(\phi_i).$$

Since $(\forall i)[\#\mathrm{QBF}(\phi_i) < 2^{m+1}]$ all the values $\#\mathrm{QBF}(\phi_i)$ can be easily deduced from $\#\mathrm{QBF}(\phi')$.

The formula ϕ' would be an ideal candidate for $T(\phi_1, \ldots, \phi_q)$ except that it is too long. We actually output a shorter formula that is equivalent to ϕ'. The idea is to introduce new variables $w_1, \ldots, w_q$ such that w_i will be equivalent to $\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab}$. The formula ϕ is the conjunction of the following three formulas

i. $w_1 = \bigwedge_{b=0}^{m} y_{1b}$.

ii. $\bigwedge_{i=2}^{q} [w_i = (w_{i-1} \wedge \bigwedge_{b=0}^{m} y_{ib})]$.

iii. $\bigvee_{i=1}^{q} [\phi_i \wedge (\vec{z} = v_i) \wedge w_i]$.

Clearly $\#\mathrm{QBF}(\phi) = \#\mathrm{QBF}(\phi')$. Note that

$$|\phi| = O(m) + O(mq) + \sum_{i=1}^{q} O(|\phi_i| + \log m) = O(|\langle \phi_1, \ldots, \phi_q \rangle|).$$

∎

Definition 7.12. Let A be a subset of quantified Boolean formulas. The set A is *nice* if the following hold.

i. All Boolean formulas (without quantifiers) are in A.

ii. If $\phi_1, \phi_2 \in A$ then $\phi_1 \vee \phi_2 \in A$ and $\phi_1 \wedge \phi_2 \in A$.

For a set of formulas to be nice we do *not* require that it be the minimal set of formulas that satisfy the conditions. For example note that the set of formulas that have at most i alternations of quantifiers is nice.

Lemma 7.13. *Let A be a nice set. Let f be the function* #QBF *restricted to A. For all polynomials q, $f_q^+ \leq_{\mathrm{m}}^{p} f$ via some reduction T_q where $(\forall z)[|T_q(z)| = O(|z|^{1+\frac{1}{deg(q)}})]$.*

Proof:

Fix a polynomial q. Let an input to f_q^+ be $\langle \phi_1, \ldots, \phi_{q(m)} \rangle$ where $(\forall i)[|\phi_i| \leq m]$. Assume that ϕ_i has $m_i \leq m$ variables. Let $\psi_i = \phi_i \wedge \bigwedge_{b=m_i+1}^{m} x_{ib}$ where the x_{ib} variables are new free variables. Note that for each i (1) ψ_i has exactly m variables, (2) $f(\phi_i) = f(\psi_i)$, and (3) $|\psi| = |\phi_i| + O(m)$.

The vector $\langle \psi_1, \ldots, \psi_{q(m)} \rangle$ is in the domain of the transformation T from Lemma 7.11. Let

$$T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle) = T(\langle \psi_1, \ldots, \psi_{q(m)} \rangle).$$

Since $T(\langle \psi_1, \ldots, \psi_{q(m)} \rangle)$ yields all the $f(\psi_i)$ and $f(\psi_i) = f(\phi_i)$, clearly $T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle)$ yields all the $f(\phi_i)$. Since A is nice, clearly $T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle) \in A$.

Note that

$$\begin{aligned}
|T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle)| &= |T(\langle \psi_1, \ldots, \psi_{q(m)} \rangle)|; \\
&= O(|\langle \psi_1, \ldots, \psi_{q(m)} \rangle|); \\
&= O(\textstyle\sum_{i=1}^{q(m)} |\psi_i|); \\
&= O(\textstyle\sum_{i=1}^{q(m)} |\phi_i| + m); \\
&= O(\textstyle\sum_{i=1}^{q(m)} 2m); \\
&= O(mq(m)); \\
&= O(q(m)^{1+\frac{1}{deg(q)}}).
\end{aligned}$$

Since the length of the input is $\Omega(q(m))$ and the length of the output is $O(q(m)^{1+\frac{1}{deg(q)}})$, we are done. ∎

Lemma 7.14. *Let f be #SAT, #QBF$_i$, or #QBF. For all polynomials q, $f_q^+ \leq_{\rm m}^{p} f$ via some reduction T_q where $(\forall z)[|T_q(z)| = O(|z|^{1+\frac{1}{deg(q)}})]$.*

7.3. Lemmas on #GA

Definition 7.15. If G is a graph then $Aut(G)$ is the group of automorphism of G. The set GA is the set of graphs G such that $|Aut(G)| \neq 1$. The function #GA takes as input a graph and outputs $|Aut(G)|$. The set GI is the set of pairs of graphs that are isomorphic. The function #GI takes as input a pair of graphs and outputs the number of isomorphisms between them. It is well known that GA, #GA, GI, and #GI are all polynomial time Turing equivalent (see [50]).

Lemma 7.16. *Let G be a graph on n vertices.*

i. If $i \leq n$ and $i^a \mid$ #GA(G) then $a \leq n$.

ii. Assume that, for all $i \leq n$ and $a \leq n$, it is known if $i^a \mid$ #GA(G). Then #GA(G) can be determined.

Proof:

Since $Aut(G)$ is a subgroup of S_n we have $\#\mathrm{GA}(G) \mid n!$. We use this in both parts.

$i)$ Since $i^a \mid \#\mathrm{GA}(G)$ we have $i^a \mid n!$. We bound a by counting the number of factors of $n!$ that have an i. The factors $i, 2i, 3i, \ldots, \lfloor \frac{n}{i} \rfloor$ contribute $\leq \frac{n}{i}$ factors (for now). Then the factors $i^2, 2i^2, \ldots, \lfloor \frac{n}{i^2} \rfloor$ contribute $\leq \frac{n}{i^2}$ more factors. continuing in this manner we see that the number of factors of i is at most $\frac{n}{i} + \frac{n}{i^2} + \cdots \leq n$.

$ii)$ Since $\#\mathrm{GA}(G) \mid n!$ all the prime factors of $\#\mathrm{GA}(G)$ are $\leq n$. We can easily identify which numbers $\leq n$ are primes (this is easy since the length is n). By i and the premise we know all the prime power factors of $\#\mathrm{GA}(G)$. From this we can easily deduce $\#\mathrm{GA}(G)$. ∎

Lemma 7.17. *Let G be a graph on at least two vertices. Let $m, n \in N$.*

i. We can find, in polynomial time, a connected graph G' such that $\#\mathrm{GA}(G') = \#\mathrm{GA}(G)$.

ii. If $|V(G)| < n$ then we can find, in time polynomial in n, a graph G' such that $|V(G')| = n$ and $\#\mathrm{GA}(G) = \#\mathrm{GA}(G')$.

iii. We can find, in time polynomial in n and m, m pairwise non-isomorphic graphs $G^1, \ldots, G^m$ such that $\#\mathrm{GA}(G) = \#\mathrm{GA}(G^1) = \#\mathrm{GA}(G^2) = \cdots = \#\mathrm{GA}(G^m)$.

Proof: We use the following gadgets for all three parts. If $H = (V, E)$ is a graph and $i \in \mathsf{N}$ then $tail_i(H)$ is formed by adding i vertices to H in a line and attaching every vertex of H to the same endpoint of that line. Formally

$$\begin{aligned} tail_i(G) = &\ (V', E') \\ V' = &\ V \cup \{a_1, \ldots, a_i\} \\ E' = &\ E \cup \{(v, a_1) \mid v \in V\} \cup \{(a_j, a_{j+1}) \mid 1 \leq j \leq i-1\} \end{aligned}$$

$i)$ Let $G' = tail_1(G)$.

$ii)$ Let $G' = tail_{n-|V(G)|}(G)$.

$iii)$ For $1 \leq i \leq m$ let $G^i = tail_i(G)$.

∎

Definition 7.18. Let $k(m)$ be a function. The function $\#\mathrm{GA}_k^+$ takes $k(m)$ graphs $(G_1, \ldots, G_{k(m)})$ with at most m vertices each as input, and returns $(\#GA(G_1), \ldots, \#\mathrm{GA}(G_k))$. This definition is a special case of f_k^+ but is stated overtly for clarity.

Lemma 7.19. *For all $k \in \mathsf{N}$, $\#\mathrm{GA}_k^+ \leq_{\mathrm{m}}^p \#\mathrm{GA}$.*

Proof: We show this for $k=2$. The general case will follow by induction.

Let G_1 and G_2 be graphs. By Lemma 7.17 (parts *i* and *ii*) we can assume they are connected, non-isomorphic, and have exactly n vertices. We will construct G such that from $\#\mathrm{GA}(G)$ one can deduce $\#\mathrm{GA}(G_1)$ and $\#\mathrm{GA}(G_2)$.

Let $m=n+1$. Let $G_1^1,\ldots,G_1^m$ be obtained by applying Lemma 7.17 to G_1 and m. Let

$$G = G_1^1 \cup G_1^2 \cup \cdots \cup G_1^m \cup G_2.$$

Note that $\#\mathrm{GA}(G) = (\#\mathrm{GA}(G_1))^m(\#\mathrm{GA}(G_2))$. We assume we know $\#\mathrm{GA}(G)$.

We determine, for each $i \le n$ and $a \le n$, if $x = i^a$ divides $\#\mathrm{GA}(G_j)$ ($j=1,2$). By Lemma 7.16 this suffices to determine $\#\mathrm{GA}(G_1)$ and $\#\mathrm{GA}(G_2)$. Given x we determine b such that x^b divides $\#\mathrm{GA}(G)$ but x^{b+1} does not. Divide b by m to determine c,d such that $b=cm+d$ where $0 \le d < m$. There are several cases.

i. If $c=d=0$ then x does not divide $\#\mathrm{GA}(G_1)$ or $\#\mathrm{GA}(G_2)$.

ii. If $c \ne 0$ and $d=0$ then x divides $\#\mathrm{GA}(G_1)$ but not $\#\mathrm{GA}(G_2)$. (To prove this we use Lemma 7.16 and the fact that if x^e is a factor of $\#\mathrm{GA}(G_1)$ then x^{em} is a factor of $\#\mathrm{GA}(G)$.)

iii. If $c=0$ and $d\ne 0$ then x does not divide $\#\mathrm{GA}(G_1)$ but x does divide $\#\mathrm{GA}(G_2)$. (This uses that $d<m$ and if $x \mid \#\mathrm{GA}(G_1)$ then $x^m \mid \#\mathrm{GA}(G)$.

iv. If $c\ne 0$ and $d \ne 0$ then x divides both $\#\mathrm{GA}(G_1)$ and $\#\mathrm{GA}(G_2)$.

∎

7.4. $b(n)$-enumerable for small $b(n)$

The main theorem of this section establishes conditions on a function g that cause the implication

$$g \;\; b(n)\text{-enumerable} \Rightarrow g \in \mathrm{PF}$$

for many sublinear functions $b(n)$.

Theorem 7.20. *Let b and g be functions and A be a set such that the following hold.*

i. There exists a polynomially bounded function q such that $F_q^A \le_{\mathrm{m}}^{p} g$ via a reduction T such that $(\forall\gamma)(\stackrel{\infty}{\forall} m)[|t| \le \gamma m q(m) \Rightarrow b(|T(t)|) \le q(m)]$. (We do not really need this for all γ, just for the particular one associated to the tupling function as indicated in the proof.)

ii. g is $b(n)$-enumerable.

iii. A is self-reducible.

Then $A \in \mathrm{P}$. (Note that in the case of $A = \mathrm{bit}_g$ the function g_q^+ can replace F_q^A in premise i, and the conclusion is $g \in \mathrm{PF}$.)

Proof: Let $k(m) = \log q(m)$. To show $A \in \mathrm{P}$ we will show that A is $k(m)$-cheatable and use Theorem 3.31.

To show that A is $k(m)$-cheatable we describe how to enumerate $\leq q(m)$ possibilities for $F_q^A(t)$, one of which must be correct, given $t = \langle x_1, \ldots, x_{q(m)} \rangle$ (where $(\forall i)[|x_i| \leq m]$).

Note that $|t| \leq \gamma m q(m)$ where γ is a constant depending on the tupling function.

First compute $z = T(t)$. Since g is $b(n)$-enumerable we can compute the $\leq b(|z|)$ possibilities for $g(z)$. Note that each possibility for $g(z)$ leads to a possibility for $F_q^A(t)$. The number of possibilities is $b(|z|) = b(|T(t)|)$. Since $|t| \leq \gamma m q(m)$, premise i of the hypothesis yields $(\overset{\infty}{\forall} m)[b(|T(t)|) \leq q(m)]$. Hence we can enumerate $\leq q(m)$ possibilities for $F_q^A(t)$. Since one of the possibilities enumerated for $g(z)$ was correct, one of the possibilities enumerated for $F_q^A(t)$ is correct. ∎

Corollary 7.21. *Let $\epsilon > 0$. Let $b(n) = n^\epsilon$. Let g be a function and A be a set such that the following hold.*

i. There exist α, d such that if $q(m) = m^\alpha$ then $F_q^A \leq_{\mathrm{m}}^{p} g$ via a reduction T with $|T(t)| \leq O(|t|^d)$. In addition $(\alpha+1)d\epsilon < \alpha$. (The hypothesis did not state $\epsilon < 1$; however, the condition $(\alpha+1)d\epsilon < \alpha$ virtually implies $\epsilon < 1$ in any application.)

ii. g is $b(n)$-enumerable.

iii. A is self-reducible.

Then $A \in \mathrm{P}$. (Note that in the case of $A = \mathrm{bit}_g$ the function g_q^+ can replace F_q^A in premise i, and the conclusion is $g \in \mathrm{PF}$.)

Proof: We show that the premise of Theorem 7.20 holds with $b(n) = n^\epsilon$, g, and A. This yields $A \in \mathrm{P}$. Clearly premises ii and iii hold. We show premise i.

Let $q(m) = m^\alpha$. We need $(\forall \gamma)(\overset{\infty}{\forall} m)[|t| \leq \gamma m q(m) \Rightarrow b(|T(t)|) \leq q(m)]$. Let γ be a constant and let t be of length $\leq \gamma m q(m) = \gamma m^{\alpha+1}$. Then

$$b(|T(t)|) = b(O(|t|^d) \leq O(|t|^{d\epsilon}) = O(m^{(\alpha+1)d\epsilon}).$$

Since $(\alpha+1)d\epsilon < \alpha$ we have $(\overset{\infty}{\forall} m)[b(|T(t)|) \leq m^\alpha = q(m)]$. ∎

Corollary 7.22. *Let $b(n) = k$, a constant. Let g be a function and A be a set such that the following hold.*

i. Let $q(m) = k + 1$. $F_q^A \leq_{\mathrm{m}}^{p} g$.

ii. g is $b(n)$-enumerable.

iii. A is self-reducible.

Then $A \in$ P. (Note that in the case of $A = \mathrm{bit}_g$ the function g_q^+ can replace F_q^A in premise i and the conclusion is $g \in$ PF.)

Proof: We show that the premise of Theorem 7.20 holds with $b(n) = k$, g, and A. This yields $A \in$ P. Clearly premises *ii* and *iii* hold. Since $(\forall t)[b(|T(t)|) = k \leq k + 1 = q(m)]$ we have premise *i*. ▮

Corollary 7.23. *Assume that* C *is closed under* $\leq_{\mathrm{m}}^p$*-reductions and that* C *has a self-reducible* $\leq_{\mathrm{m}}^p$*-complete set* A.

i. $\#$C *has a* $\leq_{\mathrm{m}}^p$*-complete function* g *such that*

$$(\forall \epsilon < \frac{1}{2})[g \ \ n^\epsilon\textit{-enumerable} \Rightarrow \mathrm{P} = \mathrm{P}^{\#\mathrm{C}}].$$

ii. If f *is* $\leq_{\mathrm{m}}^p$*-hard for* $\#$C *then*

$$(\exists \delta_0)(\forall \epsilon < \delta_0)[f \ \ n^\epsilon\textit{-enumerable} \Rightarrow \mathrm{P} = \mathrm{P}^{\#\mathrm{C}}].$$

Proof:
i. By Lemma 7.10 there exists a function g that is $\leq_{\mathrm{m}}^p$-complete for $\#$C such that bit_g is self-reducible. We will show that the premises of Corollary 7.21 (the version with $g_q^+ \leq_{\mathrm{m}}^p g$) are satisfied with $b(n) = n^\epsilon$, g, and $A = \mathrm{bit}_g$. This will yield $g \in$ PF; therefore, $\#\mathrm{C} \subseteq \mathrm{PF}$ and $\mathrm{P} = \mathrm{P}^{\#\mathrm{C}}$.

By Lemma 7.10, for all polynomials q of degree $\geq a$, $g_q^+ \leq_{\mathrm{m}}^p g$ via a reduction T such that $|T(t)| \leq O(|t|^{2+\frac{1}{deg(q)}})$.

We assume that g is n^ϵ-enumerable. Let $q(m) = m^\alpha$ where α will be specified later. We impose one condition on α now: $\alpha \geq a$. Hence we have $g_q^+ \leq_{\mathrm{m}}^p g$ via a reduction T such that $|T(t)| \leq O(|t|^{2+\frac{1}{\alpha}})$. To apply Corollary 7.21 we need $(\alpha+1)(2+\frac{1}{\alpha})\epsilon < \alpha$. Since $\epsilon < \frac{1}{2}$ there exists a large enough α to make this inequality true.

ii. Let g be from part (*i*). If f is $\leq_{\mathrm{m}}^p$-hard for $\#$C then there exists a polynomial time function T such that $g(t) = f(T(t))$. Let b be such that $|T(t)| \leq |t|^b$. Note that if f is $n^{\epsilon'}$-enumerable then g is $n^{b\epsilon'}$-enumerable.

Let $\delta = \frac{1}{2b}$. If there exists $\epsilon < \delta$ such that f is n^ϵ-enumerable then g is $n^{b\epsilon}$-enumerable. Since $b\epsilon < 1/2$, by part (*i*) we have $\mathrm{P} = \mathrm{P}^{\#\mathrm{C}}$. ▮

We can obtain corollaries about the functions in Example 7.7.*iv* by using Lemma 7.10 (and using that for all $Q \in$ P the class QP has a self-reducible complete function[58, Lemma 3.1]). We can also obtain corollaries about $\#$SAT, $\#\mathrm{QBF}_i$, and $\#$QBF by using Lemma 7.14 (and using that $\mathrm{bit}_{\#\mathrm{SAT}}$ etc. are self-reducible). However, for these functions, Corollary 7.32 yields a better result.

We have the following corollary.

Corollary 7.24. *Let* $i, j \geq 1$ *and* $\mathrm{C} = \Sigma_i\text{-SPACE}(\log^j n)$*. If* f *is* $\leq_{\mathrm{m}}^{p}$*-hard for* $\#\mathrm{C}$ *then*

$$(\exists \delta \leq 1)(\forall \epsilon < \delta)[f \;\; n^{\epsilon}\text{-}enumerable \Rightarrow \mathrm{P} = \mathrm{P}^{\#\mathrm{C}}].$$

Proof: Clearly C is closed under $\leq_{\mathrm{m}}^{p}$-reductions and C has a self-reducible $\leq_{\mathrm{m}}^{p}$-complete set A. Hence this follows from Corollary 7.23. ∎

We now look at #GA. Clearly #GA is $n!$-enumerable (where n is the number of vertices). Lazlo Babai [7] has shown that #GA is $2^{\frac{n}{2}}(n/2)!$-enumerable. We would like to obtain matching upper and lower bounds. We are unable to show this but we can show the following.

Corollary 7.25. *Let* $k \in N$*. If* #GA *is* k*-enumerable then* $\#\mathrm{GA} \in \mathrm{P}$ *(and hence* $\mathrm{GI} \in \mathrm{P}$*).*

Proof: Let g be #GA. One can show bit_g is self-reducible by the techniques used to show show GI is self-reducible (see [50]). By Lemma 7.19 $g_{k+1}^{+} \leq_{\mathrm{m}}^{p} g$. Hence $F_{k+1}^{\mathrm{bit}_g} \leq_{\mathrm{m}}^{p} g$. By Corollary 7.22 $\#\mathrm{GA} \in \mathrm{PF}$. ∎

Chang, Gasarch, and Torán [28] have used Corollary 7.21 to show that if #GA is n^{ϵ}-enumerable (any $\epsilon < 1$) then $\mathrm{GI} \in \mathrm{P}$. They have also obtained (1) if #GA is poly-enum the $\mathrm{GI} \in \mathrm{R}$, and (2) if #GA is $2^{n^{\epsilon}}$-enumerable ($\epsilon < \frac{1}{6}$) then GI is in $\mathrm{NP}[|V|^{6\epsilon} \log |V|]/\mathrm{poly}$. (A set of graphs is in $\mathrm{NP}[f(|V|)]$ if it is in NP via a machine that makes at most $f(|V|)$ nondetermisitic moves. The concept is from [47] and the notation is from [59], though they deal with sets of strings and lengths as opposed to sets of graphs and $|V|$.) These two theorems use ideas from the proof that $\overline{\mathrm{GI}} \in \mathrm{AM}[2]$ ([37], also see [50, Corollary 2.10]).

7.5. $b(n)$-enumerable for large $b(n)$

The main theorem of this section establishes conditions on a function g and a real $\epsilon < 1$ that cause the implication

$$g \;\; 2^{n^{\epsilon}}\text{-enumerable} \Rightarrow \mathrm{P}^{g} \subseteq \Sigma_4^p \cap \Pi_4^p.$$

We then apply this theorem, with the help of Lemmas 7.10 and 7.14, to obtain several examples of such functions. In addition we show that if any #P-hard function is n^k-enumerable then $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$.

Theorem 7.26. *Let* b *and* g *be functions and* A *be a set such that the following hold.*

i. There exists a polynomially bounded function q *such that* $F_q^A \leq_{\mathrm{m}}^{p} g$ *via a reduction* T *such that* $(\forall \gamma)(\overset{\infty}{\forall} m)[|t| \leq \gamma m q(m) \Rightarrow b(|T(t)|) \leq 2^{q(m)} - 1]$*. (We do not really need this for all* γ*, just for the particular one associated to the tupling function as indicated in the proof.)*

ii. g is $b(n)$-enumerable. ($b(n)$ need not be bounded by a polynomial so we use the definition of enumerable stated in Definition 2.8.)

iii. A is self-reducible.

Then the following occur.

a) $A \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$ and $\mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$.

b) If $A = \mathrm{bit}_g$ then $\mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p$. In this case the function g_q^+ can replace F_q^A in premise i.

c) If the function q in premise i is such that $q(m) = O(\log m)$ then the conclusion can be strengthened to $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$ and $\mathrm{P}^A \subseteq \Sigma_3^p \cap \Pi_3^p$. If in addition $A = \mathrm{bit}_g$ then $\mathrm{P}^g \subseteq \Sigma_3^p \cap \Pi_3^p$. In this case the function g_q^+ can replace F_q^A in premise i.

Proof:

a) We will find a set $W \in \Pi_1^p$ such that A satisfies the premise of Theorem 4.4 and Corollary 4.6.*c*. with W and q. By Theorem 4.4 we will have

$$A \in \mathrm{NP}^W/\mathrm{poly} \cap \mathrm{co\text{-}NP}^W/\mathrm{poly} \subseteq \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}.$$

By Corollary 4.6 we will have $\mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$.

To define W we will need a function e. We state what properties e will need, define W, and then show that such an e exists.

The function e will have the following properties.

i. $e \in \mathrm{PF}$.

ii. $e : \cup_{m=0}^{\infty} (\Sigma^m)^{q(m)} \times \mathsf{N} \to \Sigma^*$.

iii. $(\overset{\infty}{\forall} m)(\forall t \in (\Sigma^m)^{q(m)})(\exists i \leq 2^{q(m)} - 2)[e(t,i) = F_{q(m)}^A(t)]$.

Once we have e we can define W as follows. Let W be the union over m of the set of ordered pairs $(t, \vec{c})$ where the following hold.

i. $t \in (\Sigma^m)^{q(m)}$ and $\vec{c} \in \{0,1\}^m$.

ii. $(\forall i \leq 2^{q(m)} - 2)[\vec{c} \neq e(t,i)]$. ($W$ will be nontrivial since this condition only eliminates $\leq 2^{q(m)} - 1$ choices for $\vec{c}$, namely $e(t,0), \ldots, e(t, 2^{q(m)} - 2)$.)

It is easy to see that A satisfies the premise of Theorem 4.4 with this choice of W and q and that $W \in \Pi_1^p$.

It remains to define e. Since g is $b(n)$-enumerable there exists a function $e' \in \mathrm{PF}$, $e' : \Sigma^* \times \mathsf{N} \to \Sigma^*$, that $b(n)$-enumerates g (in the sense of Definition 2.8). We will use e' later.

Let $m \in \mathsf{N}$, and

$$t = \langle x_1, \ldots, x_{q(m)} \rangle$$

(where $(\forall i)[|x_i| = m]$). Note that $|t| \le \gamma m q(m)$ where γ is a constant depending on the tupling function. Let $i \in \mathsf{N}$. We describe how to compute $e(t,i)$.

First compute $z = T(t)$. Every possibility for $g(z)$ yields a possibility for $F_q^A(t)$. Let $POSS \in \mathrm{PF}$ map possibilities for $g(z)$ to the corresponding possibilities for $F_q^A(t)$. Let $e(t,i) = POSS(e'(z,i))$. Clearly $e \in \mathrm{PF}$.

By the definition of e' we know that $(\exists i \le b(|z|) - 1)[e'(z,i) = g(z)]$, hence $(\exists i \le b(|T(t)|))[e(t,i) = F^A_{q(m)}(t)]$. Since $|t| \le \gamma m q(m)$ premise i of the hypothesis yields $(\overset{\infty}{\forall} m)[b(|T(t)|) - 1 \le 2^{q(m)} - 2]$. Hence stipulation iii on the function e is met.

b) $\mathrm{P}^g \subseteq \mathrm{P}^{\mathrm{bit}_g} = \mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$.

c) If $q(m) = O(\log m)$ then $W \in \mathrm{P}$ hence $A \in \mathrm{NP}^W/\mathrm{poly} \cap \text{co-}\mathrm{NP}^W/\mathrm{poly} = \mathrm{NP}/\mathrm{poly} \cap \text{co-}\mathrm{NP}/\mathrm{poly}$. Since A is self-reducible, by Corollary 4.6 $\mathrm{P}^A \subseteq \Sigma_3^p \cap \Pi_3^p$. If $A = \mathrm{bit}_g$ then $\mathrm{P}^g \subseteq \mathrm{P}^{\mathrm{bit}_g} = \mathrm{P}^A \subseteq \Sigma_3^p \cap \Pi_3^p$.

∎

Corollary 7.27. *Let $\epsilon > 0$. Let $b(n) = 2^{n^\epsilon}$. Let g be a function and A be a set such that the following hold.*

i. There exists α, d such that if $q(m) = m^\alpha$ then $F_q^A \le_{\mathrm{m}}^p g$ via a reduction T with $|T(t)| \le O(|t|^d)$. In addition $(\alpha+1)d\epsilon < \alpha$. (The hypothesis did not state $\epsilon < 1$; however, the condition $(\alpha+1)d\epsilon < \alpha$ virtually implies $\epsilon < 1$ in any application.)

ii. g is $b(n)$-enumerable.

iii. A is self-reducible.

Then $A \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$ and $\mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$. If $A = \mathrm{bit}_g$ then $\mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p$. (In this case the function g_q^+ can replace F_q^A in premise i and the conclusion is $g \in \mathrm{PF}$.)

Proof: We show that the premises of Theorem 7.26 hold with $b(n) = 2^{n^\epsilon}$, g, and A. Clearly premises *ii* and *iii* hold. We show premise *i*.

Let $q(m) = m^\alpha$. We need $(\forall \gamma)(\overset{\infty}{\forall} m)[|t| \le \gamma m q(m) \Rightarrow b(|T(t)|) \le 2^{q(m)} - 1]$. Let γ be a constant and let t be of length $\le \gamma m q(m) = \gamma m^{\alpha+1}$. Then

$$b(|T(t)|) = b(O(|t|^d) \le 2^{O(|t|^{d\epsilon})} \le 2^{O(m^{(\alpha+1)d\epsilon})}$$

Since $(\alpha+1)d\epsilon < \alpha$ we have $(\overset{\infty}{\forall} m)[b(|T(t)|) \le 2^{m^\alpha} - 1 = 2^{q(m)} - 1]$. ∎

Corollary 7.28. *Assume that C is closed under $\le_{\mathrm{m}}^p$-reductions and that C has a self-reducible $\le_{\mathrm{m}}^p$-complete set A.*

i. $\#\mathrm{C}$ *has a* $\leq_{\mathrm{m}}^{p}$*-complete function* g *such that*

$$(\forall \epsilon < \frac{1}{2})[g \;\; 2^{n^{\epsilon}}\text{-}enumerable \Rightarrow \mathrm{P}^{\#\mathrm{C}} \subseteq \Sigma_4^p \cap \Pi_4^p].$$

ii. If f *is* $\leq_{\mathrm{m}}^{p}$*-hard for* $\#\mathrm{C}$ *then*

$$(\exists \delta \leq 1)(\forall \epsilon < \delta)[f \;\; 2^{n^{\epsilon}}\text{-}enumerable \Rightarrow \mathrm{P}^{\#\mathrm{C}} \subseteq \Sigma_4^p \cap \Pi_4^p].$$

Proof:
i. By Lemma 7.10 there exists a function g that is $\leq_{\mathrm{m}}^{p}$-complete for $\#\mathrm{C}$ such that bit_g is self-reducible. We will show that the premise of Corollary 7.27 (the version stated in conclusion b with g_q^+ instead of F_q^A) is satisfied with $b(n) = 2^{n^{\epsilon}}$, g, and $A = \mathrm{bit}_g$. This will yield $\mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p$ and hence $\mathrm{P}^{\#\mathrm{C}} \subseteq \Sigma_4^p \cap \Pi_4^p$.

We assume that g is $2^{n^{\epsilon}}$-enumerable. Let $q(m) = m^{\alpha}$ where α will be specified later. We impose one condition on α now: $\alpha \geq a$. By Lemma 7.10, $g_q^+ \leq_{\mathrm{m}}^{p} g$ via a reduction T such that $|T(t)| \leq O(|t|^{2+\frac{1}{deg(q)}})$. To apply Corollary 7.27 we need $(\alpha+1)(2+\frac{1}{\alpha})\epsilon < \alpha$. Since $\epsilon < \frac{1}{2}$ there exists a large enough α to make this inequality true.

ii. Let g be from part (*i*). If f is $\leq_{\mathrm{m}}^{p}$-hard for $\#\mathrm{C}$ then there exists a polynomial time function T such that $g(x) = f(T(x))$. Let b be such that $|T(x)| \leq |x|^b$. Note that if f is $2^{n^{\epsilon'}}$-enumerable then g is $2^{n^{b\epsilon'}}$-enumerable.

Let $\delta = \frac{1}{2b}$. If there exists $\epsilon < \delta$ such that f is $2^{n^{\epsilon}}$-enumerable then g is $2^{n^{b\epsilon}}$-enumerable. Since $b\epsilon < 1/2$, by part (*i*) we have that $\mathrm{P}^{\#\mathrm{C}} \subseteq \Sigma_4^p \cap \Pi_4^p$. ∎

Corollary 7.29. *Let* $Q \in \mathrm{P}$ *such that* $\mathrm{P} \subseteq \mathrm{QP}$ *(as defined in Example 7.7). If* f *is* $\leq_{\mathrm{m}}^{p}$*-hard for* $\#\mathrm{QP}$ *then*

$$(\exists \delta \leq 1)(\forall \epsilon < \delta)[f \;\; 2^{n^{\epsilon}}\text{-}enumerable \Rightarrow \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{QP}} \subseteq \Sigma_4^p \cap \Pi_4^p].$$

Proof: By [58, Lemma 3.1] the class QP has a self-reducible complete set, hence QP and f satisfies the premise of Corollary 7.28.*ii*.

If $\mathrm{P}^{\#\mathrm{QP}} \subseteq \Sigma_4^p \cap \Pi_4^p$ then, by Toda's theorem [70] that $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$, we have

$$\mathrm{P}^{\#\mathrm{QP}} \subseteq \Sigma_4^p \cap \Pi_4^p \Rightarrow \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{QP}} \subseteq \Sigma_4^p \cap \Pi_4^p.$$

∎

The next corollary states that for some problems we can obtain better results.

Corollary 7.30. *Let* ϵ *be any real such that* $0 \leq \epsilon < 1$. *Let* $i \geq 1$.

i. If $\#\mathrm{SAT}$ *is* $2^{n^{\epsilon}}$*-enumerable then* $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

ii. If $\#\mathrm{QBF}_i$ *is* $2^{n^{\epsilon}}$*-enumerable then* $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

iii. If #QBF *is* 2^{n^ϵ}*-enumerable then* $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{PSPACE}}$.

Proof: We prove (i). The other proofs are similar.

Let $g = \#\mathrm{SAT}$. We assume that g is 2^{n^ϵ}-enumerable. We will show that the premise of Theorem 7.26 (the version stated in conclusion b with g_q^+ instead of F_q^A) is satisfied with $b(n) = 2^{n^\epsilon}$, g, and $A = \mathrm{bit}_g$. This will yield $\mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p$ and hence $\mathrm{P}^{\#\mathrm{P}} \subseteq \Sigma_4^p \cap \Pi_4^p$; by [70] $\Sigma_4^p \cap \Pi_4^p \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$, hence $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

Clearly premises ii and iii are satisfied. We show that premise i is satisfied. Let $q(m) = m^\alpha$ where α will be specified later. By Lemma 7.14 $g_q^+ \leq_{\mathrm{m}}^p g$ via a reduction T such that $(\forall z)[|T(z)| \leq O(|z|^{1+\frac{1}{\alpha}})]$. In order to apply Theorem 7.26 we need that $(\alpha+1)(1+\frac{1}{\alpha})\epsilon < \alpha$. Since $\epsilon < 1$ there exists a large enough α to make this inequality true. ∎

We now look at PERM01 carefully.

Corollary 7.31.

i. PERM01 *is* $2^{n^{0.5+o(1)}}$*-enumerable.*

ii. $(\forall \epsilon < 0.5)$[PERM01 2^{n^ϵ}*-enumerable* $\Rightarrow \Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$].

Proof:

i. Note that an $m \times m$ 0-1 matrix has size $\Theta(m^2)$ and permanent at most $m!$. If the input is an $m \times m$ 0-1 matrix of size $n = \Theta(m^2)$ then one can easily enumerate (in the sense of Definition 2.8) as choices $\{0, 1, \ldots, m!\}$. Clearly $m! = O(m^m) = O((\sqrt{n})^{\sqrt{n}}) = 2^{n^{0.5+o(1)}}$.

ii. Let $\epsilon < 0.5$. The reduction of #SAT to PERM01 in [72] has a square blowup. Hence, if PERM01 was 2^{n^ϵ}-enumerable then #SAT would be $2^{n^{2\epsilon}}$-enumerable. Since $2\epsilon < 1$ this implies, by Corollary 7.30.i, that $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$. ∎

Corollaries 7.30.i and 7.31 offer an interesting contrast. Both #SAT and PERM01 are #P-complete; however, PERM01 is $2^{n^{0.5+o(1)}}$-enumerable and (assuming $\Sigma_4^p \cap \Pi_4^p \subset \mathrm{P}^{\#\mathrm{P}}$) #SAT is not 2^{n^ϵ}-enumerable for any $\epsilon < 1$.

The following corollary has been obtained independently by Cai and Hemachandra [26].

Corollary 7.32. *Let f be any function that is $\leq_{\mathrm{m}}^p$-hard for* #P*. If f is n^k-enumerable then* $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$.

Proof:

If f is n^k enumerable then for all $\epsilon > 0$ f is 2^{n^ϵ}-enumerable. Hence, by Corollary 7.29 $\mathrm{P}^{\#\mathrm{P}} = \Sigma_4^p \cap \Pi_4^p$.

We show that $\mathrm{P} = \mathrm{NP}$ which will imply $\mathrm{P} = \Sigma_4^p$. Combining that with the above yields $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$.

Let MAXSAT be the following problem: given a formula in CNF-form, find the truth assignment that maximizes the number of clauses that are satisfied. Krentel [51] showed that MAXSAT $\in \mathrm{PF}^{\mathrm{SAT}}$ via binary search techniques. Toda and Watanabe [71, Theorem 4.1] showed that $\mathrm{PF}^{\mathrm{PH}} \subseteq \mathrm{PF}^{\#\mathrm{P}[1]}$, hence MAXSAT $\in \mathrm{PF}^{\#\mathrm{P}[1]}$. Since f is $\leq_{\mathrm{m}}^{p}$-hard for #P and f is n^k-enumerable we know MAXSAT is n^k-enumerable. We use this to show SAT $\in$ P.

Given a formula ϕ we enumerate n^k possible values of MAXSAT(ϕ). Since there are only n^k of them we can plug each one into ϕ. If any of them satisfy ϕ then $\phi \in$ SAT, otherwise $\phi \notin$ SAT. ∎

All the theorems and corollaries easily relativize (using a relativized version of Theorem 4.4). From these relativized results we obtain the following corollaries.

Definition 7.33. Let $b(n)$ be a function with range N. Let $\mathcal{A} \subseteq 2^{\Sigma^*}$ and f be a function. The function f is $(b(n), \mathcal{A})$-*enumerable* if there exists $A \in \mathcal{A}$ and $e \in \mathrm{PF}^A$ such that $e : \Sigma^* \times \mathsf{N} \rightarrow \Sigma^*$ and $(\forall x)(\exists i < b(|x|)[e(x,i) = f(x)]$. (We need to have $i < b(|x|)$ instead of $i \leq b(|x|)$ since the natural numbers N include 0. We assume the second input to e is written in binary.)

Corollary 7.34. *Let $\mathcal{A} \subseteq 2^{\Sigma^*}$. Let ϵ be any real such that $0 \leq \epsilon < 1$. Let $i \geq 1$.*

i. Let Q $\in$ P and let C = QP (as defined in Example 7.7). If f is $\leq_{\mathrm{m}}^{p}$-hard for #C then

$$(\forall \epsilon < \frac{1}{2})[f \ \ (2^{n^\epsilon}, \mathcal{A})\text{-}enumerable \Rightarrow \mathrm{P}^{\#\mathrm{C}} \subseteq \Sigma_4^{p,\mathcal{A}} \cap \Pi_4^{p,\mathcal{A}}].$$

ii. Let f be any function that is $\leq_{\mathrm{m}}^{p}$-hard for #P. If f is $(n^k, \mathcal{A})$-enumerable then $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}^{A}$.

Corollary 7.35. *Let $i, j \geq 1$. Let ϵ be any real such that $0 \leq \epsilon < 1$.*

i. Let Q $\in$ P and let C = QP (as defined in Example 7.7). If f is $\leq_{\mathrm{m}}^{p}$-hard for #C then

$$(\exists \delta \leq 1)(\forall \epsilon < \delta)[f \ \ (2^{n^\epsilon}, \Sigma_j^p)\text{-}enumerable \Rightarrow \mathrm{P}^{\#\mathrm{C}} \subseteq \Sigma_{j+4}^p \cap \Pi_{j+4}^p].$$

ii. If #SAT is $(2^{n^\epsilon}, \Sigma_j^p)$-enumerable then $\Sigma_{j+4}^p \cap \Pi_{j+4}^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

iii. If $\#\mathrm{QBF}_i$ is $(2^{n^\epsilon}, \Sigma_j^p)$-enumerable then $\Sigma_{j+4}^p \cap \Pi_{j+4}^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

iv. If #QBF is $(2^{n^\epsilon}, \Sigma_j^p)$-enumerable then $\Sigma_{j+4}^p \cap \Pi_{j+4}^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{PSPACE}}$.

v. Let f be any function that is $\leq_{\mathrm{m}}^{p}$-hard for #P. If f is (n^k, Σ_j^p)-enumerable then $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}^{\Sigma_j^p}$.

8. Structural Properties

In this section we examine the *classes* of sets A that have properties based on how easy it is to compute F_k^A. Some results about which sets can be cheatable (or have other properties) fall out of our work, notably Corollaries 8.8 and 8.15.

In Section 8.1 we exhibit closure and nonclosure properties of cheatable sets. We show the following.

- If A is i-cheatable and B is j-cheatable then $A \cup B$, $A \cap B$, and $A \oplus B$ are $(i+j)$-cheatable.
- There exists sets A and B such that A is i-cheatable, B is j-cheatable, but $A \cap B$, $A \oplus B$, and $A \cup B$ are not $(i+j-1)$-cheatable (i.e., the previous result is tight).

It is of interest that Turing reductions preserve k-cheatability exactly (by Lemma 3.22), but union, intersection, and join cause a loss of some "cheatability."

In Section 8.2 we compare p-selective sets to cheatable sets. Both classes contain P. Both classes are contained in P/poly (see Ko [48] for p-selective$\subseteq$ P/poly and see Theorem 3.4 of this paper for cheatable$\subseteq$ P/poly), and the proofs are similar. Hence it is natural to ask if either class is contained in the other. We show that this is not the case by showing that all possible combinations of these two properties are possible.

It is easy to construct a p-superterse set by diagonalization. The p-generic sets were defined [5] so that (informally) any property that can be enforced by diagonalization is true for a p-generic set. Hence all p-generic sets are p-superterse. In Section 8.3 we explore the ramifications of this, which include three absolute results about which sets cannot be cheatable or p-superterse.

- There exist p-superterse sets in EXPTIME (Corollary 8.7).
- If A is cheatable then A is not $\leq_{\mathrm{T}}^{p}$-hard for EXPTIME (Corollary 8.8).
- If A is not p-superterse then A is not $\leq_{\mathrm{btt}}^{p}$-hard for EXPTIME. (Corollary 8.8).

In Section 8.4 we examine the polynomial degrees that contain p-superterse sets (called p-superterse degrees). From this work we also show, assuming P $\neq$ NP, that certain sets are p-superterse. We show the following.

- Every polynomial Turing degree contains either a p-superterse set or a cheatable set, but not both (Corollary 8.12).
- If P $\neq$ NP then all $\leq_{k\text{-tt}}^{p}$-hard sets for the Boolean hierarchy are p-superterse (Corollary 8.15). (This has been superseded by [17, 57], who have shown that if P $\neq$ NP then SAT is p-superterse.)

We also study the structure of the p-superterse degrees. We show the following.

- Let **a** and **b** be any two recursive $\leq_{\mathrm{T}}^{p}$-degrees such that $\mathbf{a}\leq_{\mathrm{T}}^{p}\mathbf{b}$ and **b** is not p-superterse. Let L be any countable distributive lattice. L can be embedded into the p-superterse $\leq_{\mathrm{T}}^{p}$-degrees on the interval between **a** and **b**. This also holds for $\leq_{\mathrm{tt}}^{p}$, $\leq_{\mathrm{btt}}^{p}$, and $\leq_{\mathrm{m}}^{p}$. (Theorem 8.22).

- If $\mathrm{P} \neq \mathrm{NP}$ and L is any countable distributive lattice then L can be embedded into the p-superterse NP $\leq_{\mathrm{T}}^{p}$-degrees. This also holds for $\leq_{\mathrm{tt}}^{p}$, $\leq_{\mathrm{btt}}^{p}$, and $\leq_{\mathrm{m}}^{p}$. (Corollary 8.23).

8.1. Closure Properties for Cheatable Sets

By Lemma 3.22 the class of cheatable sets is closed under $\leq_{\mathrm{T}}^{p}$-reductions. We show that the class of cheatable sets is also closed under union, intersection, and join.

Theorem 8.1. *If A is i-cheatable and B is j-cheatable then $A\cap B$, $A\cup B$, and $A\oplus B$ are $(i+j)$-cheatable.*

Proof: Since A is i-cheatable and B is j-cheatable there exist sets X and Y such that for all m, $F_m^A \in \mathrm{PF}_{i\text{-}\mathrm{T}}^{X}$ and $F_m^B \in \mathrm{PF}_{j\text{-}\mathrm{T}}^{Y}$. Hence $F_m^{A\oplus B}$ is in $\mathrm{PF}_{(i+j)\text{-}\mathrm{T}}^{X\oplus Y}$, so $A \oplus B$ is $(i+j)$-cheatable. Since $A\cup B \leq_{\mathrm{T}}^{p} A\oplus B$ and $A \cap B \leq_{\mathrm{T}}^{p} A \oplus B$, it follows from Lemma 3.22 that $A\cup B$ and $A\cap B$ are $(i+j)$-cheatable. ∎

The next theorem shows that the previous result is the best possible.

Theorem 8.2. *There exist sets A and B such that A is i-cheatable and B is j-cheatable, but $A\cap B$ and $A \oplus B$ are not $(i+j-1)$-cheatable. (In fact, both are $(i+j)$-p-superterse.)*

Proof: We construct A and B such that A is i-cheatable, B is j-cheatable, $A\cap B$ is $(i+j)$-p-superterse, and $A\cap B \leq_{m}^{p} A\oplus B$, hence $A\oplus B$ is also $(i+j)$-p-superterse. Let TOW be defined by $TOW(0)=1$, $TOW(n+1)=2^{TOW(n)}$. We construct sets A and B such that

$$\bigcup_{n\geq n_0}\{0^{TOW(n)+k} : i+1\leq k\leq i+j\} \subseteq A \subseteq \textstyle\bigcup_{n\geq n_0}\{0^{TOW(n)+k} : 1\leq k\leq i+j\},$$
$$\bigcup_{n\geq n_0}\{0^{TOW(n)+k} : 1\leq k\leq i\} \subseteq B \subseteq \textstyle\bigcup_{n\geq n_0}\{0^{TOW(n)+k} : 1\leq k\leq i+j\},$$

where n_0 is a constant such that $TOW(n_0+1)-TOW(n_0)\geq i+j$.

Let $M_0^{()}, M_1^{()}, \ldots$ be an effective list of all polynomial time bounded oracle Turing machines that make at most $i+j-1$ queries. Without loss of generality, assume that M_e^X runs in time bounded by $2^{|x|}$ for every input x and oracle X.

We use the strings in $\{0^{TOW(e)+k} : 1\leq k\leq i+j\}$ to diagonalize against $M_e^{()}$. We call this set of strings the e-th *block*.

ALGORITHM FOR A

Step 1: Input(0^n).

Step 2: Find $e \geq n_0$ and k such that $n = TOW(e) + k$, $1 \leq k \leq i + j$. If none exists then output NO and halt. If $i + 1 \leq k \leq i + j$ then output YES and halt.

Step 3: Run $M_e^{()}(0^{TOW(e)+1}, \ldots, 0^{TOW(e)+i+j})$ pursuing all paths. Since there are only 2^{i+j-1} possible paths there exists a string of $i+j$ bits that is not output by any path of the $M_e^{()}(0^{TOW(e)+1}, \ldots, 0^{TOW(e)+i+j})$ computation. Let $b_1 \cdots b_{i+j}$ be the least such string.

Step 4: If $b_k = 1$ then output YES, else output NO.

END OF ALGORITHM.

The algorithm for B is similar.

We show that for any X, M_e^X does not compute $F_{i+j}^{A \cap B}$. Since the first i strings of any block are in B, the membership of those strings in $A \cap B$ is entirely determined by their membership in A. Since the last j strings of any block are in A the membership of those elements in $A \cap B$ is entirely determined by their membership in B. Hence

$$F_{i+j}^{A \cap B}(0^{TOW(e)+1}, \ldots, 0^{TOW(e)+i+j}) = b_1 \cdots b_{i+j}$$

where $b_1 \cdots b_{i+j}$ is the least string that is not a possible answer for $M_e^{()}(0^{TOW(e)+1}, \ldots, 0^{TOW(e)+i+j})$. Thus for any X,

$$F_{i+j}^{A \cap B}(0^{TOW(e)+1}, \ldots, 0^{TOW(e)+i+j}) = b_1 \cdots b_{i+j} \neq M_e^X(0^{TOW(e)+1}, \ldots, 0^{TOW(e)+i+j}).$$

In the algorithm for A, steps 1, 2, and 4 run in polynomial time, and step 3 runs in time 2^n. Hence $A \in \mathrm{DTIME}(2^{O(n)})$ (the constant depends on details of the machine model). We will show that A is i-cheatable by showing that $F_{i+1}^A \in \mathrm{PF}_{i\text{-tt}}^A$, which implies that for all m, $F_m^A \in \mathrm{PF}_{i\text{-tt}}^A$. Suppose that we are to compute $F_{i+1}^A(x_1, \ldots, x_{i+1})$. Check if there exists an x_p that is not in a block. If this is the case then $x_p \notin A$, and i queries suffice to find $F_{i+1}^A(x_1, \ldots, x_{i+1})$. If this does not happen then check if there exists an x_p that is among the last j elements of a block. If this is the case then $x_p \in A$, and i queries suffice to find $F_{i+1}^A(x_1, \ldots, x_{i+1})$. If neither of these two occurs then we can assume $|x_1| < |x_2| < \cdots < |x_{i+1}|$ and that none of the elements are the last j elements of a block. Since there are only $i+j$ elements in a block, $|x_1|$ must be in a different block than x_{i+1}. This implies that $|x_1| < \log |x_{i+1}|$. Hence we can determine, by the $\mathrm{DTIME}(2^{O(n)})$ algorithm for A, the value of $A(x_1)$ in time polynomial in $|x_{i+1}|$, which is less than the length of $(x_1, \ldots, x_{i+1})$. Now i queries suffice to find $F_{i+1}^A(x_1, \ldots, x_{i+1})$.

The proof that B is j-cheatable is similar.

$A \cap B \leq_m^p A \oplus B$ by the following reduction:

$$f(0^n) = \begin{cases} 10^n & \text{if } n = TOW(e) + k,\ e \geq n_0, \text{ and } 1 \leq k \leq i \\ 00^n & \text{if } n = TOW(e) + k,\ e \geq n_0, \text{ and } i+1 \leq k \leq i+j \\ x_0 & \text{otherwise} \end{cases}$$

where x_0 is a fixed string such that $x_0 \notin A \oplus B$. ∎

Note: Let $C = \overline{A}$ and $D = \overline{B}$ where A and B are the sets constructed in Theorem 8.2. Since k-cheatability is preserved under complement, C is i-cheatable, D is j-cheatable and $C \cup D$ is not $(i+j-1)$-cheatable. Thus a version of Theorem 8.2 for unions holds as well.

8.2. P-Selective Sets

In this section we compare p-selective sets (see Definition 3.11) 1-cheatable sets, and non-cheatable sets.

Theorem 8.3. *The following exist.*

i. 1-cheatable sets that are p-selective but not in P*.*

ii. 1-cheatable sets that are not p-selective.

iii. non-cheatable sets that are p-selective.

iv. non-cheatable sets that are not p-selective.

Proof: In [6], we constructed tally sets that are 1-cheatable but not in P. Let T_1 be such a set. It is easy to construct tally sets that are not cheatable (*e.g.*, nonrecursive tally sets [15]). Let T_2 be such a set. In [66], Selman shows that given any tally set $T \notin \mathrm{P}$ there are sets $A, B \equiv^p_{\mathrm{T}} T$ such that A is p-selective and B is not p-selective. (Let A be the set of strings lexicographically preceding the characteristic sequence of T. Let $B = T \oplus \overline{T}$.) For $i = 1, 2$ let A_i and B_i be such that $A_i, B_i \equiv^p_T T_i$, A_i is p-selective and B_i is not p-selective. Since k-cheatability is preserved by polynomial-time Turing equivalence A_1 is 1-cheatable and p-selective, B_1 is 1-cheatable and not p-selective, A_2 is non-cheatable and p-selective, and B_2 is non-cheatable and not p-selective. ∎

8.3. P-Generic Sets

In [5] Ambos-Spies *et al.* defined p-generic sets, in a manner similar to the 1-generic sets in recursion theory [45]. Intuitively a p-generic set has any property that a recursive set can be constructed to have via easy diagonalization. The following definition of p-generic is not standard but is equivalent to the standard one.

Definition 8.4. If A is a tally set then the *characteristic string* of A is the infinite string of 0's and 1's that has 1 in the n-th place iff $0^n \in A$. We denote the first s bits of the characteristic string of A by $A[s]$.

Definition 8.5. A tally set A is *p-generic* if for every set $C \in$P,

$$(\exists c)(\forall s)(\exists \tau \in \{0,1\}^c)[A[s]\tau \in C] \Rightarrow (\exists s)[A[s] \in C].$$

(The standard definition fixes $c = 1$.)

It is not surprising to find that all p-generic sets are p-superterse.

Theorem 8.6. *If A is p-generic then A is p-superterse.*

Proof: Suppose that A is p-generic, but not k-p-superterse. By Fact 2.19, there exists a polynomial-time Turing machine M' that, on input $(x_1, \ldots, x_k)$, outputs a set of 2^{k-1} answers, such that one of the answers is $F_k^A(x_1, \ldots, x_k)$. Let

$$C = \{\sigma \in \{0,1\}^* : \langle \sigma(|\sigma| - k + 1), \ldots, \sigma(|\sigma|) \rangle \notin M'(0^{|\sigma|-k+1}, \ldots, 0^{|\sigma|})\},$$

where $\sigma(i)$ denotes the i-th character of σ. Obviously C is in P.

Let $n = |\sigma|$. Choose a string τ of length k such that $\tau \notin M'(0^{n+1}, \ldots, 0^{n+k})$. Then $\sigma\tau \in C$. Since every string can be extended by a constant amount to obtain an element of C there is an s such that $A[s] \in C$. Hence

$$\langle A(0^{s-k+1}), \ldots, A(0^s) \rangle \notin M'(0^{s-k+1}, \ldots, 0^s),$$

which contradicts the nature of M'. Thus A is indeed k-p-superterse. Since A is k-p-superterse for every k, A is p-superterse. ▮

The following could also be proved (of course) by a simple diagonalization.

Corollary 8.7. *There is a p-superterse set in EXPTIME.*

Proof: By [5] there are p-generic sets in EXPTIME. By Theorem 8.6 they are p-superterse. ▮

Corollary 8.8. *Cheatable sets are not $\leq_{\mathrm{T}}^{p}$-hard for EXPTIME. Non-p-superterse sets are not $\leq_{\mathrm{btt}}^{p}$-hard for EXPTIME.*

Proof:

Assume, by way of contradiction, that there exists a set A that is cheatable and $\leq_{\mathrm{T}}^{p}$-hard for EXPTIME. Let B be a p-generic set (hence also p-superterse) in EXPTIME. Since A is $\leq_{\mathrm{T}}^{p}$-hard, $B \leq_{\mathrm{T}}^{p} A$. By Lemma 3.19 this B is cheatable, which contradicts B being p-superterse.

The proof for non-p-superterse sets is similar except that it uses downward closure of non-p-superterse under $\leq_{\mathrm{btt}}^{p}$, which can be proven using the techniques in [11]. ▮

Corollary 4.3 is stronger than the above corollary since it asserts that non-p-superterse sets are not $\leq_{\mathrm{tt}}^{p}$-hard for EXPTIME.

8.4. P-Superterse Degrees

In this section we investigate which degrees contain sets that are cheatable, p-superterse, k-p-superterse, etc. As a corollary, we show that every set that is $\le^p_{k\text{-tt}}$-hard for the Boolean hierarchy is p-superterse, unless P = NP. Then we examine the structure of the lattice formed by the p-superterse degrees.

Definition 8.9. Let $\le^p_r$ denote a polynomial-time reducibility. A $\le^p_r$-*degree* is an equivalence class of the relation $\equiv^p_r$. We say that a degree is cheatable, p-superterse, etc., if it contains a set that is respectively cheatable, p-superterse, etc.

Notation 8.10. We denote degrees of all types by boldface. By convention, if **a** is a (p-superterse, p-terse, etc.) degree then A will be a (p-superterse, p-terse, etc.) set in that degree.

If $\le^p_r$ is not too strong a reducibility, then we will see that every $\le^p_r$-degree is p-superterse iff it is not cheatable.

Lemma 8.11. *Let A be a set.*

i. If A is k-cheatable then every $B \equiv^p_{\mathrm{T}} A$ is k-cheatable.

ii. If A is not k-cheatable then there exists a k-p-superterse set $B \equiv^p_{2^k\text{-tt}} A$.

Proof:
i. Assume A is k-cheatable. By Lemma 3.22 every set $B \le^p_{\mathrm{T}} A$ is k-cheatable.
ii. Assume A is not k-cheatable. Then $(\forall X)[F^A_{2^k} \notin \mathrm{PF}^X_{k\text{-T}}]$. Find the maximum $a < 2^k$ such that $(\exists X)[F^A_a \in \mathrm{PF}^X_{k\text{-T}}]$. By Fact 2.16.*iii* there exists $B \equiv^p_{2^k-\mathrm{tt}} A$ such that $F^A_a \in \mathrm{PF}^B_{k\text{-tt}}$. We prove by contradiction that B is k-p-superterse. If B is not k-p-superterse then $F^B_k \in \mathrm{PF}^Z_{k-1\text{-T}}$ for some Z, so $F^A_a \in \mathrm{PF}^Z_{k-1\text{-T}}$. Hence $F^A_{a+1} \in \mathrm{PF}^{Z\oplus A}_{k\text{-T}}$. This contradicts the maximality of a. ∎

Corollary 8.12. *Let* **d** *be a $\le^p_{\mathrm{tt}}$- or $\le^p_{\mathrm{T}}$-degree. Then* **d** *is p-superterse iff* **d** *is not cheatable.*

Definition 8.13. $\mathrm{P}^A_{\mathrm{btt}}$ is the class of sets that are $\le^p_{\mathrm{btt}} A$ ($\le^p_{\mathrm{btt}}$ is defined in Definition 2.3). $\mathrm{P}^{\mathrm{SAT}}_{\mathrm{btt}}$ is referred to as the *Boolean Hierarchy* [24] and will be denoted by BH.

Corollary 8.14. *Let A be a non-cheatable set. If there exists k such that B is $\le^p_{k\text{-tt}}$-hard for $\mathrm{P}^A_{\mathrm{btt}}$ then B is p-superterse.*

Proof: Since A is not cheatable, $\mathrm{P}^A_{\mathrm{btt}}$ contains a j-p-superterse set C_j for every j. Since $C_j \leq^p_{k\text{-tt}} B$, B must be j-p-superterse for every j. ▮

Corollary 8.15. *If* $\mathrm{P} \neq \mathrm{NP}$ *then every set that is* $\leq^p_{k\text{-tt}}$*-hard for* BH *is p-superterse.*

Proof: Assume that B is $\leq^p_{k\text{-tt}}$-hard for $\mathrm{BH} = \mathrm{P}^{\mathrm{SAT}}_{\mathrm{btt}}$. If B is not p-superterse then, by Corollary 8.14 SAT is cheatable. By Corollary 3.23 this implies $\mathrm{P} = \mathrm{NP}$. ▮

This corollary has been superseded by Beigel, Kummer, and Stephan [17, Corollary 4.5] who showed that if $\mathrm{P} \neq \mathrm{NP}$ then every set which is btt-hard for NP is p-superterse.

8.4.1. Structure of the P-Superterse Degrees

In this section we use techniques of Ladner [52] and Ambos-Spies [2, 3] to examine the structure of the polynomial many-one and Turing degrees that contain p-superterse sets. Our main contribution is the proof that certain classes are recursively presentable, and hence that Ambos-Spies' machinery can be applied.

Ladner [52] has constructed (assuming P≠NP) a set in $\mathrm{NP} - \mathrm{P}$ that is not NP-complete. Similar techniques can be used to prove the following theorem.

Theorem 8.16. *If* SAT *is p-superterse (p-terse, k-p-terse) then* NP *contains a set that is p-superterse (p-terse, k-p-terse) but is not* NP*-complete.*

Corollary 8.17. *If* $\mathrm{P} \neq \mathrm{NP}$ *then there exists a 2-p-terse set in* NP *that is not* NP*-complete.*

Proof: By [6, 14], if P≠NP then SAT is 2-p-terse. Assuming P≠NP, Theorem 8.16 yields a set in NP that is 2-p-terse but not NP-complete. ▮

Ladner's techniques have been extended in many papers dealing with the structure of the $\leq^p_{\mathrm{m}}$- and $\leq^p_{\mathrm{T}}$-degrees [2, 3, 22, 29, 53, 55]. For example, it is known that the p-degrees are dense. Ambos-Spies codified the constructions in a particularly nice way, leading to easy proofs of virtually all the previous results, as well as some new results. We apply his techniques to yield many theorems about the structure of the $\leq^p_{\mathrm{m}}$- and $\leq^p_{\mathrm{T}}$-degrees that contain p-terse sets. The techniques apply to p-superterse, non-p-selective (Definition 3.11), and many other kinds of sets.

Notation 8.18. Throughout this section "r" will denote either m or T, i.e., if one replaces all of the r's with m's or replaces all of the r's with T's, the results will hold.

Definition 8.19. If U is a set and $i \in \mathsf{N}$ then let $U^i = \{x : \langle x, i\rangle \in U\}$. A class of recursive sets $\mathcal{A}$ is *recursively presentable* if there exists a recursive set U such that $\mathcal{A} = \{U^i : i \in \mathsf{N}\}$.

In order to apply Ambos-Spies's techniques we must prove that for all A the class of non-p-superterse sets X such that $X \leq_r^p A$ is a recursively presentable class.

Lemma 8.20. *Let A be any recursive set and $i, j \in \mathsf{N}$. The following class of sets is recursively presentable.*

$$\mathcal{C}_1 = \{C : C \leq_r^p A \text{ and } (\exists Z)[F_i^C \in \mathrm{PF}_{j\text{-T}}^Z]\}$$

Proof: We prove the lemma for Turing reducibility, but a similar proof holds for many-one reducibility.

If $i \leq j$ then $\mathcal{C}_1$ is the class of all sets that are $\leq_r^p A$, which is recursively presentable [3]. Henceforth we assume $i > j$.

By Fact 2.19, if there exists Z such that $F_i^C \in \mathrm{PF}_{j\text{-T}}^Z$ then F_i^C is 2^j-enumerable. Hence there is a function f such that $f(x_1, \ldots, x_i)$ is a set of 2^j elements of $\{0,1\}^i$ (suitable coded), one of which is $F_i^C(x_1, \ldots, x_i)$. We refer to such an f as a *covering function for C.*

Let $P_e^{()}$ be the e-th polynomial time-bounded deterministic oracle Turing machine. If X is a set then P_e^X denotes the language recognized by P_e with oracle X. If no oracle is shown then the empty oracle is assumed. For every $m = \langle e_1, e_2 \rangle$ we will construct a set $E_m \in \mathcal{C}_1$. We will guarantee that

$$E_m = \begin{cases} P_{e_1}^A & \text{if } P_{e_2} \text{ computes some covering function} \\ \text{a finite set} & \text{otherwise.} \end{cases}$$

ALGORITHM for E_m

Step 1: Input(x).

Step 2: For all i-tuples $(x_1, \ldots, x_i)$ such that every component x_i is less than x (lexicographically), compute $F_i^{E_m}(x_1, \ldots, x_i)$ by calling the algorithm recursively.

Step 3: For all i-tuples $(x_1, \ldots, x_i)$ such that every element is less than x, compute $P_{e_2}(x_1, \ldots, x_i)$.

Step 4: If there exists an i-tuple $(x_1, \ldots, x_i)$ of elements less than x such that either $F_i^{E_m}(x_1, \ldots, x_i) \notin P_{e_2}(x_1, \ldots, x_i)$, or $P_{e_2}(x_1, \ldots, x_i)$ represents a set with more than 2^j strings, then output NO. Else run $P_{e_1}^A(x)$ and output the answer.

END OF ALGORITHM.

If E_m is finite then $E_m \in \mathcal{C}_1$ trivially. If E_m is infinite then (by step 4) for every i-tuple $(x_1, \ldots, x_i)$, $P_{e_2}(x_1, \ldots, x_i)$ represents a set of at most 2^j elements and $F_i^{E_m}(x_1, \ldots, x_i) \in P_{e_2}(x_1, \ldots, x_i)$. Hence P_{e_2} computes a covering function for E_m. In addition, $E_m \leq_{\mathrm{T}}^p A$ via P_{e_1}. Hence $E_m \in \mathcal{C}_1$. Therefore for each m, $E_m \in \mathcal{C}_1$.

If $C \in \mathcal{C}_1$, then there exists an $m = \langle e_1, e_2 \rangle$ such that $C = L(P_{e_1}^A)$ and $F_i^C \in \mathrm{PF}_{j\text{-T}}^Z$ via covering function P_{e_2}. Hence every set in $\mathcal{C}_1$ is of the form E_m for some m.

Finally, define $U(\langle x, m \rangle) = E_m(x)$. Thus $\mathcal{C}_1$ is recursively presentable. ∎

Lemma 8.21. *Let A, B be recursive sets and let $i, j \in \mathsf{N}$. The following classes are recursively presentable.*

$$\begin{aligned}\mathcal{C}_2 &= \{C : C \leq_r^p A \textit{ and } C \textit{ is not p-superterse}\}\\ \mathcal{D}_2 &= \{C : A \leq_r^p C \leq_r^p B \textit{ and } C \textit{ is not p-superterse}\}\end{aligned}$$

Proof: The result for $\mathcal{C}_2$ follows from the preceding lemma and the fact that the recursive union of recursively presentable classes is recursively presentable. Then the result for $\mathcal{D}_2$ follows by standard techniques of [3]. ▮

Theorem 8.22. *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be any two recursive $\leq_r^p$-degrees such that* $\mathbf{a} <_r \mathbf{b}$ *and* $\mathbf{b}$ *is p-superterse. Let L be any countable distributive lattice. L can be embedded into the p-superterse $\leq_r^p$-degrees on the interval between* $\mathbf{a}$ *and* $\mathbf{b}$.

Proof: Ambos-Spies showed that given any two $\leq_r^p$-degrees $\mathbf{a}$ and $\mathbf{b}$ such that $\mathbf{a} <_r \mathbf{b}$, and given any recursively presentable class $\mathcal{D}$ such that $\mathbf{b} \notin \mathcal{D}$, any countable distributive lattice L can be embedded into the degrees in $[\mathbf{a}, \mathbf{b}]$ that do not contain sets in $\mathcal{D}$. Taking $\mathcal{D} = \mathcal{D}_2$, the set of non-p-superterse degrees in $[\mathbf{a}, \mathbf{b}]$, yields the theorem. ▮

Corollary 8.23. *If* $\mathrm{P} \neq \mathrm{NP}$ *and L is any countable distributive lattice then L can be embedded into the p-superterse* NP *$\leq_r^p$-degrees.*

Proof: By Lemma 8.21 the class of non-p-superterse sets in NP is a recursively presentable class (taking $A = \mathrm{SAT}$ in the definition of $\mathcal{C}_2$). The result now follows from Theorem 8.22 by taking $\mathbf{a}$ to be the trivial degree and $\mathbf{b}$ to be the NP-complete degree. ▮

As corollaries, we see that the preceding results apply if we replace p-superterse degrees with p-terse, k-p-terse, or non-p-selective [66] degrees. The techniques in this section apply, as well, to non-self-reducible and non-near-testable [35] degrees.

It is open whether the structure of the p-terse $\leq_r^p$ degrees is isomorphic to the structure of the $\leq_r^p$ degrees (where r is either m or T). One possible difference is that the $\leq_{\mathrm{m}}^p$ degrees form a distributive upper semi-lattice, while for the p-terse $\leq_{\mathrm{m}}^p$ degrees this is open.

9. Summary

We set out to identify properties of non-p-superterse and cheatable sets in order to show that natural sets are p-superterse or at least not cheatable, unless they are in P. We have determined that non-p-superterse sets belong to P/poly, so they are in EL_3. Furthermore, if A is non-p-superterse via an algorithm that queries only components of the input, then A is in EL_2. In particular, cheatable and p-selective sets are in EL_2.

The class of cheatable sets is closed under $\leq_{\mathrm{T}}^{p}$-reducibility. As a consequence, P is equal to the class of all sets that are cheatable and self-reducible. A corollary is that P = NP iff there exists a cheatable set that is NP-hard under Turing reductions. Beigel, Kummer, and Stephan [17] have extended this by showing that if a non-p-superterse is NP-hard under Bounded Truth-table reductions, then P = NP.

We summarize some necessary conditions for cheatability: A cheatable set must be recursive [15], must have polynomial-size circuits, and must be in EL_2. It cannot be self-reducible or near-testable [35] unless it is in P. It cannot be EXPTIME-hard. It cannot be NP-hard unless P = NP. It cannot be PSPACE-hard unless P = PSPACE. It cannot be $\leq_{\mathrm{T}}^{p}$-hard for a well-behaved counting class #C unless $\mathrm{P} = \mathrm{P}^{\#\mathrm{C}}$. In addition, a 1-cheatable set cannot be bi-immune for P [6].

The techniques developed for the study of non-p-superterse sets also work when the number of membership tests grows as a function of input length. When $f(n) = O(\log n)$, we have

$$(\forall X)[\mathrm{PF}^{A}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}],$$

unless $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$. Assuming that $\Sigma_3^p \neq \Pi_3^p$, this lets us extend a result of Krentel [51] by showing that

$$(\forall X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}].$$

When $\epsilon < 1$ and $f(n) \leq n^{\epsilon}$, we have

$$(\forall X)[\mathrm{PF}^{A}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}],$$

unless $A \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$. Assuming that $\Sigma_4^p \neq \Pi_4^p$, we have

$$(\forall X)[\mathrm{PF}^{\Sigma_3^p}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}].$$

Although the analogous statement for NP queries is open, our result is an exponential improvement upon the level at which a natural query hierarchy could be separated assuming only that the polynomial hierarchy is proper.

Looking more carefully at the size and depth of the circuits produced by our techniques, we extend a result of Cai [23]: If there is a circuit of size s and depth d that computes $F_k^{A^{=n}}(x_1, \ldots, x_k)$ while making only $k-1$ oracle queries to some oracle X then there is a circuit of size $n^{k+o(k)}s$ and depth $d + O(1)$ that accepts $A^{=n}$.

Applying our techniques to the study of counting problems we obtain extensions (though not generalizations) of results of Cai and Hemachandra [27]. We show that #SAT is not $2^{n^{\epsilon}}$-enumerable unless $\mathrm{P}^{\#\mathrm{P}} = \Sigma_4^p \cap \Pi_4^p$. Using this we also obtained, using a Theorem of Toda and Watnabe, that #SAT is not n^k-enumerable unless $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$. This result was obtained independently by Cai and Hemachandra [26]. In addition we noted that the enumerability of #SAT and PERM01 seem to be different.

10. Acknowledgements

An early draft of this paper was written while the second author was visiting the Department of Mathematics at the University of California at Santa Barbara. We are appreciative of Ron Book's advice and hospitality during that time.

We thank Eric Allender for pointing out that we obtain membership in EL_2 in Theorem 3.4, Pankaj Rohatgi for the current form of Theorem 4.4, Carsten Lund for help on Section 7, Frank Stephan for help with Lemma 3.30, and Seinosuke Toda for making the observation that led to Corollary 7.32.

We also thank José Balcázar Sandeep Bhatt, Michael Fisher, Albrecht Hoene, Steve Homer, Anna Karlin, Rao Kosaraju, Martin Kummer, Alan Selman, and Frank Stephan for helpful discussions.

We would like to thank Jorge Castro, Richard Chang, James Foster, Georgia Martin, Carlos Seara, Frank Stephan, and Todd Wareham for proofreading and commentary.

11. Appendix

11.1. A Variant on the BP Operator

Schöning [65] defined the **BP** operator as a generalization of the complexity class BPP. We will need a generalization $\mathbf{BP}^\bullet$ of the **BP** operator. Our approach closely follows his; hence we provide only sketches.

Recall Schöning's definition of the **BP** operator.

Definition 11.1. Let $\mathcal{C}$ be a class of sets. We say that $A \in \mathbf{BP} \cdot \mathcal{C}$ iff there exists a set $B \in \mathcal{C}$ and a polynomial p such that for all $n \in \mathsf{N}$, for all $x \in \Sigma^n$,

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B \mid y \in \Sigma^{p(n)} \text{ uniformly }) \geq \frac{3}{4}; \\ x \notin A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \notin B \mid y \in \Sigma^{p(n)} \text{ uniformly }) \geq \frac{3}{4}. \end{array}$$

In Schöning's definition the string y ranged over all of $\Sigma^{p(n)}$. We need to consider what happens if y ranges over some subset $Y \subseteq \Sigma^{p(n)}$.

Definition 11.2. Let $\mathcal{C}$ be a class of sets. We say that $A \in \mathbf{BP}^\bullet \cdot \mathcal{C}$ iff there exists a set $B \in \mathcal{C}$, a polynomial p, and a set $Y \subseteq \Sigma^*$ such that for all $n \in \mathsf{N}$, for all $x \in \Sigma^n$,

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B \mid y \in Y \cap \Sigma^{p(n)} \text{ uniformly }) \geq \frac{3}{4}; \\ x \notin A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \notin B \mid y \in Y \cap \Sigma^{p(n)} \text{ uniformly }) \geq \frac{3}{4}. \end{array}$$

Definition 11.3. Let A, B be sets. We say that $A \leq_{pos}^{p} B$ iff $A \leq_{T}^{p} B$ via an oracle Turing machine $M^{()}$ with the additional property that $X \subseteq Y \Rightarrow L(M^X) \subseteq L(M^Y)$. A class of sets $\mathcal{C}$ is *closed under positive reductions* if for all A, B, if $B \in \mathcal{C}$ and $A \leq_{pos}^{p} B$ then $A \in \mathcal{C}$.

Lemma 11.4. *Let $\mathcal{C}$ be a class of sets closed under positive reductions. For any set $A \in \mathbf{BP}^{\bullet} \cdot \mathcal{C}$ and any polynomial q there is a set $B \in \mathcal{C}$, a polynomial p, and a set $Y \subseteq \Sigma^*$ such that for all $n \in \mathsf{N}$, for all $x \in \Sigma^n$,*

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B \mid y \in Y \cap \Sigma^{p(n)}) \geq 1 - \frac{1}{2^{q(n)}}; \\ x \notin A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \notin B \mid y \in Y \cap \Sigma^{p(n)}) \geq 1 - \frac{1}{2^{q(n)}}; \end{array}$$

Proof:

Since $A \in \mathbf{BP}^{\bullet} \cdot \mathcal{C}$ there exists a set $B' \in \mathcal{C}$, a polynomial p' and a set $Y' \subseteq \Sigma^*$ such that for all $x \in \Sigma^n$,

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B' \mid y \in Y' \cap \Sigma^{p'(n)}) \geq \frac{3}{4}; \\ x \notin A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \notin B' \mid y \in Y' \cap \Sigma^{p'(n)}) \geq \frac{3}{4}. \end{array}$$

Let

$$\begin{array}{rl} t(n) = & \lceil \frac{2}{\log 4/3} q(n) \rceil \\ Y = & \bigcup_{n=0}^{\infty} \{ \langle y_1, \ldots, y_{t(n)} \rangle \mid (\forall i)[|y_i| = p'(n) \text{ and } y_i \in Y'] \}; \\ B = & \bigcup_{n=0}^{\infty} \{ \langle x, \langle y_1, \ldots, y_{t(n)} \rangle \rangle \mid \text{ a majority of the } \langle x, y_i \rangle \text{ are in } B' \}. \end{array}$$

Let $p(n)$ be the length of $\langle y_1, \ldots, y_{t(n)} \rangle$ where all the y_i are of length $p'(n)$ (this $p(n)$ will depend on $t(n)$ and the pairing function being used).

Note that $B \in \mathcal{C}$ since $\mathcal{C}$ is closed under positive reductions. The rest of the proof proceeds exactly like [65, Lemma 3.3]. ∎

Theorem 11.5. *If $\mathcal{C}$ is closed under positive reductions then $\mathbf{BP}^{\bullet} \cdot \mathcal{C} \subseteq \mathcal{C}/\mathrm{poly}$.*

Proof: Let $A \in \mathbf{BP}^{\bullet} \cdot \mathcal{C}$. Apply Lemma 11.4 with $q(n) = n + 1$ to obtain $B \in \mathcal{C}$, a polynomial p, and $Y \subseteq \Sigma^*$ such that

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B \mid y \in Y \cap \Sigma^{p(n)}) \geq 1 - \frac{1}{2^{n+1}}; \\ x \notin A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B \mid y \notin Y \cap \Sigma^{p(n)}) \geq 1 - \frac{1}{2^{n+1}}. \end{array}$$

We give a probabilistic argument which yields that for all n there exists a $y \in \Sigma^{p(n)}$ such that $(\forall x)[x \in A \text{ iff } B(x, y)]$.

$$\begin{array}{rl} \mathrm{Prob}((\exists x \in \Sigma^n) & [B(x,y) \neq A(x)] \mid y \in Y \cap \Sigma^{p(n)}) \leq \\ & \sum_{x \in \Sigma^n} \mathrm{Prob}(B(x,y) \neq A(x) \mid y \in Y \cap \Sigma^{p(n)}) \\ \leq & (|\Sigma|)^n \cdot \frac{1}{2^{n+1}} \\ \leq & \frac{1}{2} \end{array}$$

Hence at least one such y must exist. That string y can serve as advice. ∎

Definition 11.2 requires that its given condition hold for all n. However, we will be dealing with languages where this condition only holds for some n, and for those n, we still need polynomial advice.

Theorem 11.6. *Let $\mathcal{C}$ be any class of languages closed under positive reductions. Let $I \subseteq \mathsf{N}$. Let A be a set such that there exists $B \in \mathcal{C}$, a polynomial p, a set $Y \subseteq \Sigma^*$ such that for all $n \in I$, for all $x \in \Sigma^n$,*

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \in B \mid y \in Y \cap \Sigma^{p(n)}) \geq \frac{3}{4}; \\ x \notin A & \Rightarrow \mathrm{Prob}(\langle x, y\rangle \notin B \mid y \in Y \cap \Sigma^{p(n)}) \geq \frac{3}{4}. \end{array}$$

Then $A' = A \cap \bigcup_{n \in I} \Sigma^n \in \mathcal{C}/\mathrm{poly}$.

Proof:

For each length n one bit of advice tells if $n \in I$ or not. If $n \in I$ then proceed as in Theorem 11.5, using an analogue of Lemma 11.4. If $n \notin I$ then clearly $A' \cap \Sigma^n = \emptyset$. ∎

11.2. A Variant of the Ajtai/Ben-Or Construction

Ajtai and Ben-Or [1] showed how to convert a probabilistic circuit into a deterministic circuit with only a constant increase in depth and a polynomial increase in size. We need a variant of their theorem. Our approach closely follows theirs; hence we provide only sketches.

Recall the standard definition of a probabilistic circuit.

Definition 11.7. The circuit model allows negation, unbounded fan-in and-gates, and unbounded fan-in or-gates. The inputs are $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$ and their negations. The x_i are called *input variables* and the y_i are called *random variables*. Let $0 \leq q \leq p \leq 1$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let C be a probabilistic circuit on n inputs and m random variables. The circuit C *separates A from B with probabilities* (p, q), denoted by (C, A, B, p, q), if

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(C(x, y) = 1 \mid y \in \{0,1\}^m \text{ uniformly }) \geq p; \\ x \in B & \Rightarrow \mathrm{Prob}(C(x, y) = 1 \mid y \in \{0,1\}^m \text{ uniformly }) \leq q. \end{array}$$

In this definition the string y ranged over all of $\{0,1\}^m$. We need to consider what happens if y ranges over some subset $Y \subseteq \{0,1\}^m$.

Definition 11.8. Let $0 \leq q \leq p \leq 1$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let C be a probabilistic circuit on n inputs and m random variables. Let $Y \subseteq \{0,1\}^m$. The circuit C *separates A from B with probabilities (p, q) using Y*, denoted by (C, A, B, p, q, Y) if for all $x \in \Sigma^n$,

$$\begin{array}{ll} x \in A & \Rightarrow \mathrm{Prob}(C(x, y) = 1 \mid y \in Y \cap \{0,1\}^m \text{ uniformly }) \geq p; \\ x \in B & \Rightarrow \mathrm{Prob}(C(x, y) = 1 \mid y \in Y \cap \{0,1\}^m \text{ uniformly }) \leq q. \end{array}$$

The following is Lemma 1 from [1] in our framework.

Lemma 11.9. *Let $n, m \in \mathbb{N}$ and $0 \le q \le p \le 1$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let $Y \subseteq \{0,1\}^m$. Let C be a probabilistic circuit such that C has size s, depth d, and (C, A, B, p, q, Y).*

i. If $p \ge p_1$ and $q \le q_1$ then (C, A, B, p_1, q_1, Y).

ii. There is a circuit C^b such that C^b has size s, depth d, and $(C^b, B, A, 1-q, 1-p, Y)$.

iii. For any natural number u there is a circuit C^u and a set $Y' \subseteq \{0,1\}^m$ such that C^u has size $us+1$ and depth $d+1$, and $(C^u, A, B, p^u, q^u, Y')$.

iv. If $1-p-q < 2^{-n}$ then there is a deterministic circuit C^d of size s, depth d such that $(C^d, A, B, 1, 0)$.

Proof:

i. and *ii.* follow from the definition.

iii. Let C^u consist of u independent copies of C where all the outputs are connected to one $\wedge$ gate. Let $Y' = Y \times \cdots \times Y$ (u times). C^u had depth size $us+1$ and depth $d+1$. Since probabilities multiply clearly $(C^u, A, B, p^u, q^u, Y')$.

iv. Identical to Lemma 1.*d* of [1]. ∎

Lemmas 2 and 3, and Theorem 4, of [1] can now be proven for the probabilistic circuits defined in this section. Hence we obtain the following.

Lemma 11.10. *Let $\{C\}_{n=1}^{\infty}$ be an $(s(n), d(n))$ probabilistic circuit family where C_n has $k(n)$ random variables. Let $\{Y\}_{n=1}^{\infty}$ be such that $Y_n \subseteq \{0,1\}^{k(n)}$. Assume that for all n $(C_n, A_n, \overline{A}_n, \frac{3}{4}, \frac{1}{4}, Y_n)$. Then there exists an $(n^{O(1)}s(n), d(n)+O(1))$ deterministic circuit family for A.*

Ajtai and Ben-Or actually proved a stronger result, which would also be true in our framework. For simplicity we state only what we need.

The proofs of Ajtai and Ben-Or (and hence our modifications) only require that we be allowed to adjust the circuits by using AND, OR and NOT. In particular they hold if the circuits have other types of gates such as MOD_a or MAJ. Hence we have the following.

Lemma 11.11. *Let $\{C\}_{n=1}^{\infty}$ be an $(s(n), d(n))$ probabilistic circuit family where C_n has $k(n)$ random variables and C_n may have MOD and MAJ gates. Let $\{Y\}_{n=1}^{\infty}$ be such that $Y_n \subseteq \{0,1\}^{k(n)}$. Assume that for all n $(C_n, A_n, \overline{A}_n, \frac{3}{4}, \frac{1}{4}, Y_n)$. Then there exists an $(n^{O(1)}s(n), d(n)+O(1))$ deterministic circuit family for A.*

References

[1] M. Ajtai and M. Ben-Or. A theorem on probabilistic constant depth circuits. In *Proc. of the 16th ACM Sym. on Theory of Computing*, pages 471–474, 1984.

[2] K. Ambos-Spies. On the structure of the polynomial time degrees of recursive sets. Technical report, Sept. 1984. This is the author's Habilitationsschrift.

[3] K. Ambos-Spies. Sublattices of the polynomial time degrees. *Information and Control*, 65:63–84, 1985.

[4] K. Ambos-Spies. Inhomogeneities in the polynomial-time degrees: The degrees of super sparse sets. *Inf. Process. Lett.*, 22:113–118, 1986.

[5] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalization over polynomial time computable sets. *Theoretical Comput. Sci.*, pages 177–204, 1987.

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

[7] L. Babai, 1994. Personal communication.

[8] J. L. Balcázar, R. V. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *J. ACM*, 33(3):603–617, July 1986.

[9] J. L. Balcázar, R. V. Book, and U. Schöning. Sparse sets, lowness and highness. *SIAM J. Comput.*, 15(3):739–747, Aug. 1986.

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

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

[12] R. Beigel. NP-hard sets are p-superterse unless R=NP. Technical Report 4, The Johns Hopkins University, Dept. of Computer Science, 1988.

[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.*, 84:199–223, 1991.

[15] R. Beigel, W. I. Gasarch, J. T. Gill, and J. C. Owings. Terse, superterse, and verbose sets. *Information and Computation*, 103:68–85, 1993.

[16] R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Comput. Sci.*, 103(1):3–23, 1992.

[17] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. In *Proc. of the 9th Annual Conference on Structure in Complexity Theory*, pages 12–23. IEEE Computer Society Press, 1994.

[18] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55:80–88, 1982.

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

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

[21] R. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 757–804. MIT Press and Elsevier, The Netherlands, 1990.

[22] S. Breidbart. On splitting recursive sets. *J. Comput. Syst. Sci.*, 17:56–64, 1978.

[23] J. Cai. Lower bounds for constant depth circuits in the presence of help bits. *Inf. Process. Lett.*, 36:79–84, 1990. Also appeared in FOCS 1989.

[24] 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.

[25] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23:95–106, 1990.

[26] J. Cai and L. Hemachandra. A note on enumerative counting. *Inf. Process. Lett.*, 38(4):215–219, 1991. Was University of Rochester TR 330 in Feb 1990.

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

[28] R. Chang, W. Gasarch, and J. Torán. On the number of automorphisms of a graph, 1994. Unpublished manuscript.

[29] P. Chew and M. Machtey. A note on structure and looking back applied to the relative complexity of computable functions. *J. Comput. Syst. Sci.*, 22:53–59, 1981.

[30] S. Clarke, J. Owings, and J. Spriggs. Trees with full subtrees. In *Proc. of the 6th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 169–172, 1975.

[31] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, 1979.

[32] W. Gasarch, M. W. Krentel, and K. Rappoport. OptP-completeness as the normal behavior of NP-complete problems. *Mathematical Systems Theory*. to appear.

[33] W. I. Gasarch. The complexity of optimization functions. Technical Report 1652, University of Maryland, Dept. of Computer Science, 1985.

[34] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6:675–695, 1977.

[35] J. Goldsmith, L. Hemachandra, D. Joseph, and P. Young. Near-testable sets. *SIAM J. Comput.*, 20(3), 1991.

[36] J. Goldsmith, D. Joseph, and P. Young. Using self-reducibility to characterize polynomial time. *Information and Computation*, 104:288–308, 1993.

[37] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, pages 186–208, 1989.

[38] J. Hastad. Almost optimal lower bounds for small depth circuits. In *Proc. of the 18th ACM Sym. on Theory of Computing*, pages 6–20, 1986.

[39] L. Hemaspaandra, A. Hoene, A. Naik, M. Ogiwara, A. Selman, T. Thierauf, and J. Wang. Selectivity: Reductions, nondeterminism, and function classes. Technical Report 69, University of Rochester, Department of Computer Science, Aug. 1993. This TR is around 40 pages.

[40] L. Hemaspaandra, A. Hoene, M. Ogiwara, A. Selman, T. Thierauf, and J. Wang. Selectivity. In *Proc. of the 5th International Conference on Computing and Information*, pages 55–59. IEEE Computer Society Press, 1993.

[41] L. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. Computing solutions uniquely collapses the polynomial hierarchy. Technical Report 467, University of Rochester, Department of Computer Science, Aug. 1993. This TR is around 12 pages.

[42] L. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. Computing solutions uniquely collapses the polynomial hierarchy. In *Proc. of the 5th International Sym. on Algorithms and Computation*, Lecture Notes in Computer Science, Berlin, 1994. Springer-Verlag. to appear.

[43] A. Hoene and A. Nickelsen. Counting, selecting, sorting by query-bounded machines. In *Proc. of the 10th Sym. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, 1993.

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

[45] C. G. Jockusch. Degrees of generic sets. In F. Drake and S. Wainer, editors, *Recursion Theory: its generalizations and applications*, pages 110–139. Cambridge University Press, 1980.

[46] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. of the 12th ACM Sym. on Theory of Computing*, pages 302–309, 1980.

[47] Kintala and Fischer. Real-time computations with restricted nondetermism. *Mathematical Systems Theory*, 12, 1979.

[48] K.-I. Ko. On self-reducibility and weak p-selectivity. *J. Comput. Syst. Sci.*, 26:209–221, 1983.

[49] K.-I. Ko and U. Schöning. On circuit-size complexity and the low hierarchy in NP. *SIAM J. Comput.*, 14(1):41–51, Feb. 1985.

[50] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhauser, Boston, 1993.

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

[52] R. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, Jan. 1975.

[53] L. Landweber, R. Lipton, and E. Robertson. On the structure of sets in NP and other complexity classes. *Theoretical Comput. Sci.*, 15:181–200, 1981.

[54] M. Lerman. *Degrees of Unsolvability*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1983.

[55] K. Mehlhorn. Polynomial and abstract subrecursive classes. *J. Comput. Syst. Sci.*, 12:147–178, 1976.

[56] A. Meyer and M. Paterson. With what frequency are apparently intractable problems difficult? Technical Report MIT/LCS/TM-126, M. I. T., 1979.

[57] M. Ogihara. Polynomial-time membership comparable sets. In *Proc. of the 9th Annual Conference on Structure in Complexity Theory*, pages 2–11. IEEE Computer Society Press, 1994.

[58] M. Ogiwara and A. Lozano. On sparse hard sets for counting classes. *Theoretical Comput. Sci.*, 112:255–275, 1993.

[59] C. Papadimitriou and M. Yannakakis. On limited nondetermism and the complexity of the v-c dimension. In *Proc. of the 8th Annual Conference on Structure in Complexity Theory*. IEEE Computer Society Press, 1993.

[60] C. H. Papadimitriou and S. K. Zachos. Two remarks on the complexity of counting. In *Proc. of the 6th GI Conference on Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276, Berlin, 1983. Springer-Verlag.

[61] N. Pippenger. On simultaneous resource bounds. In *Proc. of the 20th IEEE Sym. on Found. of Comp. Sci.*, pages 307–311, 1979.

[62] C. P. Schnorr. Optimal algorithms for self-reducible problems. In *Proc. of the 3th ICALP*, pages 322–337, 1976.

[63] U. Schöning. A low and a high hierarchy within NP. *J. Comput. Syst. Sci.*, 27:14–28, 1983.

[64] U. Schöning. *Complexity and Structure*. Springer-Verlag, Berlin, 1985.

[65] U. Schöning. Probabilistic complexity classes and lowness. *J. Comput. Syst. Sci.*, 39:84–100, Dec. 1989.

[66] 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.

[67] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. of the 19th ACM Sym. on Theory of Computing*, pages 77–82, 1987.

[68] L. Stockmeyer. The polynomial-time hierachy. *Theoretical Comput. Sci.*, 3:1–22, 1976.

[69] L. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proc. of the 5th ACM Sym. on Theory of Computing*, pages 1–9, 1973.

[70] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20:865–877, 1991.

[71] S. Toda and O. Watanabe. Polynomial time 1-Turing reductions from #PH to #P. *Theoretical Comput. Sci.*, 100:205–221, 1992.

[72] L. G. Valiant. The complexity of computing the permanent. *Theoretical Comput. Sci.*, 8:189–201, 1979.

[73] K. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

[74] K. W. Wagner. Bounded query classes. *SIAM J. Comput.*, 19(5):833–846, Oct. 1990.

[75] C. B. Wilson. Relativized circuit complexity. *J. Comput. Syst. Sci.*, 31(2):169–181, Oct. 1985.

[76] C. Wrathall. Complete sets and the polynomial-time hierachy. *Theoretical Comput. Sci.*, 3:23–33, 1976.

[77] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. of the 26th IEEE Sym. on Found. of Comp. Sci.*, pages 1–10, 1985.

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