Bounded Queries in Recursion Theory: A Survey

William Gasarch*
Dept. of Computer Science and Institute for Advanced Studies
University of Maryland
College Park, MD 20742

1 Introduction

Imagine that you are given an oracle for the halting set but you can only ask it (say) 5 questions (no time or space bound on computation). What can you compute? Could you compute more with 6 queries? More generally, if $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$ then

- What functions can we compute with m queries to A?
- Are there functions that can be computed with m queries to A that cannot be computed with $m-1$ queries to A? To any set X?

In this paper we survey much of the work that has been done on these two questions. Our framework is recursion-theoretic– the computations have no time or space bound.

Several people have studied this problem with different motivations:

While Richard Beigel was looking for a thesis topic, he (together with Jon Seigel) pondered the following question "What could be computed if a Turing machine were allowed to compute 'forever'?" Could the notion of 'forever' be quantified? One way to formalize the notion of 'forever' was to look at queries to K. One way to quantify this notion was to put explicit bounds on the number of queries. Thus the following natural question arose: "Are $i+1$ queries to K more powerful than i queries to K?" This question can be interpreted in four different ways depending on whether one is computing functions or deciding sets; and whether the queries are serial or parallel. He showed that answer is YES in all these cases [6, 16]. He later extended many of his results to general nonrecursive sets [6].

Louise Hay was initially interested in the n-r.e. sets, which are a stratification of some of the sets that are $\leq_T K$ (we define n-r.e. in Section 10). The queries-to-K hierarchy is another stratification of those same sets. Hay was concerned with how these two hierarchies relate to each other. The main theorem of [16], also stated in Section 10, answered her questions.

William Gasarch initially wanted to develop a complexity theory within recursion theory, with the number-of-queries-to-an-oracle being the resource of interest [26]. This goal is partially realized in [12, 13] where several problems about finding the chromatic number of a recursive graph are characterized in terms of bounded queries. These results will be reviewed in Section 13.

2 Technical Summary

We now formally define bounded query classes.

*Supported in part by NSF grant CCR-8803641.

Definition 2.1 Let $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$. A function f is in $FQ(m, A)$ if f is recursive in A via an algorithm that makes at most m queries to A. Later queries can depend on answers to previous queries. Such queries are called *serial*. A function f is in $FQ_{||}(m, A)$ if f is recursive in A via an algorithm that makes at most m queries that are asked all at once. Note that the queries cannot depend on previous answers. Such queries are called *parallel*.

Definition 2.2 Let $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$. A set B is in $Q(m, A)$ if its characteristic function is in $FQ(m, A)$. The class $Q_{||}(m, A)$ is defined similarly.

Bounded queries differ from truth-table reductions in two ways:

1. If $A \in Q(n, B)$ then queries can depend on previous answers. This is not the case for $A \leq_{\mathrm{tt}} B$.
2. If $A \in Q_{||}(n, B)$ or $A \in Q(n, B)$, and the wrong answers are supplied, then the computation might diverge. By contrast, if $A \leq_{\mathrm{tt}} B$ then even if the wrong answers about B are supplied, the computation terminates.
3. We parameterize the number of queries exactly. By contrast, the different notions of truth-table reduction (bounded, unbounded, and weak) differ in less refined ways.

The following definition will be helpful in several places.

Definition 2.3 If C is a set then C^{tt} is the set of all true statements that are boolean combinations of atoms of the form '$n \in C$.'

Much of the work that has been done in this area concerns how hard, in terms of number-of-queries, the following functions are to compute:

Definition 2.4 Let $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$. The functions F_m^A, $\#_m^A$, and the set V_m^A, are defined by

$$F_m^A(x_1, \ldots, x_m) = \langle \chi_A(x_1), \ldots, \chi_A(x_m) \rangle$$

$$\#_m^A(x_1, \ldots, x_m) = |\{i : x_i \in A\}|$$

$$V_m^A = \{\langle x_1, \ldots, x_m, b_1, \ldots, b_m \rangle : (\forall i) \chi_A(x_i) = b_i\}.$$

Intuitively F_m^A is asking for *membership* of m numbers, $\#_m^A$ is asking for the *cardinality* of m numbers, and V_m^A is asking to *verify* the membership status of m numbers. The most interesting theorems in this field (in the author's opinion) involve F_m^A and $\#_m^A$. They are

$(\exists X, n) F_{2^n}^A \in FQ(n, X) \Rightarrow A$ rec. [15]
$(\exists X, n) \#_{2^n}^A \in FQ(n, X) \Rightarrow A$ rec. [33]

We will refer to the latter theorem as *the cardinality theorem*.

In Sections 3 and 4 we review the prehistory of bounded queries, i.e., research in the literature that was close to this topic. In Sections 5, 7, and 8 we examine the complexity of F_m^A, $\#_m^A$, and V_m^A respectively (Section 6 is devoted to some variations on the complexity of F_m^A). Sections 9,10,11, and 12 examine the question of when extra queries increase computational power. Section 13 summarizes the work done on classifying problems in recursion theory, especially recursive graph theory, by using number-of-queries as a complexity measure. Section 14 compares and contrasts the work done on bounded queries in complexity theory with that done on bounded queries in recursion theory.

Throughout this paper we will only prove theorems that are not in the 'open' literature. Some results, though 'well known,' appear with proof here for the first time.

3 Prehistory: Kolmogorov Complexity Theory

Kolmogorov complexity (see [34, 35] for references and background) deals with how many bits are needed to produce certain objects. Both Kolmogorov Complexity and Bounded Queries are concerned with how much information is needed to compute an object; however, there is little overlap. We discuss the differences of the two fields, and then give an example of a weak overlap which will underscore the point that there is little overlap.

Definition 3.1 Let U be a fixed universal Turing machine. If x and y are finite strings then we $K_U(x \mid y)$ is the length of the shortest z such that $U(y, z) = x$.

If U_1 and U_2 are universal Turing machines then there exists a constant c such that for every x, y, we have $K_{U_1}(x \mid y) \leq K_{U_2}(x \mid y) + c$. Hence the choice of U only affects additive constants. Henceforth we fix a particular U and denote $K_U(x \mid y)$ by $K(x \mid y)$. Intuitively $K(x \mid y)$ is the number of bits needed to describe x, given y.

Notation 3.2 If A is a set then a_n is the initial segment of A of length n.

Many theorems in Kolmogorov complexity are concerned with the growth of $K(a_n \mid n)$ for various sets A. The next (easy) proposition weakly links Kolmogorov complexity to bounded queries, and is folklore.

Proposition 3.3 *Let A, Y be sets, and f a function, such that $(\forall m)[F_m^A \in FQ(f(m), Y)]$. Then there exists c such that*

$$(\forall m)[K(a_m \mid m) \leq f(m) + c].$$

Proof:

Let $F_m^A \in FQ(f(m), Y)$ via oracle Turing machine $M^{()}$. Consider the computation of $F_m^A(0, 1, \ldots, m-1)$. This computation makes $f(m)$ queries to Y. Let $b_1 b_2 \cdots b_{f(m)}$ be the correct answers to these queries. Given this sequence, the Turing machine $M^{()}$ (of constant size c), and the value m, one can easily produce the string $F_m^A(0, 1, \ldots, m-1) = a_m$. Hence from $f(m) + c + O(1)$ bits one can produce a_m. ∎

A key difference between Kolmogorov complexity and bounded queries is that Kolmogorov complexity deals with *initial segments* of sets, whereas (say) F_m^A is asking about *any* m numbers.

To our knowledge there is no nontrivial theorem in Kolmogorov theory that implies a result in bounded queries, or vice-versa.

4 Prehistory: (m, n)-computability

Frequency computations are a precursor to bounded queries. This work is not well known since most of the papers in the field are either in Russian, badly written, or both. A recent survey [28] gives a nice summary of the area. We define some of the basic terms of the field, and show that some results in it can be derived from results in bounded queries, while others (probably) cannot.

Definition 4.1 [45] A set A is (m, n)-computable if there exists a recursive function $f : \mathbb{N}^n \to \{0, 1\}^n$ such that for all pairwise distinct tuples $(x_1, \ldots, x_n)$, if $f(x_1, \ldots, x_n) = \langle b_1, \ldots, b_n \rangle$ then $|\{i : \chi_A(x_i) = b_i\}| \geq m$.

Myhill (see [37] P. 393) asked if making m close to n forces A to be recursive. Trahtenbrot [50] answered YES by showing that if

$2m > n$ then A is recursive. Kinber [30, 31] solved several variations of the problem (see [28]).

The techniques used by Trahtenbrot and Kinber are weak forms of techniques used by Owings[1] and Kummer in proving Theorems 7.1, 7.2, and 7.3 (the cardinality theorem). Consequently, everything proven by Trahtenbrot and Kinber can be derived from the cardinality theorem. We give one example: Trahtenbrot's theorem above. The interested reader should see [28] for statements and proofs of other theorems about (m, n)-computability and their relation to the cardinality theorem and various subcases of it.

Theorem 4.2 *[50] If A is (m, n)-computable, and $2m > n$, then A is recursive.*

Proof:

We show that if A is (m, n) computable via f, and $2m > n$, then $\#_n^A \in FQ(\lceil \log n \rceil, X)$; hence by the cardinality theorem A is recursive.

Let $f(x_1, \ldots, x_n) = b_1 \cdots b_n$. Since $2m > n$ one of the following happens.

(*a*) $|\{i : b_i = 0\}| \geq \frac{n}{2}$, hence one of the i with $b_i = 0$ is correct, so $\#_n^A(x_1, \ldots, x_n) \neq n$.

(*b*) $|\{i : b_i = 1\}| \geq \frac{n}{2}$, hence one of the i with $b_i = 1$ is correct, so $\#_n^A(x_1, \ldots, x_n) \neq 0$.

In both cases the range of values of $\#_n^A(x_1, \ldots, x_n)$ is reduced from $n + 1$ to n. Let X be the union of the following two sets.

$\{\langle x_1, .., x_n, i\rangle \mid (a)$ occurs and ith bit of $\#_n^A(x_1, .., x_n)$ is 1$\}$

$\{\langle x_1, .., x_n, i\rangle \mid (b)$ occurs and ith bit of $\#_n^A(x_1, .., x_n) - 1$ is 1$\}$

It is easy to see that $\#_n^A \in FQ(\lceil \log n \rceil, X)$. ∎

[1] It should be noted that Owings did not know of the papers of Trahtenbrot and Kinber when he proved Theorems 7.1 and 7.2.

Another question of interest in this field is 'For which m, n, r, s is it the case that all (m, n)-computable sets are (r, s)-computable?' The following results, which are inroads on this question, do not seem to be derivable from bounded queries. The general question is still open.

Theorem 4.3 *[18] Let n, m, r, s be such that $n \geq 2m$, $s \geq 2r$, and $n \geq s$. All (m, n)-computable sets are (r, s)-computable iff $n - m \leq s - r$.*

Theorem 4.4 *[18] For every $n \geq 2$ there is a $(1, n)$-computable set which is not $(2, n+1)$-computable.*

It is an open problem to define the notion of (m, n)-computable functions in a complexity-theoretic domain and prove something interesting about it. It is not clear how interesting this would be.

5 How hard is membership?

We examine the complexity of computing $F_m^A(x_1, \ldots, x_m) = \langle \chi_A(x_1), \ldots, \chi_A(x_m)\rangle$. It is clear that $F_m^A \in FQ_{||}(m, A)$. For which sets A can we compute F_m^A with fewer serial queries to A? To some X? For $A = K$ we can do very well:

Theorem 5.1 *[15, 16] For all n, $F_{2^n-1}^K \in FQ(n, K)$.*

Can this be improved upon? The next theorem says NO in a very strong way.

Theorem 5.2 *[15] For all sets A, if there exists a set X and a number n such that $F_{2^n}^A \in FQ(n, X)$, then A is recursive.*

Are there any other natural sets for which we can obtain $F_m^A \in FQ(m-1, A)$? Or for which there is an X with $F_m^A \in FQ(m-1, X)$? We shall see that answer is no. Recall [48] that if A is a set then A' is the halting set relative to A. Virtually all natural sets in recursion theory, including the Σ_i-complete and Π_i-complete sets, are of the form A' for some A.

Theorem 5.3 *[15] If A is nonrecursive then for all m and X $F_m^{A'} \notin FQ(m-1, X)$.*

How does the Turing degree of a set A relate to the number of queries needed to compute F_m^A? The next three theorems show that within a Turing degree a variety of behaviors are possible.

Theorem 5.4 *[15] If $\mathbf{a}$ is any nonrecursive Turing degree then there exist sets $A, B \in \mathbf{a}$, and a set Y, such that*

- *For all n, $F_{2^n-1}^A \in FQ(n, A)$.*
- *For all n, $F_{2^n-1}^A \in FQ_{||}(n, Y)$.*
- *For all m, and all X, $F_m^B \notin FQ(m-1, X)$.*

The second item cannot be improved (see Theorem 6.1).

Our next result is about r.e. degrees. For every r.e. set A, for all n, $F_{2^n-1}^A \in FQ(n, K)$. Hence a result like '$F_m^A \notin FQ(m-1, X)$' is not possible.

Theorem 5.5 *[15] If $\mathbf{a}$ is any r.e. Turing degree then there exist r.e. sets $A, B \in \mathbf{a}$ such that*

- *For all n, $F_{2^n-1}^A \in FQ(n, A)$.*
- *For all m, $F_m^B \notin FQ(m-1, B)$.*

The results stated so far have a 'feast or famine' flavor– either F_m^A is very hard or very easy. The next theorem partially explains this.

Theorem 5.6 *[4][2] Let A be any set. If there exist c, k and X such that $F_k^A \in FQ(k-1, X)$ then there exist m_0 and Y such that*

$$(\forall m \geq m_0)[F_m^A \in FQ_{||}((k-2)\log m + c, Y)].$$

Can Theorem 5.6 be improved to (say) $F_m^A \in FQ_{||}(17 \log m, Y)$? The next theorem shows that the answer is no– for every k there are sets such that (roughly) $F_m^A \in FQ(k \log m, A)$ but $(\forall X) F_m^A \notin FQ((k-1)\log m, X)$. The proof is in the appendix. The theorem is due to Beigel and Gasarch and was proven in 1988, though never published.

Theorem 5.7 *If $\mathbf{a}$ is any Turing degree above (or equal to) $\mathbf{K}$, then for every $k \in \mathbb{N}$ there exists a set $A \in \mathbf{a}$ such that*

- $(\forall m)[F_m^A \in FQ(k \lceil \log \frac{m}{k} + 1 \rceil, A)]$.
- $(\forall m)(\forall X)[F_m^A \notin FQ((k-1) \lfloor \log \frac{m}{k} \rfloor, X)]$.

Much less is known about how many queries to A are required to compute F_m^A. The following theorem is all that is known.

Theorem 5.8 *[5][3] Let A be any set. If there exists k such that $F_k^A \in FQ(k-1, A)$ then there exist m_0 and $r < 1$ such that*

$$(\forall m \geq m_0)[F_m^A \in FQ(m^r, A)].$$

It is an open problem to improve this result.

6 How hard is membership? (A second look)

The algorithm for $F_{2^n-1}^K \in FQ(n, K)$ has two features that we wish to examine:

[2] In [4] this theorem was proven in a complexity-theory framework, but the proof is the same for our recursion-theoretic framework.

[3] In [5] this theorem was proven in a complexity-theory framework, but the proof is the same for our recursion-theoretic framework.

1. The queries are made serially.

2. If incorrect answers are supplied then the computation must diverge.

We show that if either of these luxuries are disallowed, then, for all m, F_m^K requires m queries to K.

Examining Luxury 1:

Theorem 6.1 *[7] If A is any nonrecursive set then, for all m, $F_m^A \notin FQ_{||}(m-1, A)$*

Proof:
If $F_m^A \in FQ_{||}(m-1, A)$ then, by induction, for all i, $F_{m+i}^A \in FQ_{||}(m-1, A)$. This contradicts Theorem 5.2.
∎

Corollary 6.2 *$(\forall m) F_m^K \notin FQ_{||}(m-1, K)$.*

If parallel queries to a different oracle are allowed then a large savings occurs: there exists a set Y such that for all m, $F_m^K \in FQ_{||}(\lceil \log(m+1) \rceil, Y)$.

Examining Luxury 2:

Definition 6.3 Let $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$. A function f is in $FQC(n, A)$ if there exists an oracle Turing machine $M^{()}$ such that

1. M^X computes f.

2. For all oracles Y, for all x, $M^Y(x)$ makes at most m queries and halts.

Beigel and Gasarch proved the following theorem in 1987, though it appears here for the first time.

Theorem 6.4 *For all m and X, $F_m^K \notin FQC(m-1, X)$.*

Proof:
Assume $F_m^K \in FQC(m-1, X)$ via $M^{()}$. We construct programs $x_1, \ldots, x_m$ such that $F_m^K(x_1, \ldots, x_m) \neq M^X(x_1, \ldots, x_m)$. Our construction of $x_1, \ldots, x_m$ uses the m-ary recursion theorem [44, 47] so program x_i knows the programs $x_1, \ldots, x_m$.

Our algorithm for x_i is as follows. On any input, x_i runs $M^{()}(x_1, \ldots, x_m)$ using all 2^{m-1} possible sequences of answers. (All these computations halt by the assumptions about $M^{()}$). Let the outputs be $w_1, \ldots, w_{2^{m-1}} \in \{0,1\}^m$. Let w be the least element of $\{0,1\}^m$ that is not in $\{w_1, \ldots, w_{2^{m-1}}\}$. If the ith bit of w is 0 then x_i diverges, else x_i converges.

For all x_i the same w is found. The x_i's conspire to make $F_m^K(x_1, \ldots, x_m) = w$. But w is not a possible output for $M^{()}(x_1, \ldots, x_m)$. In particular $M^X(x_1, \ldots, x_m) \neq w$. Hence $F_m^A(x_1, \ldots, x_m) \neq M^X(x_1, \ldots, x_m)$.
∎

Corollary 6.5 *If A is such that $K \leq_m A$ then for all m and X, $F_m^A \notin FQC(m-1, X)$.*

All the lower bounds, and most of the upper bounds in this paper would hold if Q and FQ were replaced with QC and FQC. In fact, all the theorems in this paper are true for QC and FQC with the same prove, except for Theorem 5.1 (which is FALSE, as shown in this section), and Theorem 10.2 (which has not been investigated along these lines). It might be of some interest to see where else these two notions differ.

7 How hard is cardinality?

We examine the complexity of computing $\#_m^A(x_1, \ldots, x_m) = |\{i : x_i \in A\}|$. This function looks easier to compute than $F_m^A(x_1, \ldots, x_m)$; nevertheless, Beigel conjectured that for all n, A, and X, $\#_{2^n}^A$ cannot

be computed with only n queries to X. This conjecture turned out to be interesting and hard; it has recently been answered affirmatively. Owings took the first steps to a solution by proving the following two theorems.

Theorem 7.1 *[42] For all A, n and X, if $F_{2^n}^A \in FQ(n, X)$ then $A \leq_{\mathrm{T}} K$.*

Theorem 7.2 *[42] For all A and X, if $F_2^A \in FQ(1, X)$ then A is recursive.*

Kummer [33] built on the latter theorem, in an intricate way, to obtain a beautiful proof of Beigel's conjecture. The proof used a version of Ramsey's Theorem.

Theorem 7.3 *[33] For all A and X, if $F_{2^n}^A \in FQ(n, X)$ then A is recursive.*

We now examine what behavior $\#_m^A$ might have within a degree. We do not get the same variety of behavior we obtain for F_m^A because of the following theorem.

Theorem 7.4 *[9] For every set A and $n \in \mathbb{N}$ there exists X such that $\#_{2^n-1}^A \in FQ_{||}(n, X)$.*

Proof: Let $X = A^{\mathrm{tt}}$. The value of $\#_{2^n-1}^A(x_1, \ldots, x_{2^n-1})$ can be found with n parallel queries to X since every bit of $\#_{2^n-1}^A(x_1, \ldots, x_{2^n-1})$ can be expressed as a boolean combination of atoms of the form '$x \in A$.' ∎

We still get some variety of behavior by looking at how hard it is to compute $\#_m^A$ using queries to A.

Theorem 7.5 *[9] If $\mathbf{a}$ is any Turing degree then there exists a set A in $\mathbf{a}$ such that $\#_{2^n-1}^A \in FQ_{||}(n, A)$.*

Proof: Let A be any set of the form B^{tt} where $B \in \mathbf{a}$. Then use the technique of Theorem 7.4. ∎

Theorem 7.6 *[15] If $\mathbf{a}$ is any Turing degree above (or equal to) $\mathbf{K}$, then then there exists a set $A \in \mathbf{a}$ such that for all m, $\#_m^A \notin FQ(m-1, A)$.*

Theorem 7.7 *[15] If $\mathbf{a}$ is any r.e. Turing degree then there exists an r.e. set $A \in \mathbf{a}$ such that for all m, $\#_m^A \notin FQ(m-1, A)$.*

It is an open question to prove Theorem 7.6 for any nonrecursive Turing degree.

8 How hard is verification?

We examine the complexity of deciding the set $V_m^A = \{\langle x_1, .., x_m, b_1, .., b_m\rangle : (\forall i)\chi_A(x_i) = b_i\}$, in terms of queries to A. Unlike F_m^K and $\#_m^K$ there is (almost) no limit to savings on queries.

Theorem 8.1 *[24] For all m, $V_m^K \in Q_{||}(2, K)$ but $V_2^K \notin Q(1, K)$.*

Are there nonrecursive sets A such that, for all m, $V_m^A \in Q(1, A)$? YES! In fact, within any Turing degree such a set can be found. More generally, as the next three theorems show, within most Turing degrees a variety of behaviors occurs.

Theorem 8.2 *[24] If $\mathbf{a}$ is any nonrecursive Turing degree then there exist $A, B \in \mathbf{a}$ such that*

1. *for all m, $V_m^A \in Q(1, A)$;*
2. *for all m, $V_m^B \notin Q(m-1, B)$.*

Theorem 8.3 *[24] If $\mathbf{a}$ is any nonrecursive r.e. Turing degree then there exist r.e. sets A and B such that*

1. *for all m, $V_m^A \in Q_{||}(2, A)$;*

2. *for all* m, $V_m^B \notin Q(m-1, B)$.

Theorem 8.4 *[24] If* **a** *is any Turing degree above (or equal to)* **K** *then, for all* i, *there exists* $A \in \mathbf{a}$ *such that*

1. *for all* m, $V_m^A \in Q_{||}(i, A)$;
2. $V_i^A \notin Q(i-1, A)$.

It is an open question whether Theorem 8.4 holds for any nonrecursive Turing degree.

9 Do extra queries help to compute functions?

We show that in the case of computing functions, extra queries always help.

Theorem 9.1 *[7] For all nonrecursive sets* A, *for all* $i \in \mathsf{N}$, $FQ(i, A) \subset FQ(i+1, A)$.

Proof:
Since $F_i^A \in FQ(i, A)$ but $F_{2^i}^A \notin FQ(i, A)$ (by Theorem 5.2), there exists j such that $F_j^A \in FQ(i, A)$ but $F_{j+1}^A \notin FQ(i, A)$. It is easy to see that $F_{j+1}^A \in FQ(i+1, A) - FQ(i, A)$.
∎

By Theorem 6.1 we have

Theorem 9.2 *[7] For all nonrecursive sets* A, *for all* $i \in \mathsf{N}$, $FQ_{||}(i, A) \subset FQ_{||}(i+1, A)$.

10 Do extra queries to K help to decide sets?

We state a theorem which implies that additional queries to K do add to the power of an oracle Turing machine to decide sets. More generally, we show that the classes $D_n, Q(n, K)$, and $Q_{||}(n, K)$ interleave in a beautiful way, where the sets D_n form the difference hierarchy (to be defined).

The difference hierarchy was first studied in detail in [21], where the connection with the "k-trial predicates" of [43] was noted. The relation between the levels of the difference hierarchy and bounded truth-table reducibility with fixed norm was first noted in [44, Theorem 14.IX]. The n-r.e.sets and weakly n-r.e.sets were introduced in [19] and [20] respectively, where their Turing degrees were considered.

Definition 10.1 A set A is *n-r.e.* if there exists a total recursive 0-1 valued function $f(x, s)$ such that (1) $f(x, 0) = 0$, (2) for all x, $\lim_{s\to\infty} f(x, s) = \chi_A(x)$, and (3)$|\{s : f(x, s) \neq f(x, s+1)\}| \leq n$. We denote the class of n-r.e.sets by D_n. We denote $D_n \cap$ co-D_n by ∇_n.

In the theorem below, we use the following standard convention. If Φ is any class of sets (e.g., $Q(3, K)$) then $\mathbf{\Phi}$ is the set of Turing degrees that contain at least one set from Φ.

Theorem 10.2 *[16] The sets and degrees of the* $Q(\cdot, K)$ *and* $Q_{||}(\cdot, K)$ *hierarchies interleave with those of the difference hierarchy as follows:*

$D_1 \subset Q_{||}(1, K) = Q(1, K) = \nabla_2 \subset D_2 \subset Q_{||}(2, K) = \nabla_3 \subset D_3 \subset Q_{||}(3, K) = Q(2, K) = \nabla_4 \subset D_4 \subset Q_{||}(4, K) = \nabla_5 \subset \cdots \subset D_n \subset Q_{||}(n, K) = \nabla_{n+1} \subset D_{n+1} \subset \cdots \subset D_{2^n-2} \subset Q_{||}(2^n - 2, K) = \nabla_{2^n-1} \subset D_{2^n-1} \subset Q_{||}(2^n - 1, K) = Q(n, K) = \nabla_{2^n} \subset D_{2^n} \subset \cdots$

$\mathbf{D}_1 = \mathbf{Q}_{\|}(1, K) = \mathbf{Q}(1, K) = \mathbf{\nabla}_2 \subset \mathbf{D}_2 = \mathbf{Q}_{\|}(2, K) = \mathbf{\nabla}_3 \subset \mathbf{D}_3 = \mathbf{Q}_{\|}(3, K) = \mathbf{Q}(2, K) = \mathbf{\nabla}_4 \subset \mathbf{D}_4 = \mathbf{Q}_{\|}(4, K) = \mathbf{\nabla}_5 \subset \cdots \subset \mathbf{D}_n = \mathbf{Q}_{\|}(n, K) = \mathbf{\nabla}_{n+1} \subset \mathbf{D}_{n+1} \subset \cdots \subset \mathbf{D}_{2^n-2} = \mathbf{Q}_{\|}(2^n - 2, K) = \mathbf{\nabla}_{2^n-1} \subset \mathbf{D}_{2^n-1} = \mathbf{Q}_{\|}(2^n - 1, K) = \mathbf{Q}(n, K) = \mathbf{\nabla}_{2^n} \subset \mathbf{D}_{2^n} \subset \cdots$

11 Do extra parallel queries to A help to decide sets?

In Section 9 we showed that for every nonrecursive set A, $(\forall i)[FQ(i,A) \subset FQ(i+1,A)]$ and $(\forall i)[FQ_{||}(i,A) \subset FQ_{||}(i+1,A)]$. These results *do not* hold for $Q(i,A)$ and $Q_{||}(i,A)$ for general A. We prove this for $Q_{||}(i,A)$ in this section, and for $Q(i,A)$ in the next section. Both proofs are included because they are interesting and very different.

Definition 11.1 If B is any set then PARITY^B is the set of all tuples $\langle x_1, \ldots, x_m \rangle$ such that $|B \cap \{x_1, \ldots, x_m\}|$ is odd. (A pairing function could be defined such that the different arities do not conflict.)

Theorem 11.2 *[7] For all* $i \in \mathbb{N}$ $Q_{||}(i, \text{PARITY}^K) = Q(1, \text{PARITY}^K)$.

Proof:

Let $B \in Q_{||}(i, \text{PARITY}^K)$ via oracle Turing machine $M^{()}$. We show how to, given x, determine the value of $M^{\text{PARITY}^K}(x)$.

Run M^{PARITY^K} on x until the one parallel query '$x_1 \in \text{PARITY}^K?$,' '$x_2 \in \text{PARITY}^K?$,'..., '$x_i \in \text{PARITY}^K?$' is asked. Each x_j is itself a tuple of programs. Let the set of all programs in all the tuples be $\{y_1, \ldots, y_m\}$.

We create programs $z_0, z_1, \ldots, z_m$ and denote the output of these programs by $b_0, b_1, \ldots, b_m$ (the b_i are either 0,1, or $\uparrow$). The intention is that b_0 is an approximation to $M^{\text{PARITY}^K}(x)$, and, for all $i \geq 1$, b_i will be 1 to signal that the most recent approximation is wrong (the other values signal no change). When $b_i = 1$ we say that a *mindchange* has occurred. The key will be that the parity of the number of mindchanges will give us all the information we need.

z_0 operates as follows. Dovetail the two computations below:

a) Enumerate K looking for any of $y_1, \ldots, y_m$. Whenever any are found, restart the simulation in *b*) using this information.

b) Run $M^{\text{PARITY}^K}(x)$ under the assumption that the only elements in $\{y_1, \ldots, y_m\} \cap K$ are those found in *a*). This information determines answers to all the queries, though they may be incorrect. If this halts with a 0-1 output, then print that output and halt.

Note that z_0 will halt.

z_{i+1} operates as follows. Initially run z_i (this may diverge). If z_i converges then let m' be how many elements of $\{y_1, \ldots, y_m\} \cap K$ that z_i thinks are in K, and let b be what x_i thinks $M^{\text{PARITY}^K}(x)$ is. If $m' = m$ then output 0. If $m' < m$ then z_{i+1} enumerates K looking for $m'+1$ elements of $\{y_1, \ldots, y_m\}$. If such are found then dovetail the following computations.

a) Enumerate K looking for any of $y_1, \ldots, y_m$ that have not already been found. Whenever any are found, restart the simulation in *b*) using this information.

b) Run $M^{\text{PARITY}^K}(x)$ under the assumption that the only elements in $\{y_1, \ldots, y_m\} \cap K$ are those found in *a*), and before the simulation started. This information determines answers to all the queries, though they may be incorrect. If this halts with a 0-1 output, then if the output is b output 0 (to signal no mindchange), else output 1 (to signal a mindchange).

Note that the value of b passed to the z_{i+2} computation is *not* the value that z_{i+2} outputs; it is value that z_{i+1} found the simulation of $M^{\text{PARITY}^K}(x)$ to have.

For $i \geq 1$ let b_i be the output of z_i if it exists, and $\uparrow$ otherwise. It is easy to see that $|\{b_i : b_i = 1\}|$ is the number of mindchanges from z_0 that give the correct answer.

For $i \geq 1$ let z_i' be the machine that runs z_i and (1) if its output is 1 then converge; (2) if its output is 0 then diverge; (3) (by default) if the z_i diverges then diverge.

Let b_0 be the output of z_0. It is easy to see that

$$M^{\text{PARITY}^K}(x) = b_0 \oplus \text{PARITY}^K(z_1', \ldots, z_m').$$

Hence $M^{\text{PARITY}^K}(x)$ can be computed with one query to PARITY^K.

∎

12 Do extra serial queries to A help to decide sets?

We improve the result of the last section by showing that there exists a nonrecursive set A such that for all i, $Q(i, A) = Q(1, A)$. The set A is already in the recursion-theoretic literature. Definition 12.1 and Lemma 12.2 are from [38] but the interested reader may want to consult [40].

Definition 12.1 A Turing degree $\mathbf{a}$ is *hyperimmune-free* if for all $A \in \mathbf{a}$, for all $f \leq_{\text{T}} A$, there exists a recursive g such that $(\forall x)[f(x) < g(x)]$. (This definition is equivalent to the classical one of a degree that has no hyperimmune sets.)

Lemma 12.2 *[38] Hyperimmune-free degrees exist.*

Lemma 12.3 [4] *If $B \leq_{\text{T}} A$ and A is hyperimmune-free then $B \leq_{\text{tt}} A$.*

[4] Odifreddi [41] credits this theorem to Martin with no reference.

Proof:

Let $M^{()}$ be an oracle Turing machine such that B is decided by M^A. Let $f(x)$ be the number of steps taken by $M^A(x)$. Since $f \leq_{\text{T}} A$, and A is hyperimmune-free, there exists a recursive g such that, for all x, $f(x) < g(x)$.

Let $N^{()}$ be the following oracle Turing machine: on input x, on all query paths, shut off the machine after $g(x)$ steps. It is easy to see that M^A and N^A decide the same language. Since $N^{()}$ halts on all query paths, by an observation of [39, 49] (also see [40, 44]) $B \leq_{\text{tt}} A$.

∎

Theorem 12.4 *[7] There exists a set A such that, for all B, if $B \leq_{\text{T}} A$ then $B \in Q(1, A)$.*

Proof: Let $\mathbf{a}$ be a hyperimmune-free degree. Let $C \in \mathbf{a}$ and let $A = C^{\text{tt}}$. Note that $A \in \mathbf{a}$. If $B \leq_{\text{T}} A$ then since A is in a hyperimmune-free degree, $B \leq_{\text{tt}} A$ by Lemma 12.3. Since $A = C^{\text{tt}}$ it is easy to see that $B \in Q(1, A)$. ∎

Other examples of sets A such that $(\forall i)[Q(i, A) = Q(1, A)]$ are known [7]. It is an open problem to find a nice classification for such sets, or to prove that no such classification exists (e.g., show that the notion is not definable in some language).

13 Classifying Problems

One of the motivations for developing the theory of bounded queries was to classify problems in recursion theory in a manner finer than the arithmetic hierarchy [26]. This point of view has been most effective for determining the complexity of finding the chromatic number of a recursive graph [12, 13].

Definition 13.1 A *recursive graph* is a graph whose vertex set and edge set are recursive.

Let e_1 and e_2 be such that M_{e_1} and M_{e_2} are total 0-1 valued Turing machines. Let $e = \langle e_1, e_2 \rangle$. The *recursive graph indexed by* e, denoted G_e, has vertex set decided by M_{e_1}, and edge set decided by M_{e_2}.

We examine the problem of finding the chromatic number of a recursive graph, given an index for that graph. We do not want to consider the problem of determining if a number is an index of a recursive graph. Hence we use a variation of the notion of a promise problem, as defined in [22, 23].

Definition 13.2 A *promise problem* is a set A, called the *promise* and a function f, called the *problem* that need only be defined on A. A *solution to* (A, f) is a function g that is an extension of f. The key point is that we care about the complexity of g, but we do not care what g does when the input is not in A. The promise problem (A, f) is in class $\mathcal{C}$ (e.g., $FQ(3, K)$) if there exists *some* solution to it in $\mathcal{C}$. The promise problem (A, f) is not in class $\mathcal{C}$ if *no* solution of it is in $\mathcal{C}$.

Definition 13.3 Let $c \geq 1$ be a constant. Let χ_c be the following promise problem. The *promise* is the set of e such that G_e exists and has chromatic number $\leq c$. The *problem* is to find the chromatic number of G_e.

We have sharp bounds for the complexity of χ_c.

Theorem 13.4
[12] $\chi_c \in FQ(\lceil \log(c+1) \rceil, K)$. *For any set* X, $\chi_c \notin FQ(\lceil \log(c+1) \rceil - 1, X)$. *For any set* A *such that* $K \not\leq_{\mathrm{T}} A$, $\chi_c \not\leq_{\mathrm{T}} A$.

The upper and lower bounds are tight in two ways: The number of queries cannot be reduced *no matter what oracle is used*, and the Turing degree of the oracle cannot be lowered *no matter how many queries are allowed.*

Many variations of this problem are in [12, 13]. These include looking at the unbounded case (no bound on the chromatic number), recursive chromatic number, parallel queries, and being allowed to ask queries 'for free' to a weaker oracle. Just as the bounded case uses binary search, and is optimal as such, the unbounded case uses unbounded search, and is optimal as such; in fact, this research inspired some work on unbounded search [8].

Several other graph parameters could be examined in this light. This has been carried out for finding the number of components of a recursive graph [27]. One area outside of graph theory has been studied in this light: well quasi ordering theory. The interested reader is directed to [14].

14 Comparisons to Complexity Theory

Several people have studied bounded queries in a polynomial-bounded framework. In this section we compare and contrast several results stated in this paper with results on the same theme in that framework.

Definition 14.1 Let $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$. A function f is in $\mathrm{FP}^{A[m]}$ if $f \leq^{\mathrm{p}}_{\mathrm{T}} A$ via an algorithm that make at most m queries. A function f is in $\mathrm{FP}^{||A[m]}$ if $f \leq^{p}_{\mathrm{tt}} A$ via an algorithm that makes at most m queries.

Definition 14.2 Let $m \in \mathsf{N}$ and $A \subseteq \mathsf{N}$. A set B is in $\mathrm{P}^{A[m]}$ if its characteristic function is in $\mathrm{FP}^{A[m]}$. $\mathrm{P}^{||A[m]}$ is defined similarly.

We list results from recursion theory and complexity theory in pairs, and discuss them. Some of the results are stated informally.

1) F^K_m versus F^{SAT}_m.

F_m^K can be computed with substantially less than m queries to K, but such is *not* the case for SAT:

(Theorem 5.1):

$(\forall n)[F_{2^n-1}^K \in FQ(n,K)]$

From [1]:

$[(\exists X, m)[F_m^{SAT} \in \mathrm{FP}^{X[m-1]}]] \Rightarrow \Sigma_2^p = \Pi_2^p$.

The often stated analogy between K and SAT breaks down here. The major difference is that from $\#_m^K(x_1,\ldots,x_m)$ one can (in time) find $F_m^K(x_1,\ldots,x_m)$, but it appears to be the case that from $\#_m^{SAT}(x_1,\ldots,x_m)$ one cannot (in polynomial time) find $F_m^{SAT}(x_1,\ldots,x_m)$. This is because, for any set A, the following two are equivalent (1) $F_m^A \leq_T^p \#_m^A$ (2) A is p-selective. (See [46] for a definition of p-selective; the equivalence stated here is easy). Hence, if F_m^{SAT} were easily computable from $\#_m^{SAT}$ then SAT would be p-selective which implies P=NP.

2) Limits on savings for F_m^A.

In recursion theory there is a limit on how much fewer than m queries are needed to compute F_m^A if A is nonrecursive. In complexity theory the situation is more complicated.

Theorem 5.2:

$[(\exists X, m)F_{2^m}^A \in FQ(m,X)] \Rightarrow A$ recursive

From [1]:

$[(\exists X, m)F_{2^m}^A \in \mathrm{FP}^{X[m]})] \Rightarrow A \in EL_2$ (where EL_i is the ith level of the extended low hierarchy: $A \in EL_i$ iff $\Sigma_i^A \subseteq \Sigma_{i-1}^{A,\mathrm{NP}}$ [3]).

$[(\exists X, m)F_m^A \in FQ(m-1,X)] \Rightarrow A \in \mathrm{P/poly}$

This cannot be improved to placing sets in P since there exists arbitrarily hard sets A such that $(\forall m)[F_m^A \in FQ(1,A)]$ [2].

The first three theorems say that if some savings can be achieved in the computation of F_m^A then A has to be 'easy.' The proof of the second and third theorems points to P/poly being a plausible analogy of recursive. In many proofs that a set is recursive (including the proof of Theorem 5.2) some nonconstructive finite information is coded into the Turing machine; in many proofs that sets are in P/poly(including the second result stated above) some nonconstructive polynomial size information is needed for each n.

3) Behavior of F_m^A for various A in a degree.

Theorems 5.4 and 5.7 imply that within each Turing degree a wide variety of behaviors for the hardness of F_m^A are possible. This is not the case for polynomial-Turing degrees: within the poly-Turing degree of SAT we have the following.

From [1]:

For all sets $A \equiv_p^T SAT$,

$(\forall m, X)[F_m^A \notin \mathrm{FP}^{X[m-1]})$ unless $\Sigma_2^p = \Pi_2^p$.

There may be some polynomial-Turing degrees that contain sets that act very differently in terms of computing F_m^A with queries to some X. It is an open problem to find which types of pT-degrees have such sets.

4) Cardinality. There does not seem to be any analog of the cardinality theorem for complexity theory.

Theorem 7.3:

$(\exists m, X)[\#_{2^m}^A \in FQ(m,X)] \Rightarrow A$ is rec.

From [2]:

$(\exists A)[(\forall m)[\#_m^A \in \mathrm{P}^{A[1]}]$.

The set A is extremely sparse. A possible upper bound might be that if $\#_m^A$ can be computed 'easily' then A is in P/poly.

5) Verifying K versus verifying SAT.

Theorem 8.1:

$(\forall m \geq 2)[V_m^K \in Q_{||}(2,K) - Q(1,K)]$

From [24]: If PH does not collapse then

$(\forall m \geq 2)[V_m^{SAT} \in \mathrm{P}^{||SAT[2]} - \mathrm{P}^{SAT[1]}]$.

The second result can be stated more precisly: $V_2^{SAT} \in \mathrm{P}^{SAT[1]}$ iff $BH_{NP}[2] = \mathrm{P}^{SAT[1]}[24]$ ($BH_{NP}[2]$ is the second level of the boolean hierarchy, [17]). This consequence of $V_m^{SAT} \in \mathrm{P}^{SAT[1]}$ implies that $BH_{NP}[2] = \text{co-}BH_{NP}[2]$, which by [11] implies that $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[1]\oplus NP} = PH$.

The proofs of both upper bounds are similar and easy. The lower bounds were easier to obtain in recursion theory than in complexity theory.

6) Behavior of V_m^A for various A in a degree.

Theorems 8.2, 8.3, and 8.4 imply that within each Turing degree a wide variety of behaviors for the hardness of V_m^A is are possible. Not much is known for this problem in complexity theory.

7) Function Hierarchies.

Theorem 9.1:

$(\exists i)FQ(i,A) = FQ(i+1,A) \Rightarrow A$ recursive.

From [1]:

$(\exists i)\mathrm{FP}^{A[i]} = \mathrm{FP}^{A[i+1]} \Rightarrow A \in \mathrm{P/poly}$.

Much like subsection 2, we see that P/poly might be an analog for recursive.

8) $Q(i,K)$ versus $\mathrm{P}^{K[i]}$

The main theorem in Section 10 is an exact statement of how $Q(i,K), Q_{||}(i,K)$ and D_i relate. The same theorem holds with $Q(i,K)$ replaced by $\mathrm{P}^{SAT[i]}$, $Q_{||}(i,SAT)$ replaced by $\mathrm{P}^{||SAT[i]}$, and D_i replaced with $BH_{NP}(i)$.

Both the proof in recursion theory and the proof in complexity theory used the mind-change technique (see Theorem 11.2 for its use in recursion theory.) The proof in complexity theory was far harder since the mind-change technique is much subtler there. See [10, 51, 52] for an explanation of the mind-change technique in complexity theory.

9) Collapsing Set hierarchies.

Theorem 12.4 is that there exists a nonrecursive A such that

$$(\forall i)Q(i,A) = Q(1,A)$$

In [2] it is shown that there exists $A \notin \mathrm{P}$ such that

$$(\forall i)\mathrm{P}^{A[i]} = \mathrm{P}^{A[1]}.$$

There are two ways to obtain the first result, one of which was presented in this paper. Nothing is known about what type of sets A have the property $(\forall i)Q(1,A) = Q(i,A)$. The question of which types of sets have the property $(\forall i)\mathrm{P}^{A[i]} = \mathrm{P}^{A[1]}$ is better understood as the following is known:

If $\mathrm{P}^{A[i]} = \mathrm{P}^{A[1]}$ then there exists a sparse set S such that $A \leq_c^{NP,S} \overline{A}$ [36]

(The definition of $X \leq_c^{NP,S} Z$ is that there exists an NP oracle Turing machine $M^{()}$ such that $x \in X$ iff some path of $M^S(x)$ produces a string from Z.)

10) Classifying Problems.

In recursion theory bounded queries are used to classify functions from recursive graph theory (section 13). The usual method of classification in recursion theory, the arithmetic hierarchy, is not appropriate for two reasons: (1) it is used to classify sets, and (2) it is not fine enough.

In complexity theory Krentel [32] and Gasarch [25] have used bounded queries to used to classify functions that are optimization problems usually related to NP-complete problems. The usual method of classification in complexity theory, the polynomial hierarchy (and PSPACE), is not appropriate for two reasons: (1) it is used to classify sets, and (2) it is not fine enough.

In both recursion theory and complexity theory bounded queries have been useful as a measure of the difficulty of functions that is better suited to the task than prior methods.

15 Appendix

Proof of Theorem 5.7.

Proof:

We construct a set $A \in \mathbf{a}$ such that (1) $(\forall m)F_m^A \in FQ(k\lceil \log \frac{m}{k} + 1 \rceil, A)$, but also (2) $(\forall m)(\forall X)[F_m^A \notin FQ((k-1)\lfloor \log \frac{m}{k} \rfloor, X)]$. The first we achieve by constructing A to be the disjoint union of $k+1$ semirecursive sets (as defined by Jockusch [29]). One of the semirecursive sets is used to code a set $\tilde{A} \in \mathbf{a}$. The

second we achieve by diagonalization.

We need $(\forall m) F_m^A \in FQ(k\left\lceil \log \frac{m}{k}+1\right\rceil, A)$. We describe a type of set that will always have this property. Let $\Pi_1, \ldots, \Pi_k$ be a recursive partition of N. We view $\Pi_1, \ldots, \Pi_{k-1}$ as being isomorphic to the rationals. Let $\preceq_i$ be the ordering on Π_i viewed this way. We denote this ordering by $\preceq$ when i is clear.

Let $\tilde{A} \in \mathbf{a}$ be such that, for all n, $F_{2^n-1}^{\tilde{A}} \in FQ(n, \tilde{A})$ (such exists by Theorem 5.4). We construct A such that

1. For i, $1 \leq i \leq k-1$, $A \cap \Pi_i$ is closed downward under $\preceq_i$.
2. $A \cap \Pi_k$ is recursively isomorphic to $\tilde{A}$; $\overline{A} \cap \Pi_k$ is recursively isomorphic to $\mathsf{N} - \tilde{A}$. (We denote this fact by $A \cap \Pi_k \equiv \tilde{A}$.)

For any such A, for all m, $F_m^A \in FQ(k\left\lceil \log \frac{m}{k}+1\right\rceil, A)$. Given $\langle x_1, \ldots, x_m\rangle$, assume without loss of generality that

$x_1 \preceq_1 \cdots \preceq_1 x_{i_1} \in \Pi_1$
$x_{i_1+1} \preceq_2 \cdots \preceq_2 x_{i_2} \in \Pi_1$
$\vdots \quad \vdots \quad \vdots$
$x_{i_{k-2}+1} \preceq_{k-1} \cdots \preceq_{k-1} x_{i_{k-1}} \in \Pi_{k-1}$
$x_{i_{k-1}+1}, \ldots, x_{i_k} \in \Pi_k$ $(i_k = n)$.

By using binary search on the first $k-1$ groups, and $A \cap \Pi_k \equiv \tilde{A}$ on the last group, one can determine $F_m^A(x_1, \ldots, x_m)$ with $\sum_{j=1}^{k} \lceil \log(i_j+1)\rceil \leq k\left\lceil \log \frac{m}{k}+1\right\rceil$ queries to $\tilde{A}$.

Since $A \cap \Pi_k \equiv \tilde{A}$, we have $\tilde{A} \leq_{\mathrm{T}} A$.

We construct an A of this type that satisfies the following requirement:

$R_{\langle e,m\rangle}$: For all X
[if $(\forall x) M_e^X(x)$ makes $\leq (k-1)\lfloor \log \frac{m}{k}\rfloor$ queries then
M_e^X does not compute F_m^A].

The construction will be recursive in $\tilde{A} \oplus K \equiv_T \tilde{A}$, hence $A \leq_{\mathrm{T}} \tilde{A}$. Since $\tilde{A} \leq_{\mathrm{T}} A$, $A \in \mathbf{a}$.

The construction is carried out in stages.. At the end of stage $s+1$ we have parameters $L_1, U_1, \ldots, L_{k-1}, U_{k-1}$ such that for all i $L_i, U_i \in \Pi_i$ and $L_i \preceq_i U_i$. We define A such that for all i the elements of $\{x \mid x \prec_i L_i\}$ are in A and the elements of $\{x \mid U_i \prec x\}$ are not in A (and never will be), the elements between L_i and U_i are not yet determined at the end of stage $s+1$. Since $\preceq_i$ is dense the number of elements of Π_i that are not determined is infinite.

CONSTRUCTION

Stage 0: Set $A \cap \Pi_k \equiv \tilde{A}$. Set $L_1 = \cdots = L_{k-1} = -\infty$, and $U_1 = \cdots = U_{k-1} = \infty$.

Stage s+1: Let $s = \langle e, m\rangle$. We satisfy $R_{\langle e,m\rangle}$. Let $p = \left\lfloor \frac{m}{k-1} \right\rfloor$. Let $x_1, \ldots, x_m$ be picked such that, for i, $1 \leq i \leq k-1$, $L_i \prec x_{(i-1)p+1} \prec \cdots \prec x_{ip} \prec U_i$; and $L_{k-1} \prec x_{(k-2)p+1} \prec \cdots \prec x_m < U_{k-1}$. Run $M_e^{()}(x_1, \ldots, x_m)$ pursing all query paths. If a query path asks more than $(k-1)\lfloor \log \frac{m}{k}\rfloor$ queries or diverges then it can be ignored (oracle K is used to check this). Let $w_1, \ldots, w_N$ be all possible answers that are output. Note that $N \leq 2^{(k-1)\lfloor \log \frac{m}{k}\rfloor}$. Also note that by increasing L_i and-or decreasing U_i there are $(p+1)^{k-2}(m-(k-2)p+2)$ possible ways to determine the membership of $x_1, \ldots, x_m$. Since $2^{(k-1)\lfloor \log \frac{m}{k}\rfloor} < (p+1)^{k-2}(m-(k-2)p+2)$ there exists a way to adjust the L_i and U_i to make all possible answers incorrect. Adjust L_i and U_i as such. ∎

16 Acknowledgements

I would like to thank Richard Beigel for his many contributions to this field, including being one of the two founders. I would also like to thank Martin Kummer for coming up with the most interesting proof in this field (Theorem 7.3), which inspired me to write this

survey; and Jim Owings for explaining Kummer's proof to me; and Valentina Harizanov for translating and explaining the work of several Russian mathematicians on (m, n)-computability. I would also like to thank my proofreaders Dave Bagget, Richard Beigel, Gary Flake, Valentina Harizanov, Bill Regli, and Stephen Smith.

References

[1] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proc. of the 5th Annual Conference on Structure in Complexity Theory*, pages 232–243. IEEE Computer Society Press, July 1990. A much expanded version has been submitted to *Information and Computation*.

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

[3] J. L. Balcázar, R. V. Book, and U. Schöning. Sparse sets, lowness and highness. *SIAM J. Comput.*, 15(3):739–747, Aug. 1986.

[4] 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.

[5] R. Beigel. NP-hard sets are p-superterse unless R=NP. Technical Report 4, The Johns Hopkins University, Dept. of Computer Science, 1988.

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

[7] R. Beigel. When are $k + 1$ queries better than k? Technical Report 6, The Johns Hopkins University, Dept. of Computer Science, 1988.

[8] R. Beigel. Unbounded searching algorithms. *SIAM J. Comput.*, 19(3):522–537, June 1990.

[9] R. Beigel, 1991. Personal communication.

[10] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Comput. Sci.*, 84:199–223, 1991.

[11] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. Technical Report 91-1184, Cornell University, Dept. of Computer Science, 1991.

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

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

[14] R. Beigel and W. I. Gasarch. On the complexity of finding the obstruction set, 1989. manuscript.

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

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

[18] A. Degtev. (m, n)-computable sets. In D. Moldavanskii, editor, *Algebraic Systems*, pages 88–99. Wadsworth International, 1981. (in Russian).

[19] R. L. Epstein. *Degrees of Unsolvability: Structure and Theory*, volume 759 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1979.

[20] R. L. Epstein, R. Haas, and R. L. Kramer. Hierarchies of sets and degrees below $\mathbf{0}'$. In *Logic Year 1979–80*, volume 859 of *Lecture Notes in Mathematics*, pages 32–48, Berlin, 1981. Springer-Verlag.

[21] Y. L. Ershov. A hierarchy of sets, I. *Algebra i Logika*, 7(1):47–74, January–February 1968. English Translation, Consultants Bureau, NY, pp. 25–43.

[22] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[23] S. Even and Y. Yacobi. Cryptocomplexity and NP-completeness. In *Proc. 8th Colloq. on Automata, Languages, and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 195–207, Berlin, 1980. Springer-Verlag.

[24] W. Gasarch, L. Hemachandra, and A. Hoene. On checking and versus evaluation of multiple queries. *Information and Computation*, pages 72–93, 1993. Shorter version, appeared in 15th International Sym. of Mathematical Found. of Computer Science (MFCS '90), Banka Bystrica, Czechoslovakia.

[25] W. I. Gasarch. The complexity of optimization functions. Technical Report 1652, University of Maryland, Dept. of Computer Science, 1985.

[26] W. I. Gasarch. A hierarchy of functions with applications to recursive graph theory. Technical Report 1651, University of Maryland, Dept. of Computer Science, 1985.

[27] K. Guimaraes. *On the power of queries*. PhD thesis, Dept. of Computer Science - U. of Maryland at College Park, in preparation.

[28] V. Harizanov, M. Kummer, and J. Owings. Frequency computations and the cardinality theorem. *Journal of Symbolic Logic*, 57(2):682–687, June 1992.

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

[30] E. Kinber. Frequency calculations of general recursive predicates and frequency enumeration of sets. *Soviet Math Doklady*, 13:873–876, 1972.

[31] E. Kinber. On frequency-enumerable sets. *Algebra and Logic*, 13:226–237, 1974.

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

[33] M. Kummer. A proof of Beigel's cardinality conjecture. *Journal of Symbolic Logic*, 57(2):677–681, June 1992.

[34] M. Li and P. Vitanyi. Two decades of applied Kolmogorov complexity. In *Proc. of the 3rd Annual Conference on Structure in Complexity Theory*, pages 80–101. IEEE Computer Society Press, June 1988.

[35] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Addison-Wesley, 1991.

[36] A. Lozano. Bounded queries to arbitrary sets, 1991. Manuscript.

[37] R. McNaughton. The theory of automata, a survey. *Advances in Computers*, 2:379–421, 1962.

[38] W. Miller and D. A. Martin. The degree of hyperimmune sets. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 14:159–166, 1968.

[39] A. Nerode. General topology and partial recursive functionals. In *Cornell Summer Institute in Symbolic Logic*, pages 247–251, 1957.

[40] P. Odifreddi. *Classical Recursion Theory (Volume II)*. North-Holland, Amsterdam. To Appear.

[41] P. Odifreddi. *Classical Recursion Theory (Volume I)*. North-Holland, Amsterdam, 1989.

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

[43] H. Putnam. Trial and error predicates and the solution to a problem of Mostowski. *Journal of Symbolic Logic*, 30(1):49–57, Mar. 1965.

[44] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967.

[45] G. Rose. An extended notion of computability. In *Abstr. International Congress for Logic, Methodology, and Philosophy of Science*, page 14, 1960.

[46] 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.

[47] R. Smullyan. *Theory of Formal Systems*. Princeton University Press, Princeton, New Jersey, 1961. Annals of Mathematical Studies Vol 47.

[48] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.

[49] B. A. Trakhtenbrot. Tabular representation of recursive operators. *Doklady Academy Nauk, SSSR*, 101:417–420, 1955.

[50] B. A. Trakhtenbrot. On the frequency computation of functions. *Algebra i Logika*, 2:25–32, 1964. (in Russian).

[51] K. W. Wagner. Bound query computations. In *Proc. of the 3rd Annual Conference on Structure in Complexity Theory*, pages 260–277. IEEE Computer Society Press, June 1988.

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