When are $k + 1$ Queries Better Than k?

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

January 4, 1995

Abstract

Let A be any nonrecursive set. For each $k \geq 0$, we show that there are functions $(k+1)$-truth-table reducible to A that are not k-weak-truth-table reducible to A; and we show that there are functions $(k+1)$-Turing reducible to A that are not k-Turing reducible to A. Intuitively, this means that $k+1$ queries to a nonrecursive oracle are strictly more useful than k queries to the same oracle (for the sake of computing functions), and that $k+1$ oracle answers from a nonrecursive oracle provide information that cannot be provided by any k answers from the same oracle.

The situation is different for decision problems. We say that a set A is *parallel-supportive* if, for all k, there is a language that is $(k+1)$-wtt reducible to A but not k-wtt reducible to A. A set A is *supportive* if, for all k, there is a language that is $(k+1)$-Turing reducible to A but not k-Turing reducible to A. We show that every degree above $\mathbf{0}'$ contains a set that is supportive but not parallel-supportive. We show that every hyperimmune-free degree contains a set that is neither supportive nor parallel-supportive. We also construct a subclass of the hyperimmune degrees with the same property.

*Supported by NSF grant CCR-8808949 and by a fellowship from the Fannie and John Hertz Foundation. Parts of this work were completed while the author was a student at Stanford University.

On the other hand we show that parallel-supportive and supportive sets exist in abundance: all jumps of sets, all generic sets, all nonrecursive semi-recursive sets, all nonrecursive r.e. sets, all nonrecursive n-r.e. sets, and almost all sets are supportive and parallel-supportive. In particular every nonrecursive Turing degree contains a set that is parallel-supportive and supportive. We also show that all nonrecursive nonsuperterse sets are supportive.

1. Introduction

Oracle computations allow us to formalize the notion that one problem is computationally more difficult than another. In much previous work [Rog67, HU79, Soa87], the set B has been said to be more computationally difficult than the set A if A is decidable by a Turing machine with an oracle for B, but B is not decidable by any Turing machine with an oracle for A. That notion is essentially qualitative.

In this paper, we adopt a quantitative notion of when one problem is computationally more difficult than another. We fix a set C and we assume that our Turing machines are equipped with an oracle for C. We say that the set B is computationally more difficult than the set A if A is decidable by a Turing machine that makes only n queries to the oracle for C for some constant n, but B is not decidable by any Turing machine that makes only n queries to the oracle for C.

Throughout this paper we examine the following question: "When does the ability to ask $n+1$ queries to an oracle for A allow us to solve harder problems than we could solve with only n queries?" This general question admits several variants, depending on whether we are bounding the number of queries made in a truth-table reduction or in a Turing-reduction and on whether we are computing sets or computing functions.

2. Preliminaries

Much of the material in this paper is also derived in [Bei87a, Bei87b, Bei87d, BG87], sometimes by different techniques.

Various kinds of reducibilities have been defined in the literature [Rog67, Soa87]. We present some of the well-studied reducibilities, along with some that have received attention only recently. Throughout this paper, we will consider a Turing machine with a random-access oracle as described in [HU79]. We write $A(x) = 1$ if $x \in A$, 0 if $x \notin A$.

Definition 1

- A is *Turing reducible* to B (denoted $A \leq_{\mathrm{T}} B$) if the characteristic function of A can be computed by an oracle Turing machine using oracle B.

- A is *truth-table reducible* to B (denoted $A \leq_{\mathrm{tt}} B$) if there is a total recursive function q and a *total* recursive predicate H such that

  $$q(x) = \langle k, q_1, \ldots, q_k \rangle, \text{ and}$$
  $$A(x) = H(x, B(q_1), \ldots, B(q_k)).$$

- A is *weak-truth-table* reducible to B (denoted $A \leq_{\mathrm{wtt}} B$) if there is a total recursive function q and a *partial* recursive predicate H such that

  $$q(x) = \langle k, q_1, \ldots, q_k \rangle, \text{ and}$$
  $$A(x) = H(x, B(q_1), \ldots, B(q_k)).$$

  (Note that H may fail to converge only if its input is invalid.)

- For any of the reductions r above, A is k-r reducible to B (denoted $A \leq_{k\text{-}r} B$) if $A \leq_r B$ via a machine M where the number of oracle queries made by M on input x is bounded by the constant k.

- A is *bounded-r* reducible to B (denoted $A \leq_{\mathrm{br}} B$) if $A \leq_{k\text{-}r} B$ for some k.

Note that all of the definitions above make sense if either or both of the sets A, B is replaced by a partial function. Soare says that weak-truth-table reducibility is sometimes called bounded-Turing reducibility; *we do not adopt that convention.* In Section 5.1., we construct sets A and B such that $A \leq_{2\text{-}T} B$, but $A \not\leq_{\mathrm{wtt}} B$; thus bounded-Turing reducibility is distinct from weak-truth-table reducibility.

We define a variant of k-Turing reducibility:

Definition 2 A is *k-strong-Turing reducible* to B if A is Turing reducible to B via an oracle Turing machine M that computes a total function for all oracles.

Thus k-strong-Turing reducibility is to k-Turing reducibility as k-truth-table reducibility is to k-weak-truth-table reducibility. By a well-known theorem of Nerode [Rog67, Theorem 9-XIX], strong-Turing reducibility (without a bound on the number of queries) is the same as truth-table reducibility.

If queries are made nonadaptively as in weak-truth-table reducibility, we call them *parallel queries*. If the queries are made adaptively as in Turing reducibility we call them *serial queries*.

In [BGH89], we defined the Q and FQ notation for bounded query classes. The following definitions are equivalent to definitions in those papers.

Definition 3

- $\mathrm{FQ}(n, A)$ is the class of all partial functions f such that $f \leq_{n\text{-T}} A$.
- $\mathrm{FQ}_{||}(n, A)$ is the class of all partial functions f such that $f \leq_{n\text{-wtt}} A$.

Note that the definitions above make sense if the set A is replaced by a partial function. If a machine M computes an n-Turing reduction to an oracle A then we call M an $\mathrm{FQ}(n, A)$ *machine*, and so on for the other bounded query classes.

Given a set A we define a related function, F_n^A, which is a convenient notation for the results of n parallel queries to A.

Definition 4

$$\mathrm{F}_n^A(x_1, \ldots, x_n) = (A(x_1), \ldots, A(x_n)).$$

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

Definition 5

- If f_1 and f_2 are two functions, then

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

- If C_1 and C_2 are two sets of functions then

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

Note that the $\parallel$ operator is associative.

3. Computing Functions

In [BGGO93], we considered the question, when is A terse, *i.e.*, when is it true that for all n,

$$\mathrm{FQ}_{||}(n+1, A) \nsubseteq \mathrm{FQ}(n, A)?$$

We showed that almost all sets are terse, but every nonrecursive truth-table degree contains a set that is not terse.

In this section we consider the question, when is there a total function in $\mathrm{FQ}(n+1, A) - \mathrm{FQ}(n, A)$? We also consider the question, when is there a total function in $\mathrm{FQ}_{||}(n+1, A) - \mathrm{FQ}_{||}(n, A)$? We show that such a function exists whenever A is nonrecursive.

3.1. The Nonspeedup Theorem

The following result from [BGGO93] is the key to showing that $n+1$ queries to a nonrecursive set are better than n for computing total functions:

Theorem 6 (Nonspeedup) *If A is a nonrecursive set and B is any set then*

$$\mathrm{F}^A_{2^n} \notin \mathrm{FQ}(n, B).$$

3.2. $n+1$ Queries to a Nonrecursive Set are Better than n

We will show that $n+1$ queries to a nonrecursive set allow us to compute functions that are not computable by a machine that makes only n queries to the same set (a precise statement of the result can be found below). While this result is to be expected and seems "intuitively obvious," the proof depends on a nontrivial result, the Nonspeedup Theorem.

Assume that A is nonrecursive. If it were the case that $\mathrm{F}^A_{n+1} \notin \mathrm{FQ}(n, A)$ then F^A_{n+1} would be a total function in $\mathrm{FQ}(n+1, A) - \mathrm{FQ}(n, A)$. If it were the case that $\mathrm{F}^A_{2^n-1} \in \mathrm{FQ}(n, A)$ then $\mathrm{F}^A_{2^n}$ would be a total function in $\mathrm{FQ}(n+1, A) - \mathrm{FQ}(n, A)$ by the Nonspeedup theorem. Those two cases are the extremes, which suggest the general rule. By the Nonspeedup Theorem, for every nonrecursive set A and every natural number n, there is a largest number u such that $\mathrm{F}^A_u \in \mathrm{FQ}(n, A)$. Then F^A_{u+1} is a total function belonging to $\mathrm{FQ}(n+1, A) - \mathrm{FQ}(n, A)$. We formalize this proof below.

Lemma 7 *Let A be any nonrecursive set. For all n,*

i. *there is a total function that is $(n+1)$-Turing reducible to A but not n-Turing reducible to A,*

ii. *there is a total function that is $(n+1)$-strong-Turing reducible to A but not n-strong-Turing reducible to A, and*

iii. *there is a total function that is $(n+1)$-wtt reducible to A but not n-wtt reducible to A.*

Proof:

i. Let A be a nonrecursive set, and let n be any natural number. Let

$$u = \max\{t : \mathrm{F}^A_t \in \mathrm{FQ}(n, A)\}.$$

By the Nonspeedup Theorem, the maximum exists (in fact it is less than 2^n), so u is well defined.

$$\begin{aligned} \mathrm{F}^A_{u+1} &\in \mathrm{FQ}(1, \mathrm{F}^A_u) \parallel \mathrm{FQ}(1, A) \\ &\subseteq \mathrm{FQ}(n, A) \parallel \mathrm{FQ}(1, A) \qquad \text{because } \mathrm{F}^A_u \in \mathrm{FQ}(n, A) \\ &\subseteq \mathrm{FQ}(n+1, A). \end{aligned}$$

Since u was chosen as the maximum t such that $\mathrm{F}^A_t \in \mathrm{FQ}(n, A)$, it follows that

$$\mathrm{F}^A_{u+1} \notin \mathrm{FQ}(n, A).$$

Therefore,

$$\mathrm{F}^A_{u+1} \in \mathrm{FQ}(n+1, A) - \mathrm{FQ}(n, A).$$

ii. Let u be the largest t such that F_t^A is n-strong-Turing reducible to A, and proceed as in part (i).

iii. Define

$$u = \max\{t : \mathrm{F}_t^A \in \mathrm{FQ}_{||}(n, A)\},$$

and proceed as in part (i).

∎

Lemma 8 *Let A be nonrecursive. For every n there is a total function that is $(n+1)$-tt reducible to A but not n-wtt reducible to A.*

Proof: By part (iii) of the preceding lemma, $\mathrm{FQ}_{||}(n, A) \subset \mathrm{FQ}_{||}(n+1, A)$. Therefore F_{n+1}^A is not n-wtt reducible to A; of course F_{n+1}^A is $(n+1)$-tt reducible to A. ∎

Thus we have the following theorem:

Theorem 9 *The following statements are equivalent:*

i. A is nonrecursive.

ii. For every n, there is a total function $(n+1)$-Turing reducible to A that is not n-Turing reducible to A.

iii. For every n, there is a total function $(n+1)$-strong-Turing reducible to A that is not n-strong-Turing reducible to A.

iv. For every n, there is a total function $(n+1)$-truth-table reducible to A that is not n-weak-truth-table reducible to A.

We note that a more constructive proof of Lemma 7(iii) is possible. (The proof depends on Lemma 24, which is proved in Section 4.3..)

Proposition 10 *If A is nonrecursive then there is a total function $(n+1)$-tt reducible to A that is not n-wtt reducible to A.*

Proof: We claim that $\mathrm{FQ}_{||}(n,A) \neq \mathrm{FQ}_{||}(n+1,A)$. The claim is proved by contradiction. Assume that $\mathrm{FQ}_{||}(n,A) = \mathrm{FQ}_{||}(n+1,A)$. Then $\mathrm{FQ}_{||}(m,A) = \mathrm{FQ}_{||}(n,A)$ for all $m \geq n$, by Lemma 24(ii). In particular $\mathrm{FQ}_{||}(2^n,A) = \mathrm{FQ}_{||}(n,A)$. Therefore, $\mathrm{F}^A_{2^n} \in \mathrm{FQ}_{||}(2^n,A) = \mathrm{FQ}_{||}(n,A) \subseteq \mathrm{FQ}(n,A)$. By the Nonspeedup Theorem, A must be recursive. Thus the claim is established. Therefore, F^A_{n+1} is not n-wtt reducible to A; of course F^A_{n+1} is $(n+1)$-tt reducible to A. ▮

This proof technique can also be applied to parts (i) and (ii) of Lemma 7, but it yields only partial functions.

We note that the separation in Lemma 8 is stronger for tt-reductions than for Turing-reductions. This motivates the following open question:

Open Question 11 *When does there exist a total function that is $(n+1)$-strong-Turing reducible to A but not n-Turing reducible to A?*

We are not aware of any nonrecursive set A for which such a function does not exist. For example, letting K denote the halting set, it follows from results in [BGH89] that the function $\#^K_{2^{n+1}-1}$ is $(n+1)$-strong-Turing reducible to K but not n-Turing reducible to K. $\#^A_n$ is defined below:

Definition 12 $\#^A_n(x_1,\ldots,x_n) = \sum_{1\leq i\leq n} A(x_i)$.

4. Supportive Sets

In this section we ask when there are more characteristic functions computable by making $n+1$ queries to A than by making only n queries to A. The answer is more complicated than in the preceding section. Intuitively, we expect some subtlety because a characteristic function produces only one bit of information, regardless of how many bits of information are required for its computation. We note that the methods of Sections 4.2. and 4.3. can also be applied in the context of complexity theory [Bei91].

4.1. Additional Terminology

We will use the following additional terminology:

Definition 13

- $\mathrm{Q}(n, A)$ is the class of all sets S such that $S \leq_{n\text{-T}} A$.
- $\mathrm{Q}_{||}(n, A)$ is the class of all sets S such that $S \leq_{n\text{-wtt}} A$.
- $\mathrm{Q}(\omega, A)$ is the class of all sets S such that $S \leq_{\mathrm{T}} A$.
- $\mathrm{Q}_{||}(\omega, A)$ is the class of all sets S such that $S \leq_{\mathrm{wtt}} A$.

Note that the definitions above make sense if the set A is replaced by a partial function. If a machine M with 0,1-valued output computes an n-Turing reduction to an oracle A then we call M a $\mathrm{Q}(n, A)$ *machine*, and so on for the other bounded query classes.

Definition 14

- A set A is *supportive* if, for all n, $\mathrm{Q}(n, A) \subset \mathrm{Q}(n+1, A)$.
- A set A is *parallel-supportive* if, for all n, $\mathrm{Q}_{||}(n, A) \subset \mathrm{Q}_{||}(n+1, A)$.

We will adopt a convention of not distinguishing among sets, predicates, and 0,1-valued total functions.

4.2. The u_A Lemma

This section provides a very useful tool for proving that a set is supportive and parallel-supportive. The proof of the u_A Lemma is very similar to the proof of Lemma 7.

We define a predicate, PARITY_n^A, which determines whether an odd number of n strings are elements of A. Treated as a set, this predicate is will help us to separate bounded query classes of decision problems. Recall Definition 12 ($\#_n^A$).

Definition 15

$$\mathrm{PARITY}_n^A(x_1, \ldots, x_n) = \#_n^A(x_1, \ldots, x_n) \bmod 2.$$

We define a predicate tt_n^A, which evaluates a truth table condition of norm n over the set A. This predicate will also help us to separate bounded query classes of decision problems.

Definition 16

- The *norm* of a truth-table is the number of inputs it has.
- tt_n^A is the set of all $(f, x_1, \ldots, x_n)$ such that f is a truth table of norm n that evaluates to 1 when its ith input is set equal to $A(x_i)$ for all i.

Lemma 17 (u_A)

i. If $(\forall m)(\exists n)[\mathrm{PARITY}_n^A \notin \mathrm{Q}_{||}(m, A)]$ *then* A *is parallel-supportive.*

ii. If $(\forall m)(\exists n)[\mathrm{PARITY}_n^A \notin \mathrm{Q}(m, A)]$ *then* A *is supportive and parallel-supportive.*

iii. If $(\forall m)(\exists n)[\mathrm{tt}_n^A \notin \mathrm{Q}(m, A)]$ *then* A *is supportive.*

Proof:

i. Define

$$u_A(m) = \max\{n : \mathrm{PARITY}_n^A \in \mathrm{Q}_{||}(m, A)\}.$$

(Our assumption provides an upper bound, so the maximum exists.) Then

$$\mathrm{PARITY}_{u_A(m)+1} \notin \mathrm{Q}_{||}(m, A),$$

and

$$\mathrm{PARITY}_{u_A(m)} \in \mathrm{Q}_{||}(m, A).$$

This containment implies that

$$\mathrm{PARITY}_{u_A(m)+1} \in \mathrm{Q}_{||}(m + 1, A),$$

(We use m parallel queries to determine whether an odd number of the first $u_A(m)$ items belong to A, and we use one additional parallel query to determine whether the last item belongs to A. Then we compute the exclusive-or of the two results.) Thus

$$\mathrm{PARITY}_{u_A(m)+1} \in \mathrm{Q}_{||}(m + 1, A) - \mathrm{Q}_{||}(m, A),$$

and since the inclusion follows by definition,

$$\mathrm{Q}_{||}(m, A) \subset \mathrm{Q}_{||}(m + 1, A).$$

ii. Since $Q_{||}(m, A) \subseteq Q(m, A)$, part (i) implies that A is parallel-supportive. To show that A is supportive, define

$$u_A(m) = \max\{n : \mathrm{PARITY}_n^A \in Q(m, A)\},$$

and proceed as in (i).

iii. To show that A is supportive, define

$$u_A(m) = \max\{n : \mathrm{tt}_n^A \in Q(m, A)\},$$

and proceed as in (i). ($\mathrm{tt}^A_{u_A(m)+1} \in Q(m+1, A)$, because we can make one query to determine the value of the $(u_A(m)+1)$st input to the truth-table; plug in that value, thereby reducing the truth table to norm $u_A(m)$; and finally make m serial queries to evaluate the truth table.)

∎

Of course, the u_A Lemma is valid for other predicates like disjunction and conjunction. PARITY_n^A and tt_n^A seem to be the most useful for our purposes. It is somewhat unsatisfying that the proof of part (iii) does not work for parallel-supportiveness.

4.3. Supportiveness vs. Superterseness

In this section we will use the u_A Lemma to show that all non-superterse sets are either supportive or recursive. Before doing so, we need some technical definitions and lemmas.

Superterseness was defined in [BGGO93].

Definition 18 A set A is *superterse* if

$$(\forall B)(\forall k)[\mathrm{F}_k^A \notin \mathrm{FQ}(k-1, B)].$$

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

Definition 19 If C_1 and C_2 are two sets of functions then

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

We use the following two results from [BGH89]:

Lemma 20 (Rounds of Queries) *A function g can be computed by making at most n_1 parallel queries to f_1, followed by at most n_2 parallel queries to f_2, ... followed by at most n_r parallel queries to f_r if and only if*

$$g \in \mathrm{FQ}_{||}(n_r, f_r) \circ \mathrm{FQ}_{||}(n_{r-1}, f_{r-1}) \circ \cdots \circ \mathrm{FQ}_{||}(n_1, f_1).$$

Lemma 21 (Composition)

$$\mathrm{FQ}(i+j, A) = \mathrm{FQ}(i, A) \circ \mathrm{FQ}(j, A).$$

Let FREC denote the set of partial recursive functions. The following result about parallel queries is analogous to the Rounds-of-Queries Lemma. Recall Definition 5 ($\|$).

Lemma 22 *f can be computed by making at most n_1 parallel queries to A_1, simultaneous with at most n_2 parallel queries to A_2 , ..., and simultaneous with at most n_r parallel queries to A_r if and only if*

$$f \in \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n_1, A_1) \,\|\, \mathrm{FQ}_{||}(n_2, A_2) \,\|\, \cdots \,\|\, \mathrm{FQ}_{||}(n_r, A_r)).$$

Proof: First, assume that f can be computed by an oracle Turing machine that makes at most n_1 parallel queries to A_1, simultaneous with at most n_2 parallel queries to A_2 , ..., and simultaneous with at most n_r parallel queries to A_r. Without loss of generality, we may assume that f's queries are separated into r groups, such that the ith group always consists of at most n_i queries to A_i. Define f_i as follows: Simulate f until f is about to make some queries; make the queries in the ith group; print their results as a bit-string. Define f' as follows: Treat the input as r bit-strings; simulate f until f is about to make some queries; use the ith input bit-string as the answers to the ith group of queries; simulate the remainder of f's computation. Thus

$$f = f' \circ (f_1 \,\|\, f_2 \,\|\, \cdots \,\|\, f_r),$$

so

$$f \in \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n_1, A_1) \,\|\, \mathrm{FQ}_{||}(n_2, A_2) \,\|\, \cdots \,\|\, \mathrm{FQ}_{||}(n_r, A_r)).$$

Conversely, assume that

$$f \in \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n_1, A_1) \,\|\, \mathrm{FQ}_{||}(n_2, A_2) \,\|\, \cdots \,\|\, \mathrm{FQ}_{||}(n_r, A_r)).$$

Then

$$f = f' \circ (f_1 \,\|\, f_2 \,\|\, \cdots \,\|\, f_r),$$

where each f_i is in $\mathrm{FQ}_{||}(n_i, A_i)$ and f' is recursive. We can compute f as follows: Simulate f_1 through f_r until each has produced an output or is about to make some queries; make all the queries in parallel; simulate the remainder of their computations; simulate the computation of f'. Thus f can be computed by an oracle Turing machine that makes at most n_1 parallel queries to A_1, simultaneous with at most n_2 parallel queries to A_2, ..., and simultaneous with at most n_r parallel queries to A_r. ∎

Note that the function f' defined in the first half of the proof above is not total, because f' might diverge when its input does not encode correct oracle answers. We noted the same phenomenon in [BGH89] when we proved the Composition Lemma. This is the main reason for defining FQ and $\mathrm{FQ}_{||}$ in terms of partial functions.

As a special case of the preceding lemma we obtain:

Lemma 23 (Concatenation)

$$\mathrm{FQ}_{||}(m + n, A) = \mathrm{FREC} \circ (\mathrm{FQ}_{||}(m, A) \,\|\, \mathrm{FQ}_{||}(n, A)).$$

The following result, already used in Section 3.2., follows easily:

Lemma 24 (Collapse)

i. If $\mathrm{FQ}(n + 1, A) = \mathrm{FQ}(n, A)$ *then*

$$(\forall m \geq n)[\mathrm{FQ}(m, A) = \mathrm{FQ}(n, A)].$$

ii. If $\mathrm{FQ}_{||}(n + 1, A) = \mathrm{FQ}_{||}(n, A)$ *then*

$$(\forall m \geq n)[\mathrm{FQ}_{||}(m, A) = \mathrm{FQ}_{||}(n, A)].$$

Proof:

i. Assume that $\mathrm{FQ}(n,A) = \mathrm{FQ}(n+1,A)$. For all $t \geq 0$,

$$\begin{aligned}
\mathrm{FQ}(n+t+1,A) &= \mathrm{FQ}(n+1+t,A) \\
&= \mathrm{FQ}(n+1,A) \circ \mathrm{FQ}(t,A) \qquad \text{by the Composition Lemma} \\
&= \mathrm{FQ}(n,A) \circ \mathrm{FQ}(t,A) \qquad \text{by assumption} \\
&= \mathrm{FQ}(n+t,A) \qquad \text{by the Composition Lemma.}
\end{aligned}$$

Thus $\mathrm{FQ}(n+t+1,A) = \mathrm{FQ}(n+t,A)$ for all $t \geq 0$. By transitivity, $\mathrm{FQ}(m,A) = \mathrm{FQ}(n,A)$ for all $m \geq n$.

ii. Assume that $\mathrm{FQ}_{||}(n,A) = \mathrm{FQ}_{||}(n+1,A)$. For all $t \geq 0$,

$$\begin{aligned}
\mathrm{FQ}_{||}(n+t+1,A) &= \mathrm{FQ}_{||}(n+1+t,A) \\
&= \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n+1,A) \,\|\, \mathrm{FQ}_{||}(t,A)) \\
&\ \text{by the Concatenation Lemma} \\
&= \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n,A) \,\|\, \mathrm{FQ}_{||}(t,A)) \qquad \text{by assumption} \\
&= \mathrm{FQ}_{||}(n+t,A) \qquad \text{by the Concatenation Lemma.}
\end{aligned}$$

Thus $\mathrm{FQ}_{||}(n+t+1,A) = \mathrm{FQ}_{||}(n+t,A)$ for all $t \geq 0$. By transitivity, $\mathrm{FQ}_{||}(m,A) = \mathrm{FQ}_{||}(n,A)$ for all $m \geq n$.

∎

Lemma 25 (Synchronization)

$$\mathrm{FREC} \circ \underbrace{\mathrm{FQ}(n,A) \,\|\, \cdots \,\|\, \mathrm{FQ}(n,A)}_{m} \subseteq \underbrace{\mathrm{FQ}_{||}(m,A) \circ \cdots \circ \mathrm{FQ}_{||}(m,A)}_{n}$$

Proof: The left-hand side is the class of all sets than can be computed by m parallel processes that each make at most n serial queries to A and then pass their output to a recursive function. If we force the processes to synchronize before each query, then we can make all the queries in n rounds of m parallel queries. ∎

We use the following theorem from [Bei87b]. A complexity-theoretic variant of the theorem also appears in [Bei87c].

Theorem 26 *If A is not superterse then for every n there exists a set $C \le_{n\text{-tt}} A$ such that*

$$\mathrm{F}_n^A \in \mathrm{FQ}_{||}(O(\log n), C).$$

The following theorem provides a nice relation between supportiveness and superterseness.

Theorem 27 *If B is not superterse or recursive then B is supportive.*

Proof: Suppose that B is not superterse. By Theorem 26, for each n, there exists a set $C_n \le_{n\text{-tt}} B$ such that

$$\mathrm{F}_n^B \in \mathrm{FQ}_{||}(O(\log n), C_n).$$

Suppose also that B is not supportive. We will show that B is recursive. By part (iii) of the u_A lemma there exists k such that

$$(\forall n)[\mathrm{tt}_n^B \subseteq \mathrm{Q}(k, B)].$$

Since $C_n \le_{n\text{-tt}} B$, this implies that

$$(\forall n)[C_n \in \mathrm{Q}(k, B)].$$

Thus,

$$\begin{aligned}
\mathrm{F}_n^B &\in \mathrm{FQ}_{||}(O(\log n), C_n) \\
&\subseteq \mathrm{FREC} \circ \left(\underbrace{\mathrm{Q}(k,B) \parallel \cdots \parallel \mathrm{Q}(k,B)}_{O(\log n)} \right) \qquad \text{because } C_n \in \mathrm{Q}(k,B) \\
&\subseteq \mathrm{FREC} \circ \left(\underbrace{\mathrm{FQ}_{||}(O(\log n), B) \circ \cdots \circ \mathrm{FQ}_{||}(O(\log n), B)}_{k} \right) \\
&\qquad \text{by the Synchronization Lemma (25).}
\end{aligned}$$

Applying Theorem 26 a second time, we find that for all n there exists a set D such that

$$\mathrm{FQ}_{||}(O(\log n), B) \subseteq \mathrm{FQ}_{||}(O(\log\log n), D).$$

Therefore,

$$\begin{aligned} F_n^B &\in \mathrm{FREC} \circ \underbrace{\mathrm{FQ}_{\|}(O(\log\log n), D) \circ \cdots \circ \mathrm{FQ}_{\|}(O(\log\log n), D)}_{k} \\ &\subseteq \mathrm{FQ}(O(k \log\log n), D). \end{aligned}$$

By the Nonspeedup Theorem, B must be recursive. ∎

Corollary 28

i. *All nonrecursive semi-recursive sets (defined in Section 4.6.) are supportive.*

ii. *All nonrecursive r.e. sets are supportive.*

iii. *All nonrecursive n-r.e. sets (defined in the next section) are supportive.*

Proof:

i. In [BGGO93] we showed that semi-recursive sets are not superterse.

ii. In [BGGO93] we showed that r.e. sets are not superterse.

iii. In [BGGO93] we showed that n-r.e. sets are not superterse.

∎

4.4. r.e. and n-r.e. Sets

In the preceding section we showed that all nonrecursive r.e. and n-r.e. sets are supportive. In this section, we use different techniques to show that all nonrecursive r.e. and n-r.e. sets are parallel-supportive. These methods also provide an alternative proof that r.e. and n-r.e. sets are supportive. The proof uses the notion of computability by a set of partial recursive functions.

In [BGGO93, Bei87b], we defined a notion of computability that captures the information-theoretic aspects of n-query oracle computations. The new notion of computability is independent of the particular oracle being used, thus allowing us to apply purely combinatorial techniques to the study of bounded-query computations.

Definition 29 The partial function h is *computable by a set of n partial recursive functions* if there exist n partial recursive functions $g_1, \ldots, g_n$ such that

$$(\forall x)[\text{if } h(x) \text{ converges then } h(x) \in \{g_i(x) : 1 \leq i \leq n\}].$$

(If $g_i(x)$ does not converge then we exclude its value from the set above, by convention.)

Thus, for example, if for each x we can effectively compute a list of length $\leq n$ that includes $h(x)$, then the function h is computable by a set of n partial recursive functions. In this case, we say that *there are only n possible values* for $h(x)$.

The next theorem says that computability by a set of partial recursive functions captures the information-theoretic aspects of computability by an oracle. It is proved in [BGGO93, Bei87b].

Theorem 30

i. If there exists a set B such that $h \in \mathrm{FQ}(n, B)$, then h is computable by a set of 2^n partial recursive functions.

ii. If h is computable by a set of 2^n partial recursive functions, then there exists a set B such that $h \in \mathrm{FQ}_{||}(n, B)$. If h is total, then B is in $\mathrm{Q}(1, h)$

We can state a version of the Nonspeedup Theorem in terms of computability by a set of partial recursive functions. This result was proved in [BGGO93] as a lemma to the Nonspeedup Theorem.

Theorem 31 *If F_n^A is computable by a set of n partial recursive functions, then A is recursive.*

In the next two lemmas we will show that if A is r.e. then F_n^A is computable by a set of $n + 1$ partial recursive functions.

Lemma 32 *If A is r.e. then $\mathrm{F}_n^A \in \mathrm{FQ}(1, \#_n^A)$.*

Proof: If we know that exactly m out of n items belong to A then we can run an enumerator of A until it enumerates exactly m of the n items. Those m items belong to A. The remaining $n - m$ items do not belong to A. ∎

Lemma 33 *If A is r.e. then F_n^A is computable by a set of $n+1$ partial recursive functions.*

Proof: By Lemma 32, $\mathrm{F}_n^A \in \mathrm{FQ}(1, \#_n^A)$. There are only $n+1$ possible values $(0, \ldots, n)$ that $\#_n^A$ can take on. ∎

The predicate GEQ^A, defined below, determines whether at least t out of n strings are elements of A.

Definition 34

$$\mathrm{GEQ}^A(t; x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \#_n^A(x_1, \ldots, x_n) \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

It is obvious that if A is r.e. then GEQ^A is r.e. The next two lemmas show that if A is r.e. then PARITY_{2n+1}^A helps us to compute $\#_{2n+1}^A$. Because $\#_{2n+1}^A$ is as powerful as F_{2n+1}^A when A is r.e., we will be able to show that $\mathrm{PARITY}_{2n+1}^A \notin \mathrm{Q}_{||}(n, A)$.

Lemma 35 *For all sets A,*

$$\#_{2n+1}^A \in \mathrm{FREC} \circ (\mathrm{FQ}(1, \mathrm{PARITY}_{2n+1}^A) \,\|\, \mathrm{FQ}_{||}(n, \mathrm{GEQ}^A)).$$

Proof: If $0 \leq m \leq 2n+1$ then we can determine m by asking if m is odd and asking if $m \geq 2, m \geq 4, \ldots, m \geq 2n$. ∎

Lemma 36 *If A is r.e. then*

$$\#_{2n+1}^A \in \mathrm{FREC} \circ (\mathrm{FQ}(1, \mathrm{PARITY}_{2n+1}^A) \,\|\, \mathrm{FQ}_{||}(n, K)).$$

Proof: By Lemma 35,

$$\#_{2n+1}^A \in \mathrm{FREC} \circ (\mathrm{FQ}(1, \mathrm{PARITY}_{2n+1}^A) \,\|\, \mathrm{FQ}_{||}(n, \mathrm{GEQ}^A)).$$

Because A is r.e., GEQ^A is r.e. Therefore $\mathrm{GEQ}^A \leq_{\mathrm{m}} K$, so $\mathrm{FQ}_{||}(n, \mathrm{GEQ}^A) \subseteq \mathrm{FQ}_{||}(n, K)$. ∎

Now we show that PARITY_{2n+1}^A requires at least n queries. This will enable us to apply the u_A Lemma.

Lemma 37 *If A is a nonrecursive r.e. set then*

$$(\forall n)[\mathrm{PARITY}^A_{2n+1} \notin \mathrm{Q}_{||}(n,K)].$$

Proof: Let A be an r.e. set, and suppose that

$$\mathrm{PARITY}^A_{2n+1} \in \mathrm{Q}_{||}(n,K).$$

By Lemma 36,

$$\begin{aligned}
\#^A_{2n+1} &\in \mathrm{FREC} \circ (\mathrm{FQ}(1, \mathrm{PARITY}^A_{2n+1}) \parallel \mathrm{FQ}_{||}(n,K)) \\
&\subseteq \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n,K) \parallel \mathrm{FQ}_{||}(n,K)) \\
&= \mathrm{FQ}_{||}(2n,K).
\end{aligned}$$

Since $\mathrm{F}^A_{2n+1} \in \mathrm{FQ}(1, \#^A_{2n+1})$ by Lemma 32, we have

$$\begin{aligned}
\mathrm{F}^A_{2n+1} &\in \mathrm{FQ}_{||}(2n,K) \\
&= \mathrm{FQ}(1, \mathrm{F}^K_{2n}).
\end{aligned}$$

By Lemma 33, F^K_{2n} is computable by a set of $2n+1$ partial recursive functions. Therefore F^A_{2n+1} is computable by a set of $2n+1$ partial recursive functions. By Theorem 31, A is recursive. ∎

Theorem 38 *If A is a nonrecursive r.e. set then A is supportive and parallel supportive.*

Proof: By Lemma 37,

$$\mathrm{PARITY}^A_{2^{m+1}-1} \notin \mathrm{Q}_{||}(2^m-1,K)$$

In [BGH89] we showed that $\mathrm{Q}(m,K) = \mathrm{Q}_{||}(2^m-1,K)$. Therefore,

$$\mathrm{PARITY}^A_{2^{m+1}-1} \notin \mathrm{Q}(m,K).$$

Since $A \leq_{\mathrm{m}} K$,

$$\mathrm{PARITY}^A_{2^{m+1}-1} \notin \mathrm{Q}(m,A).$$

Lemma 17(ii) implies that A is supportive and parallel supportive. ∎

Theorem 39 *For all B, the set B' is supportive and parallel-supportive.*

Proof: By relativizing the preceding proofs, we find that if A is r.e. in B, but not recursive in B, then A is supportive and parallel supportive. In particular, B' is supportive and parallel-supportive. ∎

The n-r.e. sets were defined by Epstein, Haas, and Kramer in [EHK81]. The definition can also be found in [Soa87]. The following definition is equivalent to theirs and, for our purposes, is easier to use.

Definition 40 A is *n-r.e.* if $A \leq_m \mathrm{PARITY}_n^K$.

Theorem 41 *If A is a nonrecursive n-r.e. set then A is supportive and parallel supportive.*

Proof: By the u_A Lemma, it is sufficient to prove the following claim:

$$(\forall m)[\mathrm{PARITY}^A_{2^{mn+1}-1} \notin \mathrm{Q}(m, A)].$$

Suppose that the claim is false, so that there exists m such that

$$\mathrm{PARITY}^A_{2^{mn+1}-1} \in \mathrm{Q}(m, A).$$

Since $A \in \mathrm{Q}_{||}(n, K)$,

$$\mathrm{PARITY}^A_{2^{mn+1}-1} \in \mathrm{Q}(mn, K).$$

Therefore,

$$\mathrm{PARITY}^{\bar{A}}_{2^{mn+1}-1} \in \mathrm{Q}(mn, K).$$

If A is a nonrecursive n-r.e. set then Epstein, Haas, and Kramer have shown that there exists a nonrecursive r.e. set B such that $B \leq_m A$ or $B \leq_{\mathrm{m}} \bar{A}$ [EHK81, Theorem 3][1]. Since $B \leq_{\mathrm{m}} A$ or $B \leq_{\mathrm{m}} \bar{A}$,

$$\mathrm{PARITY}^B_{2^{mn+1}-1} \in \mathrm{Q}(mn, K).$$

However, in the proof of Theorem 38 we showed that if B is a nonrecursive r.e. set then

$$(\forall i)[\mathrm{PARITY}^B_{2^{i+1}-1} \notin \mathrm{Q}(i, K)].$$

This contradiction establishes the claim. ∎

[1] They only claim to prove that $B \leq_{\mathrm{T}} A$; however their proof sketch yields the stronger result. A more detailed proof sketch appears in [Eps79, Theorem 7 on page 152]

Corollary 42 *If A is a nonrecursive set belong to* $\mathrm{Q}(n, K)$ *then A is supportive and parallel supportive.*

Proof: In [BGH89] we showed that

$$\mathrm{Q}(n, K) = \mathrm{Q}_{||}(2^n - 1, K) \subset \text{the class of } 2^n\text{-r.e. sets.}$$

$\blacksquare$

4.5. Jumps

In the preceding section we showed that B' is supportive and parallel-supportive for all B. In this section, we strengthen that result by showing that for all n there is a set that is $(n+1)$-tt reducible to B' but not n-wtt to B'; in the next section, we will apply the same proof technique to semi-recursive sets. We also show that an extra serial query to B' allows us to solve the halting problem for machines that compute with an oracle for B'. As a corollary, we further strengthen the results of the preceding section by showing that for all n there is a set that is $(n+1)$-strong-Turing reducible to B' but not n-Turing reducible to B'. Via related techniques, we show that the set $\mathrm{tt}^{B'}$ is supportive; in contrast, in Section 5.1. we will show that $\mathrm{tt}^{B'}$ is not parallel-supportive. Thus there exists a set that is supportive but not parallel-supportive.

Definition 43 tt^A is the set of truth tables that evaluate to 1 when the ith indeterminate is set equal to $A(i)$ for all i.

By, Theorem 39 we know that B' is parallel-supportive. Using different techniques, we construct a set that is $(n+1)$-tt reducible to B' but not n-wtt reducible to B'. The proof uses only two very simple properties of B', and in the next section we will see that the same proof is valid for nonrecursive semi-recursive sets.

The following lemma says that, for any set A, $\mathrm{PARITY}_{n+1}^{\mathrm{GEQ}^A}$ helps us to compute $\#_{2n+1}^A$. Because $\#_{2n+1}^K$ is 1-tt equivalent to F_{2n+1}^K and GEQ^K is m-equivalent to K, we will be able to show that $\mathrm{PARITY}_{n+1}^K \notin \mathrm{Q}_{||}(n, K)$. Our proof will relativize to show that $\mathrm{PARITY}_{n+1}^{B'} \notin \mathrm{Q}_{||}^B(n, B')$.

Lemma 44 *For all sets A,*

$$\#_{2n+1}^A \in \mathrm{FREC} \circ (\mathrm{FQ}(1, \#_n^{\mathrm{GEQ}^A}) \parallel \mathrm{FQ}(1, \mathrm{PARITY}_{n+1}^{\mathrm{GEQ}^A})).$$

Proof: Suppose that we are to compute $\#_{2n+1}^A(\vec{x})$, where $\vec{x} = (x_1, \ldots, x_{2n+1})$. Let $\vec{y} = (y_1, \ldots, y_{2n+1})$, where

$$y_i = (i; \vec{x}).$$

Then

$$\#_{2n+1}^A(\vec{x}) = \#_{2n+1}^{\mathrm{GEQ}^A}(\vec{y}).$$

Let

$$\begin{aligned} z &= \#_{2n+1}^{\mathrm{GEQ}^A}(\vec{y}), \\ t &= \#_n^{\mathrm{GEQ}^A}(y_2, y_4, \ldots, y_{2n}), \\ p &= \mathrm{PARITY}_{n+1}^{\mathrm{GEQ}^A}(y_1, y_3, \ldots, y_{2n+1}). \end{aligned}$$

Then $z = 2t$ or $z = 2t + 1$, depending on whether $y_{2t+1} \in \mathrm{GEQ}^A$. Since the parity function computed above changes value if $y_{2t+1} \in \mathrm{GEQ}^A$, the value of p determines whether $z = 2t$ or $z = 2t + 1$; thus z is determined by the values of t and p. In fact

$$z = \begin{cases} 2t + p & \text{if } t \text{ is even} \\ 2t + 1 - p & \text{otherwise.} \end{cases}$$

Thus, we can compute $\#_{2n+1}^A(\vec{x}) = \#_{2n+1}^{\mathrm{GEQ}^A}(\vec{y})$ by making one query to $\#_n^{\mathrm{GEQ}^A}$ simultaneous with one query to $\mathrm{PARITY}_{n+1}^{\mathrm{GEQ}^A}$. ∎

Lemma 45

$$\mathrm{PARITY}_{n+1}^K \notin \mathrm{Q}_{||}(n, K).$$

Proof: By Lemma 44,

$$\#_{2n+1}^K \in \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n, \mathrm{GEQ}^K) \parallel \mathrm{FQ}_{||}(1, \mathrm{PARITY}_{n+1}^{\mathrm{GEQ}^K})).$$

Since K is r.e., GEQ^K is r.e., so $\mathrm{GEQ}^K \leq_{\mathrm{m}} K$. Therefore,

$$\#_{2n+1}^K \in \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n, K) \parallel \mathrm{FQ}_{||}(1, \mathrm{PARITY}_{n+1}^K)).$$

Suppose, for the sake of contradiction, that

$$\mathrm{PARITY}^K_{n+1} \in \mathrm{Q}_{||}(n, K).$$

Then

$$\begin{aligned} \#^K_{2n+1} &\in \mathrm{FREC} \circ (\mathrm{FQ}_{||}(n, K) \parallel \mathrm{FQ}_{||}(n, K)) \\ &= \mathrm{FQ}_{||}(2n, K). \end{aligned}$$

By Lemma 32, $\mathrm{FQ}_{||}(m, K) \subseteq \mathrm{FQ}(1, \#^K_m)$ for all m. Letting $m = 2n + 1$ and using the preceding containment, we have

$$\mathrm{FQ}_{||}(2n + 1, K) \subseteq \mathrm{FQ}_{||}(2n, K).$$

However, since K is nonrecursive, Lemma 7(iii) says that

$$(\forall m)[\mathrm{FQ}_{||}(m + 1, K) \not\subseteq \mathrm{FQ}_{||}(m, K)].$$

This is a contradiction (with $m = 2n$). ∎

Theorem 46 *There exists a set that is $(n + 1)$-tt reducible to K but not n-wtt reducible to K.*

Proof: By the preceding lemma, PARITY^K_{n+1} is such a set. ∎

By relativizing the proof, we obtain the following corollary:

Corollary 47 *For every set B and every n, there exists a set that is $(n+1)$-tt reducible to B' but not n-wtt reducible to B'.*

Next we show that there is a problem that is $(n + 1)$-strong-Turing reducible to B' but not n-Turing reducible to B'.

Theorem 48 *The halting problem for $\mathrm{FQ}^B(n, B')$ machines is $(n + 1)$-strong-Turing reducible to B'.*

Proof: The following algorithm determines whether an $\mathrm{FQ}^B(n, B')$ machine halts on empty input:

Step 1: Input x.

Step 2: For $i = 1$ to $n + 1$ do the following:

(a) Construct a Turing machine x_i with oracle B that computes as follows: Using the values $B'(x_1), \ldots, B'(x_{i-1})$ computed in step 2(b) as the first $i - 1$ oracle answers, simulate x on empty input until x has halted or x is about to make its ith query q_i; if x is about to make its ith query q_i then simulate q_i until q_i has halted.

(b) Make one query to B' to determine whether x_i halts.

(Note that if x makes an ith query q_i, then $B'(x_i) = B'(q_i)$.)

Step 3: Output the value of $B'(x_{n+1})$ that was computed in step 2(b).

We assert that x halts if and only if x_{n+1} halts. Let j be the actual number of queries to B' made by machine x. By construction, $x_i \in B'$ if and only if $q_i \in B'$ for $1 \leq i \leq j$. If x halts then x_{j+1} through x_{n+1} halt; if x diverges then x_{j+1} through x_{n+1} diverge.

The algorithm constructed above makes exactly $n + 1$ queries to K. ▮

Lemma 49 *The halting problem for* $\mathrm{FQ}^B(n, B')$ *machines does not belong to* $\mathrm{Q}^B(n, B')$*.*

Proof: This is an easy modification to standard proofs [Rog67, Soa87] that the halting problem for Turing machines is not solvable by a Turing machine. ▮

Corollary 50 *For every n, there is a set that is $(n+1)$-strong-Turing reducible to B' but not n-Turing reducible to B'*

Proof: By Theorem 48 and Lemma 49, the halting problem for $\mathrm{FQ}^B(n, B')$ machines is such a set. ▮

Next, we present another another supportive set, $\mathrm{tt}^{B'}$ (see Definition 43), in every jump degree. In contrast, in Section 5.1. we will show that $\mathrm{tt}^{B'}$ is not parallel-supportive. Thus, supportive sets need not be parallel-supportive. To begin, we present a result analogous to Theorem 48 for parallel queries.

Theorem 51 *For every n, the halting problem for $\mathrm{FQ}_{\|}^{B}(n, B')$ machines is $(2n+1)$-tt reducible to B'.*

Proof: We solve the halting problem for $\mathrm{FQ}_{\|}^{B}(n, B')$ machines as follows:

Step 1: Input M.

Step 2: For $1 \leq i \leq n$ construct a B-machine x_i that does the following: Simulate M until it prepares its list of queries; then simulate the ith query until it halts. (Note that if M actually makes its round of queries then $x_i \in B'$ iff M's ith query belongs to B'.)

Step 3: For $0 \leq i \leq n$ construct a B-machine M_i that does the following: Simulate M until it prepares its list of queries; timeshare all of the queries until exactly i of them halt; assuming that those are the only queries that belong to B', complete the simulation of M.

Step 4: In parallel ask whether $x_1, \ldots, x_n$ and $M_0, \ldots, M_n$ halt (belong to B'). If exactly i of the machines $x_1, \ldots, x_n$ halt, then report that M halts iff M_i halts.

As in the proof of Theorem 48, it is a straightforward matter to verify the correctness of this algorithm. ∎

Theorem 52 *For all n and m, the halting problem for $\mathrm{FQ}^{B}(n, \mathrm{tt}^{B'}) \circ \mathrm{FQ}_{\|}^{B}(m, B')$ machines is contained in $\mathrm{Q}(n+1, \mathrm{tt}^{B'})$.*

Proof: The following algorithm solves the halting problem for $\mathrm{FQ}^{B}(n, \mathrm{tt}^{B'}) \circ \mathrm{FQ}_{\|}^{B}(m, B')$ machines:

Step 1: Input a $\mathrm{FQ}^{B}(n, \mathrm{tt}^{B'}) \circ \mathrm{FQ}_{\|}^{B}(m, B')$ machine x. Without loss of generality assume that x makes exactly m parallel queries to B' followed by exactly n serial queries to $\mathrm{tt}^{B'}$ whenever x halts.

Step 2: If $n = 0$ then determine whether x halts by making a single query to $\mathrm{tt}^{B'}$ as in Theorem 51; otherwise go on.

Step 3: Construct a machine x' that (1) simulates x until x prepares its first query to $\mathrm{tt}^{B'}$ and then (2) halts. Since that is a $\mathrm{Q}_{||}^{B}(m, B')$ computation, we can determine whether x' halts by making one query to $\mathrm{tt}^{B'}$ (by Theorem 51). If x' does not halt, then halt and reject.

Step 4: Simulate x until it prepares its first query to $\mathrm{tt}^{B'}$. Construct a new machine x'' that simulates the remainder of the computation from where x' left off. Recursively, determine whether x'' halts by making n serial queries to $\mathrm{tt}^{B'}$.

∎

Corollary 53 $\mathrm{tt}^{B'}$ *is supportive.*

Proof: By the preceding theorem, the halting problem for $\mathrm{FQ}(n, \mathrm{tt}^{B'})$ machines is in $\mathrm{Q}(n+1, \mathrm{tt}^{B'})$. By standard reasoning, it is not in $\mathrm{Q}(n, \mathrm{tt}^{B'})$. ∎

Note that it is not possible to separate $(n+1)$-strong-Turing reducibility to $\mathrm{tt}^{B'}$ from n-strong-Turing reducibility to $\mathrm{tt}^{B'}$, because strong-Turing reducibility is the same as tt-reducibility.

Note also that for all sets A, $\mathrm{PARITY}_n^{\mathrm{tt}^A}$ and $\mathrm{tt}_n^{\mathrm{tt}^A}$ belong to $\mathrm{Q}_{||}(1, \mathrm{tt}^A)$. Thus the converses to parts (ii) and (iii) of the u_A Lemma (17) are false.

We close this section by proving a lemma that will be useful in Section 5.1., when we show that $\mathrm{tt}^{B'}$ is not parallel-supportive. The lemma is a modification of a theorem in [BGH89] that states that if $L \leq_{n\text{-wtt}} K$ then $L \leq_{1\text{-tt}} \mathrm{PARITY}_n^K$.

Lemma 54

i. If $L \in \mathrm{Q}_{||}^{B}(n, B')$ *then* $L \leq_{\mathrm{m}} \mathrm{PARITY}_{n+1}^{B'}$.

ii. If $L \leq_{\mathrm{wtt}} B'$ *then* $L \leq_{\mathrm{tt}} B'$.

Proof:

i. Assume that $L \in \mathrm{Q}_{||}^{B}(n, B')$. In [BGH89] we showed that if $A \in \mathrm{Q}_{||}(n, K)$ then L is weakly n-r.e., meaning that

$$A(x) = \lim_{s \to \infty} f(x, s)$$

where f is total recursive and there exist at most n values of s for which $f(x, s-1) \neq f(x, s)$. By relativizing that proof we see that

$$L(x) = \lim_{s \to \infty} f(x, s)$$

where f is total recursive in B and there exist at most n values of s for which $f(x, s-1) \neq f(x, s)$. (If $f(x, s-1) \neq f(x, s)$ we say that $f(x, .)$ *changes its mind* at s.) Let $N(x)$ denote the number of times that $f(x, .)$ changes its mind, and let

$$P(x) = N(x) \bmod 2.$$

Note that

$$L(x) = P(x) \oplus f(x, 0),$$

where $\oplus$ denotes the exclusive-or operator. For each k such that $1 \leq k \leq n$ we define an oblivious Turing machine p_k with oracle B that simulates $f(x, s)$ for $s = 1, 2, \ldots$ and halts as soon as $f(x, .)$ makes its kth mind change. Thus p_k halts if and only if $N(x) \geq k$. Then $p_1, \ldots, p_k$ can be effectively computed, and

$$P(x) = \mathrm{PARITY}_n^{B'}(p_1, \ldots, p_n),$$

so that

$$L(x) = \mathrm{PARITY}_n^{B'}(p_1, \ldots, p_n) \oplus f(x, 0).$$

Since f is B-recursive, we can effectively construct a Turing machine p_0 with oracle B that halts iff $f(x, 0) = 1$. Thus

$$L(x) = \mathrm{PARITY}_{n+1}^{B'}(p_0, p_1, \ldots, p_n).$$

ii. This follows because the proof of part (i) was constructive.

▮

We note that the techniques used in the preceding proof can be modified so as to yield a proof by diagonalization of Corollary 47.

4.6. Semi-recursive Sets

In Section 4.3. we showed that nonrecursive semi-recursive sets are supportive. By using different techniques, we show that they are also parallel-supportive.

Semi-recursive sets were defined by Jockusch [Joc68]. The following definition is easily seen to be equivalent to his.

Definition 55 C is semi-recursive if there is a total recursive function f of two arguments such that

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

In some ways semi-recursive sets behave like the halting problem K. (See Lemma 32 and the observation after Definition 34.)

Lemma 56 *If C is semi-recursive then*

i. $\mathrm{GEQ}^C \leq_{\mathrm{m}} C$.

ii. $\mathrm{FQ}_{||}(n, C) \subseteq \mathrm{FQ}(1, \#_n^C)$.

Proof: In [Joc68] Jockusch proved that C is semi-recursive if and only if C is a lower cut of a recursive linear order.

i. Given $x = x_1, \ldots, x_n$ we sort them according to the recursive linear order promised above. Let x'_k be the kth smallest. Then $(k; x) \in \mathrm{GEQ}^C$ if and only if $x'_k \in C$.

ii. Given $x = x_1, \ldots, x_n$ we sort them according to the recursive linear order promised above. Let $x'_1, \ldots, x'_n$ be the sorted sequence. Let $k = \#_n^C(x)$. Then $x'_1 \in C, \ldots, x'_k \in C$ and $x'_{k+1} \notin C, \ldots, x'_n \notin C$. Thus $\mathrm{F}_n^C \in \mathrm{FQ}(1, \#_n^C)$, which implies that $\mathrm{FQ}_{||}(n, C) \subseteq \mathrm{FQ}(1, \#_n^C)$.

▮

Some other properties of semi-recursive sets are presented in [Joc68] and [BGGO93].

Lemma 57 *If C is a nonrecursive, semi-recursive set then*

$$\mathrm{PARITY}_{n+1}^{C} \notin \mathrm{Q}_{||}(n, C).$$

Proof: The proof is essentially the same as the proof of Lemma 45. The only properties of K used in that proof are also true for semi-recursive sets by Lemma 56. ∎

Theorem 58 *If C is a nonrecursive, semi-recursive set then for all n there exists a set that is $(n+1)$-tt reducible to C but not n-wtt reducible to C.*

Proof: By the preceding lemma, $\mathrm{PARITY}_{n+1}^{C}$ is such a set. ∎

Corollary 59 *All nonrecursive semi-recursive sets are parallel-supportive.*

4.7. Generic Sets

For definitions and a discussion of generic sets see [Joc79]. Let A be 1-generic. In [BGGO93] we showed that

$$(\forall k)[A^{k+1} \notin \mathrm{Q}(k, A)],$$

where A^n denotes the cross-product of A with itself n times. Consequently, there is a set that is $(k+1)$-tt reducible to A but not k-Turing reducible to A. In particular, A is supportive and parallel-supportive.

Jockusch proved that the set of 1-generic sets is co-meager [Joc79]. (See [Str81] for more on co-meager sets and Baire category. Stromberg calls co-meager sets "residual.") Hence we have the following corollary:

Corollary 60 *The set of oracles that are both supportive and parallel-supportive is co-meager.*

In other words, most (in the sense of Baire category) sets are supportive and parallel-supportive. In the next section, we will show that most (in the sense of Lebesgue measure) sets are supportive and parallel-supportive.

4.8. Almost All Sets

A measure can be defined on the set of all sets of strings by treating a set of strings as a real number and using Lebesgue measure on reals [BG81]. In this section we show that almost all (*i.e.*, a measure-1 set of) sets of strings are supportive and parallel supportive.

Lemma 61 *For almost all A,* $\mathrm{PARITY}^A_{n+1} \notin \mathrm{Q}(n, A)$.

Proof: Let M be an oracle Turing machine that computes with at most n queries to its oracle A. We show that M correctly computes PARITY^A_{n+1} only for a set of oracles whose measure is 0. For each k consider the behavior of M on input $(0^k1^0, 0^k1^1, \ldots, 0^k1^n)$. Since M makes only n queries to A, M cannot query A about all of the inputs. If we complement the membership of the unqueried input, then the behavior of M is unaffected; however, the value of PARITY^A_{n+1} is complemented. Therefore M computes correctly on this input for exactly half of all oracles. By Bennett and Gill's Lemma 1, the class of oracles A for which there exists a $\mathrm{Q}(n, A)$ machine M that correctly computes PARITY^A_{n+1} has measure 0. ∎

Theorem 62 *For almost all A, for all n, there exists a set that is $(n+1)$-truth-table reducible to A but not n-Turing reducible to A.*

Proof: By the preceding lemma, PARITY^A_{n+1} is such a set. ∎

Corollary 63 *Almost all sets are supportive and parallel-supportive.*

We will produce additional supportive and parallel-supportive sets in Section 6.4..

5. Non-supportive Sets

In this section we will produce sets that are supportive but not parallel-supportive, and nonrecursive sets that are neither supportive nor parallel-supportive. It is open whether there exist sets that are parallel-supportive but not supportive.

5.1. Non-parallel-supportive Sets

In Section 4.5., we showed that every jump is supportive and parallel-supportive. In contrast, in this section we will show that every jump degree contains a set that is supportive, but not parallel-supportive.

Definition 64 A is *1-Turing complete* for a set $\mathcal{S}$ of decision problems if $A \in \mathcal{S}$ and

$$\mathcal{S} \subseteq \mathrm{Q}(1, A).$$

Lemma 65 *If C is 1-Turing complete for $\mathrm{Q}_{||}(\omega, A)$ then C is not parallel-supportive.*

Proof: Assume that C is 1-Turing complete for $\mathrm{Q}_{||}(\omega, A)$. Choose any set $L \leq_{\mathrm{wtt}} C$. Since $C \leq_{\mathrm{wtt}} A$, we have $L \leq_{\mathrm{wtt}} A$, because weak-truth-table reducibility is transitive. Since C is 1-Turing complete for $\mathrm{Q}_{||}(\omega, A)$, we obtain $L \leq_{1\text{-}\mathrm{T}} C$. We have shown that $\mathrm{Q}_{||}(\omega, C) = \mathrm{Q}(1, C)$, so C is not parallel-supportive. ∎

Theorem 66 *For all B, the set $\mathrm{tt}^{B'}$ is supportive but not parallel supportive.*

Proof: We claim that $\mathrm{tt}^{B'}$ is 1-Turing complete for $\mathrm{Q}_{||}(\omega, B')$. Obviously $\mathrm{tt}^{B'} \in \mathrm{Q}_{||}(\omega, B')$. Suppose that $L \in \mathrm{Q}_{||}(\omega, B')$. By Lemma 54(ii), $L \leq_{\mathrm{tt}} B'$, so $L \leq_m \mathrm{tt}^{B'}$. Thus $\mathrm{tt}^{B'}$ is 1-Turing complete for $\mathrm{Q}_{||}(\omega, B')$. By Theorem 65, $\mathrm{tt}^{B'}$ is not parallel-supportive. By Theorem 53, $\mathrm{tt}^{B'}$ is supportive. ∎

Corollary 67 *There exist sets X and Y such that $X \leq_{2\text{-}\mathrm{T}} Y$, but $X \not\leq_{\mathrm{wtt}} Y$.*

Proof: Let $Y = \mathrm{tt}^{B'}$ for some B'. By Theorem 53, Y is supportive, so there is a set $X \in \mathrm{Q}(2, Y) - \mathrm{Q}(1, Y)$. As shown in the proof of Theorem 66, $\mathrm{Q}_{||}(\omega, Y) = \mathrm{Q}(1, Y)$, so $X \in \mathrm{Q}(2, Y) - \mathrm{Q}_{||}(\omega, Y)$. ∎

We note that Lachlan [Lac65] constructed r.e. sets X and Y such that $X \leq_{1\text{-}\mathrm{T}} Y$, but $X \not\leq_{\mathrm{tt}} Y$. It would be interesting if our result above could be extended to r.e. sets X and Y.

Corollary 68 *Every degree $\mathbf{a}$ such that $\mathbf{0}' \leq_{\mathrm{T}} \mathbf{a}$ contains a set that is supportive but not parallel-supportive.*

Proof: Every degree $\mathbf{a}$ such that $\mathbf{0}' \leq_{\mathrm{T}} \mathbf{a}$ is the jump of some degree, say $\mathbf{b}$ [Rog67]. Let $B \in \mathbf{b}$; therefore $B' \in \mathbf{a}$, so $\mathrm{tt}^{B'} \in \mathbf{a}$. By the preceding lemma, $\mathrm{tt}^{B'}$ is supportive but not parallel-supportive. ∎

5.2. Hyperimmune-Free Degrees

For background on hyperimmunity see [Rog67, Soa87]. A degree $\mathbf{a}$ is called hyperimmune if $\mathbf{a}$ contains a hyperimmune set; $\mathbf{a}$ is called *hyperimmune-free* otherwise. The following characterization of hyperimmune-free degrees is due to Jockusch [Joc69] and Martin. Let $\deg(A)$ denote the Turing-degree of the set A.

Proposition 69 $\deg(A)$ *is hyperimmune-free if and only if*

$$(\forall B)[B \leq_{\mathrm{T}} A \Longleftrightarrow B \leq_{\mathrm{tt}} A].$$

Recall Definition 43 (tt^A).

Lemma 70 *Assume that* $\deg(A)$ *is hyperimmune-free. Then*

$$B \leq_{\mathrm{T}} \mathrm{tt}^A \Longleftrightarrow B \leq_m \mathrm{tt}^A.$$

Proof: $B \leq_{\mathrm{T}} \mathrm{tt}^A \Longrightarrow B \leq_{\mathrm{T}} A \Longrightarrow B \leq_{\mathrm{tt}} A \Longrightarrow B \leq_m \mathrm{tt}^A$. ∎

Theorem 71 *Every hyperimmune-free degree contains a set that is neither supportive nor parallel-supportive.*

Proof: If $\deg(A)$ is hyperimmune-free then tt^A is such a set. ∎

Corollary 72 *There exists a nonrecursive set that is neither supportive nor parallel-supportive.*

Proof: A nonrecursive hyperimmune-free degree was produced in [MM68] using a tree construction. ∎

We note that the degree exhibited by Miller and Martin is $< \mathbf{0}''$. In the next section, we will construct a subclass of the hyperimmune degrees $\leq \mathbf{0}''$ containing some sets that are neither supportive nor parallel-supportive.

6. Some Hyperimmune Degrees Contain Nonsupportive Sets

In the previous section we saw that every hyperimmune-free degree contains a set that is neither supportive nor parallel-supportive. In this section, we will define a strong form of hyperimmunity: mish-immunity with respect to C (w.r.t. C), where C is an oracle. We show that if A is bounded mish-immune w.r.t. $\emptyset'$ then A is supportive and parallel-supportive. However, if A is mish-immune w.r.t. $\emptyset'$ then A is tt-equivalent to a set that is neither supportive nor parallel-supportive. Since hyperimmune-freeness allows us to easily construct a nonsupportive set, it is surprising that this strong form of hyperimmunity does so also. As a consequence of work by Kurtz [Kur83], we will show that every weakly 2-generic degree contains a set that is neither supportive nor parallel-supportive.

The following is equivalent to the standard definition of immunity (because every infinite r.e. set contains an infinite recursive set [Rog67, Soa87]).

Definition 73 An infinite set A is *immune* if A does not contain an infinite recursive set.

Equivalently, A is immune iff there does not exist a total recursive predicate R such that

- $(\exists^\infty n)[R(n)]$, and
- $(\forall n)[R(n) \Longrightarrow n \in A]$.

In other words an infinite set A is immune if it is not possible to determine that $n \in A$ infinitely often, without error. We say that an infinite set A is initial-segment immune if it is not possible to determine that $n \in A$ infinitely often, without error, even if we know all smaller elements of A.

Definition 74 Let $A[s]$ denote the initial segment $(A(1), \ldots, A(s))$. An infinite set A is *initial-segment immune (i.s.-immune)* if there does not exist a total recursive predicate R such that

- $(\exists^\infty n)[R(A[n-1])]$, and
- $(\forall n)[R(A(n-1)) \Longrightarrow n \in A]$.

There are several standard definitions of hyperimmunity [Rog67, Soa87]. The following definition is easily seen to be equivalent to them.

Definition 75 Let D_n denote the finite set whose canonical representation is n, *i.e.*, the characteristic sequence of D_n is equal to the binary representation of n. An infinite set A is *hyperimmune* if there does not exist a total recursive function g such that

- for infinitely many i, $\min(D_{g(i)}) > i$, and
- for all i, $D_{g(i)} \cap A \neq \emptyset$.

The following proposition is well known and was stated by Martin in [Mar67]:

Proposition 76 *The class of hyperimmune sets has measure 0.*

Proof: Define g so that

$$D_{g(i)} = \{i^2 + 1, \ldots, (i+1)^2\}.$$

The probability that $D_{g(i)} \cap A = \emptyset$ is $2^{-(2i+1)}$, so the probability that A is hyperimmune is at most

$$\sum_{i \geq 0} 2^{-(2i+1)} = 2/3.$$

By the 0-1 law for tail events [Bre68], the probability that A is hyperimmune is either 0 or 1, so the probability must be 0. ∎

We define i.s.-hyperimmunity by analogy with i.s.-immunity.

Definition 77 An infinite set A is *i.s.-hyperimmune* if there does not exist a total recursive function g such that

- for infinitely many i, $\min(D_{g(A[i])}) > i$, and
- for all i, $D_{g(A[i])} \cap A \neq \emptyset$.

We note that if "infinitely many i" were replaced by "for all i" in the definition of i.s.-hyperimmunity, then we would obtain yet another definition of hyperimmunity. The details are left to the reader (let $D_{g'(i)} = \{i+1, i+2, \ldots\} \cap (\cup_{|s|=i} D_{g(\sigma)})$). We do not know whether every hyperimmune set is i.s.-hyperimmune. However, it is obvious that all i.s.-hyperimmune sets are hyperimmune, and we will see in Section 6.3. that every hyperimmune *degree* is i.s.-hyperimmune.

6.1. Bi- and Multi-Hyperimmunity

Recall that a set A is *bi-immune* iff A and $\bar{A}$ are immune. Equivalently, A is bi-immune iff the set $A \Delta B$ is immune for all B, where $A \Delta B$ denotes the *symmetric difference* $(A - B) \cup (B - A)$. Recall that a set A is *bi-hyperimmune* iff A and $\bar{A}$ are hyperimmune. We say that a set A is *multi-hyperimmune* if $A \Delta B$ is hyperimmune for all B. In defining multi-initial-segment-hyperimmunity (mish-immunity), we will let the predicate B depend on the initial segment.

Kurtz [Kur83] has shown that every hyperimmune degree is bi-hyperimmune. A similar proof, sketched at the end of Section 6.3., shows that every hyperimmune degree is also mish-immune.

In Section 6.2., we will see that mish-immunity can be thought of in terms of gambling systems, which were used by Church in defining random sets [Rog67, Exercise 12-34].

Definition 78 An infinite set A is *multi-i.s.-hyperimmune (mish-immune)* if there does not exist a total recursive function g and a total recursive predicate B such that

- for infinitely many i, $\min(D_{g(A[i])}) > i$, and
- for all i, $(\exists x \in D_{g(A[i])})[A(x) = B(A[x-1], x)]$.

Note that we obtain an equivalent definition if "infinitely many" is replaced by "all" and/or if "$A[x-1]$" is replaced by "$A[i]$." It is an open question to determine whether all multi-hyperimmune sets are mish-immune. However, the techniques of [Kur83] show that every hyperimmune degree is mish-immune. In Section 6.4., we will see that every set that is mish-immune with respect to $\emptyset'$ is tt-equivalent to a set that is neither supportive nor parallel-supportive, and thus hyperimmune degrees can contain nonsupportive sets.

Corollary 79

i. The class of multi-hyperimmune sets has measure 0.

ii. The class of mish-immune sets has measure 0.

Proof: This follows from Proposition 76. ∎

Although uncommon, mish-immune sets are easy to construct:

Theorem 80 *There exists a mish-immune set that is recursive in $\emptyset'$.*

Proof: The desired set can be constructed via an easy initial segment argument. ∎

6.2. Bounded Hyperimmunity

In this section we define bounded hyperimmune sets, and we show that almost all sets are bounded hyperimmune. In Section 6.4., we will see that if A is bounded mish-immune w.r.t. $\emptyset'$ then A is supportive and parallel-supportive.

We define bounded hyperimmunity by bounding the cardinality of $D_{g(i)}$ in the definition of hyperimmunity.

Definition 81 An infinite set A is *bounded hyperimmune* if there does not exist a total recursive function g and a constant k such that

- for infinitely many i, $\min(D_{g(i)}) > i$,
- for all i, $D_{g(i)} \cap A \neq \emptyset$, and
- for all i, $|D_{g(i)}| \leq k$.

Definition 82 An infinite set A is *bounded i.s.-hyperimmune* if there does not exist a total recursive function g and a constant k such that

- for infinitely many i, $\min(D_{g(A[i])}) > i$,
- for all i, $D_{g(A[i])} \cap A \neq \emptyset$.
- for all i, $|D_{g(A[i])}| \leq k$.

Definition 83 An infinite set A is *bounded mish-immune* if there does not exist a total recursive function g, a constant k, and a total recursive predicate B such that

- for infinitely many i, $\min(D_{g(A[i])}) > i$,

- for all i, $(\exists x \in D_{g(A[i])})[A(x) = B(A[x-1], x)]$, and
- for all i, $|D_{g(A[i])}| \leq k$.

Theorem 84 *Almost all sets are bounded mish-immune.*

Proof: Let B be any total recursive predicate. Let g be a total recursive function such that for some fixed k

$$(\forall i)[|D_{g(A[i])}| \leq k].$$

Let e_i be the event that

$$(\exists x \in D_{g(A[i])})[A(x) = B(A[x-1], x)].$$

Then $p_i \leq 1 - 2^{-k}$. Suppose that $i_1, i_2, \ldots,$ is an infinite sequence such that each for each i_j we have $\min(D_{g(A[i_j])}) > i_j$. Then let $i'_1 = i_1$ and i'_{j+1} be the least i_j greater than $\max(D_{g(A[i'_j])})$; thus the sets $D_{g(A[i'_1])}$, $D_{g(A[i'_2])}$, …, are pairwise disjoint. Then the events $e_{i'_1}, e_{i'_2}, \ldots$, are independent. Therefore, the probability that all those events occur is $(1-2^{-k})^{\infty} = 0$. Summing over all g and B (countably many), we find that the probability that A is not bounded mish-immune is 0. ∎

Mish-hyperimmunity can be thought of in terms of gambling systems. Imagine a game of chance in which the house exposes the characteristic sequence of a set A one bit at a time, always accepting even-money bets on the next unseen bit. If A is not mish-immune then a gambler might proceed as follows: If the smallest element of $D_{g(A[i])}$ is greater than i then the gambler bets one dollar on that number; if he loses then he bets two dollars on the second-smallest element of $D_{g(A[i])}$; etc. The predicate B tells him whether to bet that the next bit is 0 or that the next bit is 1. The system is applicable for infinitely many i, and the gambler makes one dollar each time the system is applicable. Thus, with an initial stake of $2^{k+1} - 1$ dollars, the gambler can win an infinite amount of money. By the well-known principle of gambler's ruin [Bre68] this occurs with probability 0. That is why almost all sets are bounded mish-immune. Unbounded mish-immunity can be thought of in terms of a gambler with infinite credit; he can expect his winnings to oscillate between positive and negative infinity.

6.3. Relativizations

The definitions of immunity, hyperimmunity, i.s.-immunity, and i.s.-hyperimmunity can be relativized to an oracle C by allowing the function g to be recursive in C. The definitions of multi-hyperimmunity and mish-immunity can be relativized to an oracle C by allowing g and B to be recursive in C. We call the relativized concepts *immunity with respect to C (w.r.t. C), etc.* We leave the details of the relativizations to the reader.

In subsequent sections we will usually take C to be an oracle for the halting problem $\emptyset'$. Because this allows us to guarantee convergence, we will be able to convert n-Turing reductions into truth-table reductions, as we did with hyperimmune-free degrees.

By relativizing the proof of Theorem 80 we obtain:

Theorem 85 *For every set C there exists a set that is mish-immune w.r.t. C and recursive in C'.*

Generic sets are discussed in [Joc79, Kur83]. Because we can construct a set mish-immune w.r.t. C via an initial segment argument, we have the following:

Corollary 86 *For all arithmetic sets C, all generic sets are mish-immune w.r.t. C.*

By relativizing Theorem 84, we obtain

Theorem 87 *For all C, almost all sets are bounded hyperimmune w.r.t. C.*

In [Kur83], Kurtz showed that a degree $\mathbf{a}$ is the nth jump of a weakly $(n+1)$-generic degree (see his paper for a definition) iff $\mathbf{a}$ is hyperimmune w.r.t. $\emptyset^{(n)}$ and $\mathbf{0^{(n)}} \leq_{\mathrm{T}} \mathbf{a}$. It is easy to see that weakly $(n+1)$-generic sets are mish-immune w.r.t. $\emptyset^{(n)}$, so that every degree above $\mathbf{0^{(n)}}$ that is hyperimmune w.r.t. $\emptyset^{(n)}$ is also mish-immune w.r.t. $\emptyset^{(n)}$. Letting $n = 0$, we see that every hyperimmune degree is mish-immune, as claimed in Section 6.1.. However, for $n \geq 1$, it is not known whether every degree that is hyperimmune w.r.t. $\emptyset^{(n)}$ is also mish-immune w.r.t. $\emptyset^{(n)}$.

6.4. A Nonsupportive Set in a Hyperimmune Degree

In this section we show that if a set A is mish-immune w.r.t. $\emptyset'$, then A is supportive and parallel-supportive. In contrast, there exists a set tt-equivalent to A that is neither supportive nor parallel-supportive. Since A is hyperimmune, this is a surprising contrast to Theorem 71.

Lemma 88 *Let A be bounded mish-immune w.r.t. $\emptyset'$.*

i. A function f is n-Turing reducible to A iff f is n-strong-Turing reducible to A.

ii. A function f is n-wtt reducible to A iff f is n-tt reducible to A.

Proof:

i. Let M be a $\mathrm{Q}(n, A)$ machine that computes f. We need to show that there is a $\mathrm{Q}(n, A)$ machine M' that also computes the characteristic function of f, but such that M' does not diverge even if given incorrect oracle answers. We claim that there is an initial segment σ of A such that M is total whenever the oracle is an extension of σ. The claim implies that we can produce M' by building a finite table of oracle answers into M. Thus f is n-strong-Turing reducible to A. The converse is obvious.

   We prove the claim by contradiction. Suppose that every initial segment $A[i]$ has an extension that causes M to be nontotal. We show that A is not mish-immune w.r.t. $\emptyset'$ by constructing g and B as follows:

   > Using an oracle for the halting set $\emptyset'$, simulate M on all inputs and all possible sequences of oracle answers that are consistent with $A[i]$, stopping when we find a sequence of oracle queries $q_1, \ldots, q_n$ and answers $b_1, \ldots, b_n$ respectively that makes M diverge. Because M must halt when using correct oracle answers, one of the oracle answers must be incorrect. Let $D_{g(A[i])}$ contain those queries greater than i. For each $q_j \in D_{g(A[i])}$, let $B(A[q_j - 1], q_j) = 1 - b_j$.

ii. Similar to (i).

∎

This ability to guarantee that our reductions converge for all oracles will allow us to show that all sets that are bounded mish-immune w.r.t. $\emptyset'$ are parallel-supportive. A fortiori, all sets A that are mish-immune w.r.t. $\emptyset'$ are parallel-supportive. By straightforward techniques we will extend the result to show supportiveness. Surprisingly, we will use similar techniques to show that tt^A is neither supportive nor parallel-supportive.

Lemma 89 *Let A be bounded mish-immune w.r.t. $\emptyset'$. Then A is parallel-supportive.*

Proof: We proceed by contradiction. Suppose that A is not parallel-supportive. By the u_A Lemma,

$$(\exists k)(\forall n)[\mathrm{PARITY}_n^A \leq_{k\text{-wtt}} A].$$

For each n, define a set S_n such that

$$S_n(t) = \mathrm{PARITY}_n^A(t, t+1, \ldots, t+n-1).$$

Then $(\exists k)[S_{k+1} \leq_{k\text{-wtt}} A]$. Choose the least such k. If $k = 0$ then A is recursive, which is impossible because A is immune, so we can assume that $k \geq 1$. By Lemma 88, $S_{k+1} \leq_{k\text{-tt}} A$.

Let S_{k+1} be k-tt reducible to A via Turing machine M. We write $D \supset \sigma$ to denote that the set D is an extension of the string σ, *i.e.*, the string σ is a prefix of the characteristic sequence of the set D. We say that q *affects the value of* $M^A(t)$ if

$$(\exists S \supset A[q-1])[M^S(t) \neq M^A(t)].$$

Let $q(t)$ be the largest query that affects the value of $M(t)$; the maximum exists and can be determined effectively because M computes a tt-reduction. We consider three cases:

Case 1: ($q(t) < t + k$ infinitely often.) If $q(t) < t + k$ we can compute $\mathrm{PARITY}_{k+1}^A(t, t+1, \ldots, t+k)$ from $A[t+k-1]$. Since

$$A(t+k) = \mathrm{PARITY}_{k+1}^A(t, t+1, \ldots, t+k) \oplus \mathrm{PARITY}_{k+1}^A(t, t+1, \ldots, t+k-1),$$

we can compute $A(t+k)$ from $A[t+k-1]$. Therefore A is not bi-i.s.-immune, a contradiction.

Case 2: **($q(t) > t + k$ infinitely often.)** We will show that A is not bounded mish-immune, by defining g and B as follows:

> Let t be any number such that $q(t) > t + k$. Let $i = t + k$. Using only $A[i]$, compute $S_{k+1}(t)$. Let $D_{g(A[i])}$ be equal to the set of queries greater than i made by $M^A(t)$. We construct B via a technique related to forcing (see [Soa87]). We say that $A(x) = 1$ *forces* $M^A(t) = v$ if
>
> (a) $(\forall D \supset A[x-1] \cup \{x\})[M^D(t) = v]$, and
>
> (b) $(\exists D \supset A[x-1])[M^D(t) \neq v]$.
>
> We define $A(x) = 0$ *forces* $M^A(t) = v$ similarly. By condition (b) it cannot happen that $A(x) = 1$ forces $M^A(t) = v$ and $A(x) = 0$ also forces $M^A(t) = v$. For each $x \in D_{g(A[i])}$, let
>
> $$B(A[x-1], x) = \begin{cases} 0 & \text{if } A(x) = 0 \text{ forces } M^A(t) = S_{k+1}(t) \\ 1 & \text{if } A(x) = 1 \text{ forces } M^A(t) = S_{k+1}(t) \\ \text{arbitrary} & \text{otherwise.} \end{cases}$$

Because M^A computes S_{k+1}, there must exist an x queried by $M^A(t)$ such that $A(x) = 1$ forces $M^A(t) = S_{k+1}(t)$ or such that $A(x) = 0$ forces $M^A(t) = S_{k+1}(t)$. If y is the largest such x, then $A(y) = B(A[y-1], y)$ as required.

Case 3: **($q(t) = t + k$ almost always)** In this case we obtain a contradiction by showing that $S_k \in \mathrm{Q}_{||}(k-1, A)$. Without loss of generality, assume that $q(t) = t + k$ for all t (we can build a finite table into M to cover the finite set of exceptions). We compute $S_k(t)$ as follows: Run M on input t until M produces its truth table of norm k. Since the largest query that affects the value of $M^A(t)$ is $t + k$, M must query $t+k$; therefore M cannot make more than $k-1$ queries less than (other than) $t+k$. We make all queries less than $t+k$, plug their answers into the truth table, and simplify it. The result is a nonconstant Boolean function of the single variable $A(t+k)$. The function must be either the identity function or its complement. If the function is the identity function then $S_{k+1}(t) = A(t+k)$. Since $S_{k+1}(t) = S_k(t) \oplus A(t+k)$, it follows that $S_k(t) = 0$. If the function is the complement of the identity function then $S_k(t) = 1$. Thus $S_k \in \mathrm{Q}_{||}(k-1, A)$, contradicting the minimality of k.

$\blacksquare$

Essentially the same proof shows that if A is bounded mish-immune then there is a set $(n+1)$-tt reducible to A that is not n-tt reducible to A.

Lemma 90 *If L is n-strong-Turing reducible to A then L is $(2^n - 1)$-tt reducible to A.*

Proof: Assume that L is n-strong-Turing reducible to A via a machine M. We can define a new machine that simulates A for all possible oracle answers and makes all $2^n - 1$ possible queries in parallel. $\blacksquare$

Lemma 91 *If A is mish-immune w.r.t. $\emptyset'$ then* $\mathrm{Q}(n, A) \subseteq \mathrm{Q}_{||}(2^n - 1, A)$.

Proof: Assume that L is n-Turing reducible to A. By Lemma 88(i), L is n-strong-Turing reducible to A. By Lemma 90, L is $(2^n - 1)$-tt reducible to A. $\blacksquare$

Theorem 92 *If A is bounded mish-immune w.r.t. $\emptyset'$ then A is supportive and parallel-supportive.*

Proof: Suppose that A is not supportive. By the u_A Lemma,

$$(\exists m)(\forall n)[\mathrm{PARITY}_n^A \in \mathrm{Q}(m, A)].$$

Therefore, by Lemma 91,

$$(\exists m)(\forall n)[\mathrm{PARITY}_n^A \in \mathrm{Q}_{||}(2^m - 1, A)].$$

However, in the proof of Lemma 89, we showed that that statement is false. The contradiction shows that A is supportive. $\blacksquare$

Because almost all sets are bounded mish-immune w.r.t. $\emptyset'$ by Theorem 84, this provides another proof that almost all sets are supportive (Theorem 63). Since bounded mish-immunity implies mish-immunity, we have the following corollary:

Corollary 93 *If A is mish-immune w.r.t. $\emptyset'$ then A is supportive and parallel-supportive.*

The next theorem is similar to Proposition 69. It will allow us to show for all sets mish-immune w.r.t. $\emptyset'$ that tt^A is neither supportive nor parallel-supportive.

Theorem 94 *If A is mish-immune w.r.t. $\emptyset'$ then*

$$f \leq_{\mathrm{bT}} \mathrm{tt}^A \Longleftrightarrow f \leq_{\mathrm{tt}} A.$$

Proof: The proof is similar to the proof of Lemma 88. It suffices to show there exists some i such that M^{tt^C} converges whenever C is an extension of $A[i]$ (then we can encode the finite table of answers into M). If such an i does not exist, then we show that A is not mish-immune w.r.t. $\emptyset'$ by constructing g and B recursive in $\emptyset'$ as follows:

> Treat M as a machine that reduces to the oracle A, and simulate M on all inputs and all possible sequences of oracle answers that are consistent with $A[i]$, stopping when we find a sequence of oracle queries $q_1, \ldots, q_n$ and answers $b_1, \ldots, b_n$ respectively that makes M diverge. Because M must halt when using correct oracle answers, one of the oracle answers must be incorrect. Let $D_{g(A[i])}$ be the set of queries in the sequence that are greater than i. Let $B(A[q_j - 1], q_j) = 1 - b_j$ for each $q_j \in D_{g(A[i])}$.

The converse is obvious. ∎

Note that if we assumed only that $f \leq_{\mathrm{T}} A$ in the preceding theorem, then the proof would break down because there might not be any *finite* sequence of oracle answers that forces M to diverge. In fact, in Corollary 96, we will see the it is not possible to extend the theorem in this way.

Corollary 95 *If A is mish-immune w.r.t. $\emptyset'$ then tt^A is neither supportive nor parallel-supportive.*

Proof: Assume that A is mish-immune w.r.t. $\emptyset'$. By the preceding theorem, if $L \leq_{\mathrm{bT}} \mathrm{tt}^A$ then $L \leq_{\mathrm{tt}} A$, so $L \leq_m \mathrm{tt}^A$. Therefore for all n, $\mathrm{Q}(n, \mathrm{tt}^A) = \mathrm{Q}(1, \mathrm{tt}^A)$. ∎

The collapse proved above does not extend to unbounded Turing reductions.

Corollary 96 *If A is mish-immune w.r.t. $\emptyset'$ then for all n*

$$\mathrm{Q}(1, \mathrm{tt}^A) = \mathrm{Q}(n, \mathrm{tt}^A) \subset \mathrm{Q}(\omega, \mathrm{tt}^A).$$

Proof: By Theorem 94, every set bounded-Turing reducible to tt^A is tt-reducible to A, proving the equality. Since A is hyperimmune, there exists a set Turing-reducible to A that is not tt-reducible to A by Proposition 69, proving the inequality. ▮

7. Conclusions

In section 3.2. we showed that if A is nonrecursive, then there is a total function in $\mathrm{FQ}_{||}(n+1, A) - \mathrm{FQ}_{||}(n, A)$, and there is a total function in $\mathrm{FQ}(n+1, A) - \mathrm{FQ}(n, A)$. The intuition behind our results is that $n+1$ oracle answers encode more information than n answers. Because the results seem "intuitively obvious," it would be interesting to find proofs that do not depend on the Nonspeedup Theorem.

There is no analogous separation between $\mathrm{Q}_{||}(n+1, A)$ and $\mathrm{Q}_{||}(n, A)$, or between $\mathrm{Q}(n+1, A)$ and $\mathrm{Q}(n, A)$; intuitively, the answer to a decision problem can convey only one bit of information, no matter how much information was required for its solution. In technical terms, the methods in this paper do not help us to construct a set in $\mathrm{Q}_{||}(n+1, A) - \mathrm{Q}_{||}(n, A)$ or in $\mathrm{Q}(n+1, A) - \mathrm{Q}(n, A)$, because there is no analogue of the Nonspeedup theorem that applies to $\mathrm{Q}_{||}$ and Q.

In fact, in Section 5. we showed that every degree above $\mathbf{0}'$ contains a set that is supportive but not parallel-supportive. We also showed that every hyperimmune-free degree contains a set that is neither supportive nor parallel-supportive, and we defined certain subclasses of the hyperimmune degrees with the same property. Thus, nonrecursive sets need not be supportive or parallel-supportive, in contrast with the results in Section 3.2..

On the other hand, we were able to show that many sets are supportive and parallel-supportive: namely generic sets, nonrecursive semi-recursive sets, nonrecursive r.e. sets, nonrecursive n-r.e. sets, jumps of sets, and almost all sets. In particular, every tt-degree contains a set that is supportive and parallel-supportive. We also showed that all nonrecursive nonsuperterse sets are supportive.

8. Open Questions

- When is there a function f that is $(n+1)$-strong-Turing reducible to A, but not n-Turing reducible to A?

- If $(\forall m)(\exists n)[\mathrm{tt}_n^A \notin \mathrm{Q}(m, A)]$ then we know that A is supportive by part (iii) of the u_A Lemma (17). Can we conclude that A is also parallel-supportive? This would imply that nonrecursive nonsuperterse sets are parallel-supportive.

- On the other hand, does there exist a set that is neither recursive, parallel-supportive, nor superterse?

- In Section 4.5., we saw that the converses to parts (ii) and (iii) of the u_A Lemma are false. Does part (i) of the u_A Lemma have a valid converse?

- Is there a non-supportive set A such that $0 < A \leq_{\mathrm{T}} \emptyset'$?

- Is there a parallel-supportive set that is not supportive?

- Do there exist r.e. sets X and Y such that $X \leq_{2\text{-T}} Y$ but $X \not\leq_{\mathrm{wtt}} Y$?

- Are there any degrees that contain only supportive sets? (Such a degree would have to be hyperimmune but not mish-immune w.r.t. $\emptyset'$.)

- Does there exist a hyperimmune set that is not i.s.-hyperimmune? a multi-hyperimmune set that is not mish-immune?

- Is there a degree that is hyperimmune w.r.t. $\emptyset'$, but not mish-immune w.r.t. $\emptyset'$? not multi-immune w.r.t. $\emptyset'$?

- In [Bei87b, Lemma 4.4.5], we showed that if $K \leq_{1\text{-wtt}} A$ then for all n, $\mathrm{Q}(n, A) \subset \mathrm{Q}(2n+1, A)$. Louise Hay poses the following question: Does $K \leq_1 A$ imply that $\mathrm{Q}(n, A) \subset \mathrm{Q}(n+1, A)$? (For the special case when A is a jump, the separation is established by Theorem 39.) On the other hand, does there exist a set A such that $\mathrm{Q}(1, A) = \mathrm{Q}(2, A) \subset \mathrm{Q}(3, A)$?

9. Acknowledgments

I am grateful to my advisor, John Gill, for helpful discussions of the Non-speedup Theorem over cocoa and coffee; to my friend and colleague, Bill Gasarch, for suggesting the supportive sets problem, for proofreading nearly final drafts of this paper, and for helpful discussions; to Gasarch and Jim Owings for pointing out Jockusch's and Martin's characterization of hyperimmune-free degrees (Proposition 69); and to Stuart Kurtz for helpful discussions. The comments from people who read early drafts of this paper and of [Bei87a, Bei87d, BG87] have been extremely helpful — special thanks go to Gasarch, Owings, Louise Hay, Cathy Schevon, and Dwight Wilson.

References

[Bei87a] Richard Beigel. Functionally supportive sets. Technical Report 87-10, The Johns Hopkins University, Dept. of Computer Science, 1987.

[Bei87b] Richard Beigel. *Query-Limited Reducibilities*. PhD thesis, Stanford University, 1987. Also available as Report No. STAN-CS-88-1221.

[Bei87c] Richard Beigel. A structural theorem that depends quantitatively on the complexity of SAT. In *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory*, pages 28–32. IEEE Computer Society Press, June 1987.

[Bei87d] Richard Beigel. Supportive sets — II. Technical Report 87-15, The Johns Hopkins University, Dept. of Computer Science, 1987.

[Bei91] Richard Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, July 1991.

[BG81] Charles H. Bennett and John Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. *SICOMP*, 10:96–112, February 1981.

[BG87] Richard Beigel and William I. Gasarch. Supportive and parallel-supportive sets. Technical Report 1805, University of Maryland, Dept. of Computer Science, 1987.

[BGGO93] Richard Beigel, William I. Gasarch, John T. Gill, and James C. Owings. Terse, superterse, and verbose sets. *Inf. & Comp.*, 103:68–85, 1993.

[BGH89] Richard Beigel, William I. Gasarch, and Louise Hay. Bounded query classes and the difference hierarchy. *Archive for Mathematical Logic*, 29(2):69–84, December 1989.

[Bre68] Leo Breiman. *Probability*. Addison-Wesley Series in Statistics. Addison-Wesley, Reading, MA, 1968.

[EHK81] Richard L. Epstein, Richard Haas, and Richard 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. Volume 859 of *Lecture Notes in Mathematics*.

[Eps79] Richard L. Epstein. *Degrees of Unsolvability: Structure and Theory*, volume 759 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1979.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.

[Joc68] C. G. Jockusch, Jr. Semirecursive sets and positive reducibility. *T. AMS*, 131:420–436, May 1968.

[Joc69] C. G. Jockusch, Jr. Relationships between reducibilities. *T. AMS*, 142:229–237, 1969.

[Joc79] C. G. Jockusch, Jr. Degrees of generic sets. In *Recursion Theory: its Generalisations and Applications*, Proceedings of Logic, pages 140–157, London, August 1979. Cambridge University Press.

[Kur83] S. A. Kurtz. Notions of weak genericity. *JSL*, 48:764–770, September 1983.

[Lac65] Alistair H. Lachlan. Some notions of reducibility and productiveness. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 11:17–44, 1965.

[Mar67] Donald A. Martin. Measure, category, and degrees of unsolvability, ca. 1966-67. Unpublished manuscript.

[MM68] W. Miller and D. A. Martin. The degree of hyperimmune sets. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 14:159–166, 1968.

[Rog67] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967.

[Soa87] Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.

[Str81] Karl R. Stromberg. *An Introduction to Classical Real Analysis.* Wadsworth International Mathematics Series, Belmont, California, 1981.