ILLiad Request Printout

Transaction Number: 427390
Username: 100934397 Name: William Gasarch
ISSN/ISBN: 0890-5401
NotWantedAfter: 11/09/2009
Accept Non English: Yes
Accept Alternate Edition: No
Request Type: Article - Article

Loan Information

LoanAuthor:
LoanTitle:
LoanPublisher:
LoanPlace:
LoanDate:
LoanEdition:
NotWantedAfter: 11/09/2009

Article Information

PhotoJournalTitle: Information and computation
PhotoJournalVolume: 77
PhotoJournalIssue: 1
Month:
Year: 1988
Pages:
Article Author: Amir and Gasarch
Article Title: Polynomial Terse Sets

Citation Information

Cited In:
Cited Title:
Cited Date:
Cited Volume:
Cited Pages:

OCLC Information

ILL Number:
OCLC Number:
Lending String: Direct Request
Original Loan Author:
Original Loan Title:
Old Journal Title:
Call Number: UMCP EPSL Periodical Stacks Q350.I51 v.76-v.77 (1988)
Location:

Notes

9/11/2009 7:41:58 AM System 1. No Matching Bib/2. ISSN Search With Too Many Hits.

INFORMATION AND COMPUTATION 77, 37–56 (1988)

Polynomial Terse Sets

AMIHOOD AMIR* AND WILLIAM I. GASARCH

Department of Computer Science, and
**Institute for Advanced Computer Studies,*
University of Maryland, College Park, Maryland 20742

Let A be a set and $k \in N$ be such that we wish to know the answers to $x_1 \in A$?, $x_2 \in A$?, ..., $x_k \in A$? for various k-tuples $\langle x_1, x_2, ..., x_k \rangle$. If this problem *requires* k queries to A in order to be solved in polynomial time then A is called *polynomial terse* or *pterse*. We show the existence of both arbitrarily complex pterse and non-pterse sets; and that $P \neq NP$ iff every NP-complete set is pterse. We also show connections with p-immunity, p-selective, p-generic sets, and the boolean hierarchy. In our framework unique satisfiability (and a variation of it called kSAT is, in some sense, "close" to satisfiability. © 1988 Academic Press, Inc.

1. INTRODUCTION

NP-complete problems are considered hard or "intractable." However, problems appearing in the real world cannot be chosen because of their ease, and practical concerns may necessitate solving "intractable" problem.

We consider a partial solution by "bounded query optimization." The idea is the following: Assume S is a hard but useful problem for which we wish to solve many instances. Further assume it is not a real time problem, so a solution is not needed immediately. If S is such that when presented with k instances, there is *one* new instance whose solution gives enough information to (quickly) solve the k instances, then a good strategy would be to batch k queries, and solve the one that helps yield the other solutions.

Note that there are many optimistic assumptions in this scenario, and the following questions need to be answered:

- Do such "hard" sets, that give many answers for the "price" of one, exist?
- If they do, are they natural? Can we classify them?

Roughly speaking, if membership in a set does not allow "quick" decisions about other members, we call the set *polynomial terse*. We show that there are arbitrarily complex polynomial terse sets, and that there are arbitrarily complex sets that are not polynomial terse. The big challenge is to find out which sets are polynomial terse and which sets are not.

0890-5401/88 $3.00

Bounded queries can also be used to classify the complexity of functions. Krentel [19] and Gasarch [11] have classified many functions as being complete in certain bounded query classes. For example, (1) Krentel has shown that the traveling salesman function (which returns the cost of the minimum tour) is complete for the set of all functions that can be computed with a polynomial number of queries to a SAT oracle, and (2) Gasarch has shown that the minimum automata function (on input two finite sets S and T return the size of the minimal deterministic finite automata that accepts S and rejects T) is complete for the set of functions that can be computed with a logarithmic number of queries to a SAT oracle. Other problems remain to be classified.

Extensive work about terseness in the recursion-theoretic context has been done by Beigel, Gasarch, Gill, Hay, and Owings [6, 7, 5]. Most of the proofs and techniques used in these papers do not translate directly into a polynomial framework; and some of the theorems that hold in a recursion-theoretic framework do not hold in ours.

In Section 2 we define the notion of pterseness; informally, a set is pterse if k questions must be asked in order to obtain k answers. In Sections 3, 4, 5, and 8 we relate the notion of pterseness to complexity, bi-immunity, p-genericity, NP-completeness, p-selectivity, and polynomial reals. In Section 6 we look at kSAT (the set of formulas that have exactly k satisfying assignments) in our framework and prove that "not too far" from being NP. In Section 7 we compare some of the complexity classes we define with the Boolean hierarchy of Cai and Hemachandra [10].

2. Definitions, Notation, and Conventions

We are interested in classifying sets and functions that can be computed with a bounded number of queries to an oracle. In particular we are concerned with the function that computes membership in some set A of each element in a given k-tuple. Formally:

Definition. If A is any set and $k \in N$ then

$$F_k^A(x_1, ..., x_k) = \langle \chi_A(x_1), ..., \chi_A(x_k) \rangle$$

(where χ_A is the characteristic function for A).

Some notation, definitions, and conventions follow:

Definition. If g is a function and $k \in N$ then $Q(k, g)$ is the class of sets that can be decided by a polynomial oracle Turing machine with function oracle g that makes at most k queries
$Q(k, A)$.

The class $Q(k, A)$ is related to tru
$B \in Q(k, A)$ then $B \leqslant_{(2^k-1)-tt} A$. The v
we deal with functions in general, and
sets and with how the different types

Definition. If g is a function and
functions that can be computed by a
with function oracle g, that makes a
denote this class $FQ(k, A)$.

Convention. Let k be any natura
NP-complete then $Q(k, A)$ and $Q(k,$
sets $Q(k, \text{NPC})$. Alsó note that in tern
the functions F_k^A and F_k^B are equally h
F_k^{NPC} in the following way: Whateve
where A is any NP-complete set.

We are interested in finding out wh
to A. In recursion theory (no time b
[6]. In a polynomial framework we c

Definition. Let $i \geqslant 2$. A set A is
polynomial time by an M^A machine
every input. A set is *pterse* if it is i-pt

All machines mentioned are assum
oracle machine where the oracle has n
phrase "run $M^{(\ )}(x)$ along every possi
a query arises, and then run the mach
the answer is YES, and the path taken
we know that M^A asks at most i qu
then we do not necessarily know th
However, when we run $M^{(\ )}$ we can i
questions. In this manner note that
time.

We will use the notion of polynom

Definition. A set A is *p-close* i
$(A - B) \cup (B - A)$ is sparse.

oracle g that makes at most k queries. If $g = \chi_A$ then we denote this class $Q(k, A)$.

The class $Q(k, A)$ is related to truth-table reductions [21] in that if $B \in Q(k, A)$ then $B \leqslant_{(2^k-1)-tt} A$. The work in [21] differs from ours in that we deal with functions in general, and F_k^A in particular, while they deal with sets and with how the different types of reducibilities relate.

DEFINITION. If g is a function and $k \geqslant 1$ then $FQ(k, g)$ is the class of functions that can be computed by a polynomial oracle Turing machine with function oracle g, that makes at most k queries. If $g = \chi_A$ then we denote this class $FQ(k, A)$.

CONVENTION. Let k be any natural number. Note that if A and B are NP-complete then $Q(k, A)$ and $Q(k, B)$ are identical. We call this class of sets $Q(k, \text{NPC})$. Also note that in terms of queries to any NP-complete set, the functions F_k^A and F_k^B are equally hard to compute. We use the symbol F_k^{NPC} in the following way: Whatever is said of F_k^{NPC} will be true of F_k^A where A is any NP-complete set.

We are interested in finding out when the function F_k^A requires k queries to A. In recursion theory (no time bounds) this notion is called *terseness* [6]. In a polynomial framework we call it *pterseness*.

DEFINITION. Let $i \geqslant 2$. A set A is *i-pterse* if F_i^A cannot be computed in polynomial time by an M^A machine that makes fewer than i queries on every input. A set is *pterse* if it is i-pterse for all i.

All machines mentioned are assumed to work in polynomial time. An oracle machine where the oracle has not been specified is denoted $M^{(\,)}$. The phrase "run $M^{(\,)}(x)$ along every possible path" means to run M on x until a query arises, and then run the machine on both paths—the path taken if the answer is YES, and the path taken if the answer is NO. If A is a set and we know that M^A asks at most i questions (i an constant) on any input then we do not necessarily know that *every* path asks only i questions. However, when we run $M^{(\,)}$ we can ignore all paths that ask more than i questions. In this manner note that running all paths takes polynomial time.

We will use the notion of polynomial closeness [24, 30].

DEFINITION. A set A is *p-close* if there exists a set $B \in P$ such that $(A - B) \cup (B - A)$ is sparse.

3. Non-pterse Sets

In recursion theory one can show [6] that 2^m queries to A cannot be answered by m queries to A, for any nonrecursive A. Here we show that if $F^A_{2^m} \in FQ(m, A)$ then any number of queries to A can be answered by m queries to A. This does *not* lead to a contradiction, and in fact we show that there are arbitrarily complex sets A such that, for all k, $F^A_k \in FQ(1, A)$.

Theorem 1. *Let B be a set, and m be a natural number. If $F^B_{2^m} \in FQ(m, B)$, then for every natural number k, $F^B_k \in FQ(m, B)$.*

Proof. We prove this theorem by induction on k. If $k \leqslant 2^m$ then it is given. Assume that $F^B_k \in FQ(m, B)$ via machine M to show that $F^B_{k+1} \in FQ(m, B)$.

Let $\langle \alpha_1, ..., \alpha_{k+1} \rangle \in (\Sigma^*)^{k+1}$. We determine $F^B_{k+1}(\alpha_1, ..., \alpha_{k+1})$ with only m queries to B. Simulate $M(\alpha_1, ..., \alpha_k)$ *without* oracle calls, for all possibilities. Since M queries B at most m times, and after each reply follows one of two paths, there are no more than $2^m - 1$ queries that can be asked of B. This set of queries can be found in polynomial time.

Let the set of queries be $\{\beta_1, ..., \beta_{2^m-1}\}$. By the hypothesis of the theorem the value of $F^B_{2^m}(\beta_1, ..., \beta_{2^m-1}, \alpha_{k+1})$ can be deduced in m queries to B. The first $2^m - 1$ elements of this tuple can be used to simulate the $M(\alpha_1, ..., \alpha_k)$ calculations to yield $F^B_k(\alpha_1, ..., \alpha_k)$, and the last element is $\chi_B(\alpha_{k+1})$. This is all the information needed to compute $F^B_k(\alpha_1, ..., \alpha_{k+1})$.

We exhibit arbitrary complex sets A that are not 2-pterse. By the above theorem, for any k, $F^A_k \in FQ(1, A)$. The sets we use are super sparse; Ambos-Spies [1] used super sparse sets in a similar way in studying the relation between polynomial truth-table degrees and polynomial m-1 degrees.

Theorem 2. *For any time constructible function T' there exists a set $A \notin \mathrm{DTIME}(T'(n))$ that is not 2-pterse (i.e., $F^A_2 \in FQ(1, A)$).*

Proof. Let T be a time constructible function such that $T(x) \geqslant T'(x)$, and $T(x) > x$. Let g be the function

$$g(0) = 0$$

$$g(n+1) = T(g(n)).$$

Let A' be the set

$$A' = \{x \mid \exists_n x = 0^{g(n)}\}.$$

Membership in A' is decidable in polynomial time. By a modification of the time hierarchy theorem [14] there exists DTIME$(T(n)^2)$ but not in time DTIME function in DTIME$(T(n) \log T(n))$ that this is not important for our purposes.) see that $A \notin \mathrm{DTIME}(T(n))$. We show t both x and y are in A' (else membersh then the length of x is small compared determined easily since A is in DTIM following algorithm:

ALGORITHM (To determine F

(1) Input(x, y).

(2) If $x = y$ then query $x \in A$, output

(3) Determine if $x \in A'$ and if $y \in A'$.

(4) If $e = f = 0$ then output $(0, 0)$ and $x \in A$, output $(\chi_A(x), 0)$, and halt $(0, \chi_A(y))$, and halt.

(5) If both $x \in A'$ and $y \in A'$ then, $x < y$. (From step 2 we know th DTIME$(T(n)^2)$ algorithm for A $(\chi_A(x), \chi_A(y))$.

The algorithm obviously computes F operates in polynomial time. The only when we run the DTIME$(T^2(n))$ algorit there exists a k such $|y| = g(k) = T^{(k)}$ Therefore the part of the algorithm in

$$(T(|x|))^2 \leqslant (T(g(k-1)))^2 = (T($$

which is polynomial in the length of th

Note. If A is as constructed in t polynomial set then $A \cup B$, $A - B$, and obtain the answers to 2 queries for 1 (All these sets are p-close.

Convention. The phrase "the sets refer to all sets of the form $A \cup B$, A constructed in Theorem 2 and B is any

Corollary 3. *For any time constr $A \notin \mathrm{DTIME}(T(n))$ such that for all k, F*

time hierarchy theorem [14] there exists a function $f: A' \to \{0, 1\}$ that is in DTIME$(T(n)^2)$ but not in time DTIME$(T(n))$. (One can actually obtain a function in DTIME$(T(n) \log T(n))$ that is not in DTIME$(T(n))$, although this is not important for our purposes.) Let $A = \{x \mid f(x) = 1\}$. It is easy to see that $A \notin$ DTIME$(T(n))$. We show that A is not 2-pterse. Intuitively, if both x and y are in A' (else membership in A is trivially NO) and $x < y$ then the length of x is small compared to the length of y that $\chi_A(x)$ can be determined easily since A is in DTIME $(T(n)^2)$. Formally, we have the following algorithm:

ALGORITHM (To determine $F_2^A(x, y)$ with one query to A)

(1) Input(x, y).

(2) If $x = y$ then query $x \in A$, output $(\chi_A(x), \chi_A(x))$, and halt.

(3) Determine if $x \in A'$ and if $y \in A'$. Let $\chi_{A'}(x) = e$, $\chi_{A'}(y) = f$.

(4) If $e = f = 0$ then output $(0, 0)$ and halt. If $e = 1$ and $f = 0$ then query $x \in A$, output $(\chi_A(x), 0)$, and halt. If $e = 0$ then query $y \in A$, output $(0, \chi_A(y))$, and halt.

(5) If both $x \in A'$ and $y \in A'$ then, without loss of generality, assume $x < y$. (From step 2 we know that $x \neq y$.) Query $y \in A$. Using the DTIME$(T(n)^2)$ algorithm for A on x, determine if $x \in A$. Output $(\chi_A(x), \chi_A(y))$.

The algorithm obviously computes $F_2^A(x, y)$. We need only check that it operates in polynomial time. The only part for which this is not obvious is when we run the DTIME$(T^2(n))$ algorithm on x. If we get to step five then there exists a k such $|y| = g(k) = T^{(k)}(0)$ and $|x| \leqslant g(k-1) = T^{(k-1)}(0)$. Therefore the part of the algorithm in question takes time

$$(T(|x|))^2 \leqslant (T(g(k-1)))^2 = (T(T^{(k-1)}(0)))^2 = (T^{(k)})^2 = |y|^2,$$

which is polynomial in the length of the original input. ∎

Note. If A is as constructed in the above theorem, and B is any polynomial set then $A \cup B$, $A - B$, and $B - A$ ae also set for which you can obtain the answers to 2 queries for 1 (and by Theorem 1, k queries for 1). All these sets are p-close.

CONVENTION. The phrase "the sets A constructed in Theorem 2" will refer to all sets of the form $A \cup B$, $A - B$, and $B - A$ where A is a set constructed in Theorem 2 and B is any set in P.

COROLLARY 3. *For any time constructible function T there exists a set $A \notin$ DTIME$(T(n))$ such that for all k, $F_k^A \in FQ(1, A)$.*

Proof. If A is the A in Theorem 2 then by Theorem 1, for all k, $F_k^A \in FQ(1, A)$. ∎

4. 2-PTERSE AND k-PTERSE SETS

In Section 3 we showed that there are sets A that are arbitrarily complex, yet quite far from being pterse. Hence there is no obious connection between complexity and pterseness. The sets A are p-close, so perhaps there is a connection between p-closeness and non-pterseness. We conjecture the following:

Conjecture. If A is such that $F_k^A \in FQ(1, A)$ then A is p-close. As a first step towards this conjecture we show that every polynomial bi-immune set is 2-pterse. (Beigel [4] was the first person to show this, although we exhibit a simpler proof) Geske, Huynh, and Selman [12] have shown that there exist arbitrarily hard polynomial bi-immune sets (they actually prove a much stronger result), which when combined with our resuls shows that there exist arbitrarily hard sets which are 2-pterse.

DEFINITION. A set A is *polynomial immune* if it contains no infinite polynomial subset. A set A is *polynomial bi-immune* if both A and $\bar{A}$ are polynomial immune. If A is a tally set, a subset of 1^*, then by $\bar{A}$ we mean $1^* - A$.

We show that tally sets that are polynomial bi-immune are 2-pterse. From this it will easily follow that all polynomial bi-immune sets are 2-pterse.

THEOREM 4. *If A is a tally set that is polynomial bi-immune then A is 2-pterse.*

Proof. Let A be a tally set that is polynomial bi-immune, and assume $F_2^A \in FQ(1, A)$. Thus there is a polynomial oracle Turing machine $M^{()}$ that computes $F_2^A(x, y)$, using only one query to A. If we simulate $M^{()}$ then we can, at the A-query, follow both paths. Both paths finish in time polynomial in $|x|$ and $|y|$, and at least one path outputs a correct reply to F_2^A; the other path may output an incorrect answer.

There are three cases.

Case 1. There are an infinite number of k such that the two paths of $M^{()}(1^k, 1^{k-1})$ agree on the first component. We show that either A or $\bar{A}$ has a polynomial subset. Assume that for infinitely many k the two paths of $M^{()}(1^k, 1^{k-1})$ both output 1 in the first component. The following polynomial time algorithm recognizes an infinite subset of A: On input 1^k

run both paths of $M^{()}(1^k, 1^{k-1})$; if component then output 1, else output both paths of $M^{()}(1^k, 1^{k-1})$ output 0 t of these two subcases must occur.

Case 2. There are an infinite numb $M^{()}(1^k, 1^{k-1})$ agree on the second polynomial subset of either A or $\bar{A}$ by

Case 3. There are only a finite nu agree on either the first or the second

$$X = \{1^k \mid \text{the two paths of } M^{()}(1^k, \text{component}\} \cup \{1^k \mid \text{the two paths of } M^{()}(1^k, \text{component}\} \cup \{1\}.$$

By assumption X is a finite set. We hardwired into it $\chi_A(x)$ for every $x \in X$

ALGORITH

(1) Input(1^k)

(2) If $1^k \in X$ then determine if $1^k \in X$ answer that is stored, and halt.

(3) Run both paths of $M^{()}(1^k, 1^{k-1})$ both slots. Hence we can assum $1-b, 0)$ for some $b \in \{0, 1\}$. W $\chi_A(1^{k-1}) = 1$.

(4) Recursively call the algorithm on with the information obtaine in $\chi_A(1^k)$.

It is easy to see that this algorithm ru

In the proof we never use the fact tha Hence we actually show

THEOREM 5. *If A is a polynomial b then $F_2^A \notin FQ(1, B)$.*

COROLLARY 6. *If A is polynomi Moreover, if B is any set, then $F_2^A \notin FQ$*

Proof. If A is not 2-pterse then F_2^A

run both paths of $M^{()}(1^k, 1^{k-1})$; if they both output 1 on the first component then output 1, else output 0. A similar argument yields that if both paths of $M^{()}(1^k, 1^{k-1})$ output 0 then $\bar{A}$ has a polynomial subset. One of these two subcases must occur.

Case 2. There are an infinite number of k such that the two paths of $M^{()}(1^k, 1^{k-1})$ agree on the second component. This yields an infinite polynomial subset of either A or $\bar{A}$ by reasoning similar to that in Case 1.

Case 3. There are only a finite number of k such that $M^{()}(1^k, 1^{k-1})$ agree on either the first or the second component. We show $A \in P$. Let

$$X = \{1^k \mid \text{the two paths of } M^{()}(1^k, 1^{k-1}) \text{ agree on the first component}\}$$
$$\cup \{1^k \mid \text{the two paths of } M^{()}(1^k, 1^{k-1}) \text{ agree on the second component}\} \cup \{1\}.$$

By assumption X is a finite set. We can assume that the algorithm has hardwired into it $\chi_A(x)$ for every $x \in X$.

ALGORITHM for A

(1) Input(1^k)

(2) If $1^k \in X$ then determine if $1^k \in X$ by table lookup. Output the correct answer that is stored, and halt.

(3) Run both paths of $M^{()}(1^k, 1^{k-1})$. Since $1^k \notin X$ the answers differ in both slots. Hence we can assume that the answers are $(b, 1)$ and $1-b, 0)$ for some $b \in \{0, 1\}$. We now know that $\chi_A(1^k) = b$ iff $\chi_A(1^{k-1}) = 1$.

(4) Recursively call the algorithm on input(1^{k-1}). This answer, together with the information obtaine in step 3, yields the correct value for $\chi_A(1^k)$.

It is easy to see that this algorithm runs in polynomial time. ▮

In the proof we never use the fact that the oracle being queried is A itself. Hence we actually show

THEOREM 5. *If A is a polynomial bi-immune tally set and B is any set then $F_2^A \notin FQ(1, B)$.*

COROLLARY 6. *If A is polynomial bi-immune then A is 2-pterse. Moreover, if B is any set, then $F_2^A \notin FQ(1, B)$.*

Proof. If A is not 2-pterse then $F_2^A \in FQ(1, A)$; hence both $F_2^{A \cap 1^*}$ and

$F_2^{\bar{A}\cap 1^*}$ are in $FQ(1, A)$. At least one of $A \cap 1^*$ and $\bar{A} \cap 1^*$ is polynomial bi-immune (within 1^*), which contradicts the above corollary. ▮

Beigel has shown that there are polynomial bi-immune sets A for which $F_4^A \in FQ(2, A)$, and hence (by Theorem 1) that $F_{2^n}^A \in FQ(n, A)$ for any $n \geq 2$. Therefore we cannot replace 2-pterse with any other pterseness-type condition.

We use the above corollary to exhibit a variety of different types of 2-pterse sets. We need the following proposition of Geske, Huynh, and Selman [12].

PROPOSITION 7. *If T_1 and T_2 are any time constructible functions such that $\lim_{n\to\infty} T_1(n) \log T_1(n)/T_2(n) = 0$ then there exists a language that is in* DTIME($T_2(n)$) *that is* DTIME($T_1(n)$)*-bi-immune.*

This proposition yields polynomial bi-immune sets of subexponential, and arbitrarily high complexities. Hence, combining the proposition with Corollary 6 yields the following three corollaries.

COROLLARY 8. *There are 2-pterse sets that are subexponential.*

COROLLARY 9. *There exists a 2-pterse set $A \in$ EXPTIME $- P$.*

COROLLARY 10. *There exist arbitrarily complex* 2-pterse sets.

In a recursion-theoretic setting all 1-generic sets (see [16]) are terse [6]. Ambos-Spies, Fleischhack, and Huwig have defined a notion of p-generic [2] set that is similar. The definition is somewhat complicated so we omit it, but one easy consequence of the definition is that p-generic sets are polynomially bi-immune. Thus we have the following corollary.

COROLLARY 11. *If A is p-generic then A is 2-pterse.*

In [3] we prove, using methods of p-genericity, that strongly p-generic sets are k-pterse for all k.

The sets A in Theorem 2 are constructed by diagonalization, which raises the possibility of a connection between "naturalness" and pterseness. We prove a theorem along these lines by showing that (if P $\neq$ NP) every NP-complete set is 2-pterse.

THEOREM 12. *If P $\neq$ NP then for all $k > 0$, $F_{2^k}^{\text{NPC}} \notin FQ(k,$ NPC$)$. (In particular P $\neq$ NP iff every* NP*-complete set is 2-pterse.)*

Proof. Assume that A is an NP-complete set for which $F_{2^k}^A \in FQ(k, A)$. By Theorem 1 for all m, $F_m^A \in FQ(k, A)$. If $f \in FQ(k+1, A)$ then by simulating the computation of $f(x)$ th
that may arise we see that $f \in FQ(1, F_{2^k}^A$
we have $FQ(k+1, A)$. Since A is NP
and $FQ(k+1, A) = FQ(k+1,$ SAT$)$; t
Krentel [19] showed that if P $\neq$ N
$FQ(k+1,$ SAT$)$. Hence if P $\neq$ NP then
all k, $F_{2^k}^A \notin FQ(k,$ NPC$)$. ▮

There is not much known about k-
exist and can show this three different

THEOREM 13. *Every nonrecursive T*
that for all k, A is k-pterse.

Proof. In [6] it is shown that
contains a superterse set. This set will

The last theorem exhibited nonrecurs
recursive ones exist.

THEOREM 14. *For all k, there exists*

Proof. Following [21] we define B
(in polynomial time) that maps strin
variables, where each variable is of t
evaluates to true. They show how
DTIME(2^n)) set A such that $\{B \mid B \leq_{(k}$
not k-pterse then $F_k^A \leq_{(k-1)-tt} A$ an
reducible to A is $\leq_{(k-1)-tt}$ reducible t

The next theorem is proven in [3].

THEOREM 15. *If A is strongly p-gene*

5. POLYNOMIAL S

If a set is not 2-pterse then it is, in
Even if the set is (as in Theorem 2)
complexity, there is something about it
sets. Polynomially selective sets have
explore similarities between these two
from [25] though it resembles Jockusc

A set A is *polynomial selective* (hen
polynomial time computable function
$A \cap \{x, y\} \neq \varnothing$ then $f(x, y) \in A$.

simulating the computation of $f(x)$ through all $2^{k+1}-1$ possible queries that may arise we see that $f \in FQ(1, F^A_{2^{k+1}-1})$. Since $F^A_{2^{k+1}-1} \in FQ(k, A)$, so we have $FQ(k+1, A)$. Since A is NP-complete, $FQ(k, A) = FQ(k, \text{SAT})$ and $FQ(k+1, A) = FQ(k+1, \text{SAT})$; thus $FQ(k, \text{SAT}) = FQ(k+1, \text{SAT})$. Krentel [19] showed that if $\text{P} \neq \text{NP}$ then for all k, $FQ(k, \text{SAT}) \neq FQ(k+1, \text{SAT})$. Hence if $\text{P} \neq \text{NP}$ then for all NP-complete sets A and for all k, $F^A_{2^k} \notin FQ(k, \text{NPC})$. ∎

There is not much known about k-pterse sets. We do know that they exist and can show this three different ways.

THEOREM 13. *Every nonrecursive Turing degree contains a set A such that for all k, A is k-pterse.*

Proof. In [6] it is shown that every nonrecursive Turing degree contains a superterse set. This set will also be k-pterse for every k.

The last theorem exhibited nonrecursive k-pterse sets. We now show that recursive ones exist.

THEOREM 14. *For all k, there exists a recursive set A that is k-pterse.*

Proof. Following [21] we define $B \leqslant_{k-tt} A$ if there exists a function f (in polynomial time) that maps strings to propositional formulas of k variables, where each variable is of the form $x_i \in B$, and $x \in A$ iff $f(x)$ evaluates to true. They show how to construct a recursive (in fact DTIME(2^n)) set A such that $\{B \mid B \leqslant_{(k-1)-tt} A\} \neq \{B \mid B \leqslant_{k-tt} A\}$. If A is not k-pterse then $F^A_k \leqslant_{(k-1)-tt} A$ and thus every set which is $\leqslant_{k-tt}$ reducible to A is $\leqslant_{(k-1)-tt}$ reducible to A. That is a contradiction. ∎

The next theorem is proven in [3].

THEOREM 15. *If A is strongly p-generic then for all k, A is k-pterse.*

5. POLYNOMIAL SELECTIVE SETS

If a set is not 2-pterse then it is, in some sense, computationally easy. Even if the set is (as in Theorem 2) arbitrarily hard in terms of time complexity, there is something about it that makes it easier than other such sets. Polynomially selective sets have the same kind of property. We explore similarities between these two notions. The following definition is from [25] though it resembles Jockusch's semicursive sets.

A set A is *polynomial selective* (henceforth p-selective) if there exists a polynomial time computable function f such that $f(x, y) \in \{x, y\}$ and if $A \cap \{x, y\} \neq \varnothing$ then $f(x, y) \in A$.

The properties of being non-2-pterse and being p-selective seem similar. They both make the set easier computationally, in an indirect way. The sets constructed in Theorem 2 are p-selective, thus yielding the following theorem.

THEOREM 16. *For any time constructible function T there exists a set $A \notin \mathrm{DTIME}(T(n))$ such that A is* p-*selective.*

Proof. Let A and A' be as in Theorem 2. The set A is in $\mathrm{DTIME}(T(n)^2)$ but not in $\mathrm{DTIME}(T(n))$. We show that it is p-selective.

ALGORITHM

(1) Input(x, y).
(2) If $x = y$ then output x and halt.
(3) If $x \notin A'$ then output y and halt.
(4) If $y \notin A'$ then output x and halt.
(5) (both $x, y \in A'$ and $x \neq y$) Assume $x < y$. For the $\mathrm{DTIME}(T(n)^2)$ algorithm for A on x, determine if $x \in A$; output x if it is, y otherwise.

For reasons analogous to those in the proof of Theorem 2, the algorithm runs in polynomial time.

These selective sets appear contrived and (as in the case of non-2-pterse sets) make us ponder if such sets are unnatural. Grollman and Selman [13] hae shown a theorem analogous to our Theorem 12 along these lines.

THEOREM 17. *If* $\mathrm{P} \neq \mathrm{NP}$ *then* SAT (*and any other* NP-*complete set*) *is not* p-*selective.*

The two above theorems indicate that p-selective sets and non-2-pterse sets have a similar flavor. It would be of interest to push the analogy further and prove (say) that every polynomial bi-immune set is not p-selective. As of now this is an open question. We have shown, in [3], that p-selectiveness and non-2-pterseness are not equivalent.

6. kSAT AND BOUNDED QUERIES TO SAT

kSAT is the set of all Boolean formulas that have exactly k satisfying assignments. This set is not known to be in NP, although it is easily seen to be in $\mathrm{P}^{\mathrm{SAT}}$. We use bounded query classes to clarify the complexity of kSAT. We show that it is "closer" to NP then to $\mathrm{P}^{\mathrm{SAT}}$ in terms of the number of questions needed. In particular, we show that $k\mathrm{SAT} \in Q(1, F_2^{\mathrm{SAT}})$ and $\mathrm{SAT} \in Q(1, k\mathrm{SAT})$. As a corollary we obtain that kSAT is co-NP-hard.

Valiant and Vazirani [27] have studi
exactly one satisfying assignment) and
D_2 (differences of NP sets) using r
Gurevich [8] show that there are ora
(properly defined) is complete for D_2^A, a
(using deterministic reductions). They
USAT there) is co-NP-hard. Our pro
theirs.

CONVENTION. If α is a Boolean for
TRUE iff $\alpha \in B$, and NOT $B(\alpha)$ means

The following two theorems were obt

THEOREM 18. $k\mathrm{SAT} \in Q(1, F_2^{\mathrm{SAT}})$.

Proof. For a any constant, let B_a be
or more satisfying assignments. Since
formula in B_a can be determined by on
The following algorithm solves kSAT

ALGORI

(1) Input(α) (a Boolean formula).
(2) Using F_2^{SAT} find out $B_k(\alpha)$ and B_k
(3) If $B_k(\alpha)$ is TRUE and B_{k+1} is FA
NO. ∎

Note. Alternatively, one can obtain
theorem (due to [18] and also to [4]):
is in D_2, it is in $Q(1, F_2^{\mathrm{SAT}})$.

The next two results appear, for the
modifications of theirs.

THEOREM 19. $\mathrm{SAT} \in Q(1, k\mathrm{SAT})$.

Proof. The following algorithm so
kSAT.

ALGORI

(1) Input$(\alpha(x_1, \ldots, x_n))$ (a Boolean for
(2) If $2^n \leqslant k$ then try all 2^n possib
appropriate value, and halt.

Valiant and Vazirani [27] have studied USAT (the set of formulas with exactly one satisfying assignment) and have shown that it is complete for D_2 (differences of NP sets) using *randomized* reductions. Blass and Gurevich [8] show that there are oracles A and B such that USAT^A (properly defined) is complete for D_2^A, and USAT^B is not complete for D_2^B (using deterministic reductions). They also obtain that 1-SAT (called USAT there) is co-NP-hard. Our proof is essentially a modification of theirs.

CONVENTION. If α is a Boolean formula and B is a set then $B(\alpha)$ is TRUE iff $\alpha \in B$, and NOT $B(\alpha)$ means that $\alpha \notin B$.

The following two theorems were obtained in collaboration with Beigel.

THEOREM 18. $k\text{SAT} \in Q(1, F_2^{\text{SAT}})$.

Proof. For a any constant, let B_a be the set of all formulas that have a or more satisfying assignments. Since B_a is in NP the membership of a formula in B_a can be determined by one query to SAT.

The following algorithm solves $k\text{SAT}$ and makes one call to F_2^{SAT}.

ALGORITHM

(1) Input(α) (a Boolean formula).

(2) Using F_2^{SAT} find out $B_k(\alpha)$ and $B_{k+1}(\alpha)$.

(3) If $B_k(\alpha)$ is TRUE and B_{k+1} is FALSE then output YES, else output NO. ∎

Note. Alternatively, one can obtain this result by using the following theorem (due to [18] and also to [4]): $D_2 \cup \bar{D}_2 \subseteq Q(1, F_2^{\text{SAT}})$. Since $k\text{SAT}$ is in D_2, it is in $Q(1, F_2^{\text{SAT}})$.

The next two results appear, for the $k = 1$ case, in [8]. Our proofs are modifications of theirs.

THEOREM 19. $\text{SAT} \in Q(1, k\text{SAT})$.

Proof. The following algorithm solves SAT and makes one call to $k\text{SAT}$.

ALGORITHM

(1) Input($\alpha(x_1, ..., x_n)$) (a Boolean formula)

(2) If $2^n \leqslant k$ then try all 2^n possible truth assignments, output the appropriate value, and halt.

(3) Construct a formula $\beta(x_1, ..., x_n)$ that has exactly k satisfying assignments (this is possible since $k \leqslant 2^n$). If one of those assignments satisfies α then output(YES) and halt.

(4) If NOT kSAT($\alpha \vee \beta$) then output YES, else output NO.

The above algorithm works because:

(a) If α is satisfiable then either (1) α is satisfied by one of the k truth assignments that satisfy β in which case step 3 of the algorithm outputs YES, or (2) there is an assignment that satisfies α but not β, hence ($\alpha \vee \beta$) has at least $k+1$ satisfying assignments so NOT kSAT($\alpha \vee \beta$) is true, and step 4 outputs YES.

(b) If α is not satisfiable then $\alpha \vee \beta$ has exactly k satisfying assignment, NOT kSAT($a \vee \beta$) is FALSE, so step 4 outputs NO. ∎

COROLLARY 20. kSAT *is* co-NP-*hard.*

Proof. If the above reduction is modified to output some $\gamma \notin k$SAT instead of YES in step 3 and $\alpha \vee \beta$ in step 4 then we have a polynomial m-reduction of the complement of SAT to kSAT. ∎

7. Relation to the Difference Hierarchy of NP Sets

We compare bounded query classes to other classes that are "in the vicinity of P^{SAT}." In particular, we show that the kth level of the Boolean hierarchy (defined by Cai and Hemachandra [10]) is contained in $Q(\lceil \log(k+1) \rceil, \text{NPC})$. The proof is essentially a binary search. We then prove an analogous theorem in the setting where one can ask p questions simultaneously, which uses a technique similar to $(p+1)$-ary search, i.e., binary search with p processors [20, 26].

Our definition of the Boolean hierarchy looks different from the original one, but they are equivalent. See [9] for a proof of that.

DEFINITION. The *Boolean hierarchy* is the following sequence of classes of sets:

$$D_1 = \{X \mid \in \text{NP}\}$$
$$D_{n+1} = \{X \mid X = Y - Z,\ Y \in \text{NP},\ Z \in D_n\}.$$

If A is in D_k then there exist NP sets $L_1, ..., L_n$ such that

$$A = L_1 - (L_2 - (L_3 - \cdots - (L_{n-2} - (L_{n-1} - L_n)) \cdots)).$$

Note. The Boolean hierarchy has m
it is shown that the statement $A \in BH$ is

(1) A is a finite union of D_2 sets,

(2) A is a finite intersection of co

(3) A is in the Boolean closure o

(4) $A \leqslant_{k-tt}$ SAT for some k.

(5) $L_1 - (L_2 - (L_3 - \cdots - (L_{n-2}$
L_i.

Note. To determine if an element x
such that $x \notin L_i$, and then note that x

CONVENTION. $\bar{D}_k$ is the class of all s
$\bar{D}_k = \{A \mid \bar{A} \in D_k\}$.

Papadimitriou and Yannakakis [2
[22], have shown that many proble
complete. Cai and Hemachandra [10
hierarchy. They exhibit problems relat
for each level of the hierarchy. They als
levels have sparse sets. Wagner and
similar notions.

In a recursion-theoretic setting ther
$Q(i, K)$ classes (K is the halting set) an
[7]. A similar interleaving holds in a
section, together with results of Beigel,

$$D_1 \subseteq Q(1, K) \subseteq D_2 \subseteq D_3 \subseteq Q(2, K) \subseteq$$
$$\subseteq Q(3, K) \subseteq D_8 \subseteq \cdots \subseteq D_{2^k-1} \subseteq$$

We are unable to obtain any proper
on P ≠ NP may be possible.

THEOREM 21. $D_k \cup \bar{D}_k \subseteq Q(\lceil \log(k -$

Proof. Let k and m be such that 2
and let $\langle L_1, ..., L_k \rangle$ be NP sets such t

$$A = L_1 - (L_2 - (L_3 - \cdots (L$$

By convention let $L_{k'} = \varnothing$ for $k+1 \leqslant$
algorithm that, given x, determines th
the question of membership of x in A.

Note. The Boolean hierarchy has many equivalent formulations. In [9] it is shown that the statement $A \in BH$ is equivalent to each of the following:

(1) A is a finite union of D_2 sets,

(2) A is a finite intersection of co-D_2-sets,

(3) A is in the Boolean closure of NP,

(4) $A \leqslant_{k-tt} \mathrm{SAT}$ for some k.

(5) $L_1 - (L_2 - (L_3 - \cdots - (L_{n-2} - (L_{n-1} - L_n)) \cdots))$, where $L_{i+1} \subseteq L_i$.

Note. To determine if an element x is in A itsuffices to find the least i such that $x \notin L_i$, and then note that $x \in A$ iff i is even.

CONVENTION. $\bar{D}_k$ is the class of all sets whose complements are D_k, i.e., $\bar{D}_k = \{A \mid \bar{A} \in D_k\}$.

Papadimitriou and Yannakakis [23], and Papadimitriou and Wolfe [22], have shown that many problems, including TSP facets, are D_2-complete. Cai and Hemachandra [10] (originally) defined the Boolean hierarchy. They exhibit problems related to colorability that are complete for each level of the hierarchy. They also discussed oracle results and which levels have sparse sets. Wagner and Wechsung [28, 29] have defined similar notions.

In a recursion-theoretic setting there is a nice interleaving between the $Q(i, K)$ classes (K is the halting set) and the difference hierarchy of r.e. sets [7]. A similar interleaving holds in a complexity theory framework. This section, together with results of Beigel, will establish the following diagram:

$$D_1 \subseteq Q(1, K) \subseteq D_2 \subseteq D_3 \subseteq Q(2, K) \subseteq D_4 \subseteq D_5 \subseteq D_6 \subseteq D_7$$
$$\subseteq Q(3, K) \subseteq D_8 \subseteq \cdots \subseteq D_{2^k-1} \subseteq Q(k, K) \subseteq D_{2^k} \subseteq D_{2^k+1} \subseteq \cdots.$$

We are unable to obtain any proper containments. A theorem contingent on $\mathrm{P} \neq \mathrm{NP}$ may be possible.

THEOREM 21. $D_k \cup \bar{D}_k \subseteq Q(\lceil \log(k+1) \rceil, \mathrm{NPC})$.

Proof. Let k and m be such that $2^{m-1} < k+1 \leqslant 2^m$. Let A be a D_k set, and let $\langle L_1, \ldots, L_k \rangle$ be NP sets such that

$$A = L_1 - (L_2 - (L_3 - \cdots (L_{k-2} - (L_{k-1} - L_k)) \cdots)).$$

By convention let $L_{k'} = \varnothing$ for $k+1 \leqslant k' \leqslant 2^m$. We present a $Q(m, \mathrm{NPC})$ algorithm that, given x, determines the least i such that $x \notin L_i$, and hence the question of membership of x in A. The algorithm is essentially a binary

search: each question to NPC cuts in half the interval where i can be found. This is why there is a logarithm.

ALGORITHM

(1) Input(x).

(2) $a \leftarrow 0, b \leftarrow 2^m$ (i will always be in $[a+1, b]$).

(3) Ask NPC

$$x \in \bigcap_{j=a+1}^{a+(b-a)/2} L_j?$$

If YES then $a \leftarrow a+(b-a)/2$ else $b \leftarrow a+(b-a)/2$. (We will see later that $b-a$ is always even.)

(4) If $a+(b-a)/2 > a+1$ (i.e., if $b-a>2$) then go to step 3. If $a+(b-a)/2=2$ (i.e., if $b-a=2$) then ask NPC, $x \in L_{a+1}$? If YES then $i \leftarrow a+2$ else $i \leftarrow a+1$. (We will see later that $b-a$ will eventually be 2.)

(5) If i is even then output(YES), else output(NO).

By induction one can show that after step 3 is executed q times, $b-a=2^{m-q}$, and we know that the least i such that $x \notin L_i$ is in the interval $[a+1, b]$. After m steps we have $b-a=2$ and the algorithm terminates with the correct answer. The number of queries is m which is $\leqslant \lceil \log(k+1) \rceil$. Hence $D_k \subseteq Q(k, \text{NPC})$. Since $Q(k, \text{NPC})$ is closed under complementation, $\bar{D}_k \subseteq Q(k, \text{NPC})$, so $D_k \cup \bar{D}_k \subseteq Q(k, \text{NPC})$. ▮

THEOREM 22. *If A is in $Q(k, \text{NPC})$ then there exist D_2 sets $B_1, \ldots, B_{2^k}$ such that $A = \bigcup_{i=1}^{2^k} B_i$. This implies that $Q(k, \text{SAT}) \in D_{2^{k+1}}$.*

Proof. Let $M^{(\,)}$ be a polynomial oracle Turing machine that recognizes A and makes at most k queries to SAT. By convention, $M^{(\,)}$ makes at most k queries on any path that it takes. Let $w_1, \ldots, w_{2^k}$ be all 2^k elements of $\{0,1\}^k$. We will define, for every i ($1 \leqslant i \leqslant 2^k$), sets C_{i1} and C_{i2} such that C_{i1} is NP, C_{i2} is co-NP; and if $x \in A$ and the correct computation that accepts x follows the query answers proided by w_i, then x is in both C_{i1} and C_{i2}.

For each x one can run $M^{(\,)}$ using the jth bit of w_i to answer the jth query. In this manner we can easily determine what queries would be asked and what the final answer would be if w_i were used for the answers.

Let

$C_{i1} = \{x \mid M^{(\,)}(x)$ using w_i accepts
$\forall_j\, w_i[j]=1 \Rightarrow$ the jth query

$C_{i2} = \{x \mid M^{(\,)}(x)$ uses w_i then
$\forall_j\, w_i[j]=0 \Rightarrow$ the jth query

Checking membership in C_{i1} is
formulas are in SAT, hence, C_{i1} is in
essentially seeing if k (a constant) form
in co-NP.

If $x \in C_{i1} \cap C_{i2}$ then path w_i is correc
the computation $M^{\text{SAT}}(x)$ followed so

$$A = \bigcup_{i=1}^{2^k} C_{i1} \cap C_{i2} = \bigcup_{i=}^{2^k}$$

where $B_i = C_{i1} - \bar{C}_{i2}$, a D_2 set.

In [9] it is shown that $A \in D_{2k}$ iff
sets in D_2. Combining this with
$Q(k, \text{SAT}) \subseteq D_{2^{k+1}}$.

For the sake of completeness we in
mentioned in the last paragraph. If A

$$A = L_1 - (L_2 - (L_3 - \cdots - (L$$

where for each i, L_i is an NP set and L
as $A = (L_1 - L_2) \cup (L_3 - L_4) \cup \cdots \cup ($
desired form.

For the reverse direction we give
idea. Let $A = (L_1 - L_2) \cup (L_3 - L_4)$, wh
as a set in D_4. Let

$$A_1 = L_1 \cup L_2$$
$$A_2 = L_2 \cup L_4$$
$$A_3 = (L_1 \cap L_4) \cup (L$$
$$A_4 = (L_1 \cap L_3 \cap L_4)$$

A can be expressed as

$$A = (A_1 - (A_2 -$$

Note. Beigel [4] has improved
$Q(k, \text{NPC}) \subseteq Q(1, D_{2^k-1}) \subseteq D_{2^k}$.

Let

$C_{i1} = \{x \mid M^{( )}(x)$ using w_i accepts and
$\forall_j\, w_i[j] = 1 \Rightarrow$ the jth query encountered has answer YES$\}$

$C_{i2} = \{x \mid M^{( )}(x)$ uses w_i then
$\forall_j\, w_i[j] = 0 \Rightarrow$ the jth query encountered has answer NO$\}$.

Checking membership in C_{i1} is essentially seeing if k (a constant) formulas are in SAT, hence, C_{i1} is in NP. Checking membership in C_{i2} is essentially seeing if k (a constant) formulas are NOT in SAT; hence, C_{i2} is in co-NP.

If $x \in C_{i1} \cap C_{i2}$ then path w_i is correct and accepts, so $x \in A$; if $x \in A$ then the computation $M^{\mathrm{SAT}}(x)$ followed some path w_i, so $x \in C_{i1} \cap C_{i2}$. Hence

$$A = \bigcup_{i=1}^{2^k} C_{i1} \cap C_{i2} = \bigcup_{i=1}^{2^k} C_{i1} - \bar{C}_{i2} = \bigcup_{i=1}^{2^k} B_i,$$

where $B_i = C_{i1} - \bar{C}_{i2}$, a D_2 set.

In [9] it is shown that $A \in D_{2k}$ iff A can be written as the union of k sets in D_2. Combining this with what we have shown we obtain $Q(k, \mathrm{SAT}) \subseteq D_{2^k+1}$.

For the sake of completeness we include a sketch of the result of [9] mentioned in the last paragraph. If $A \in D_{2k}$ then A can be written as

$$A = L_1 - (L_2 - (L_3 - \cdots - (L_{2k-2} - (L_{2k-1} - L_{2k})) \cdots)),$$

where for each i, L_i is an NP set and $L_{i+1} \subseteq L_1$. The set A can be rewritten as $A = (L_1 - L_2) \cup (L_3 - L_4) \cup \cdots \cup (L_{2k-1} - L_{2k})$. This is clearly in the desired form.

For the reverse direction we give an example that expresses the main idea. Let $A = (L_1 - L_2) \cup (L_3 - L_4)$, where each L_i is in NP. We express A as a set in D_4. Let

$$A_1 = L_1 \cup L_2$$
$$A_2 = L_2 \cup L_4$$
$$A_3 = (L_1 \cap L_4) \cup (L_2 \cap L_3)$$
$$A_4 = (L_1 \cap L_3 \cap L_4) \cup (L_1 \cup L_2 \cup L_3).$$

A can be expressed as

$$A = (A_1 - (A_2 - (A_3 - A_4))). \quad \blacksquare$$

Note. Beigel [4] has improved the above result by showing that $Q(k, \mathrm{NPC}) \subseteq Q(1, D_{2^k-1}) \subseteq D_{2^k}$.

We now look at what happens if we can ask p questions at a time. In particular, we see how D_k relates to $Q(-, F_p^A)$ where p is a fixed constant. We use a technique similar to $(p+1)$-ary search, i.e., binary search with p processors [20, 26], in a manner similar to our use of binary search in Theorem 20. The constant p is arbitrary but fixed.

We show that the classes $Q(k, F_p^{\mathrm{NPC}})$ bear a relationship to the difference hierarchy.

THEOREM 23. $D_k \cup \bar{D}_k \subseteq Q(\lceil \log(k+1)/\log(p+1) \rceil, F_p^{\mathrm{NPC}})$.

Proof. Let k and m be such that $(p+1)^{m-1} < k+1 \leqslant (p+1)^m$. Let A be a D_k set, and let $\langle L_1, ..., L_k \rangle$ be NP sets such that

$$A = L_1 - (L_2 - (L_3 - \cdots - (L_{k-2} - (L_{k-1} - L_k)) \cdots)).$$

By convention let $L_{n'} = \varnothing$ for $k+1 \leqslant k' \leqslant (p+1)^m$. We present a $Q(m, F_p^{\mathrm{NPC}})$ algorithm which, given x, determines the least i such that $x \notin L_i$, and hence the question of membership of x in A. The algorithm is essentially a $(p+1)$-ary search; each question to F_p^{NPC} reduces the length of the interval in which i can be found by a factor of $p+1$.

ALGORITHM

(1) Input(x).

(2) $a \leftarrow 0$, $b \leftarrow (p+1)^m$ (i will always be in $[a+1, b]$).

(3) Ask F_p^{NPC} simultaneously

$$x \in \bigcap_{j=a+1}^{a+(b-a)/(p+1)} L_j?$$

$$x \in \bigcap_{j=a+1}^{a+2(b-a)/(p+1)} L_j?$$

$$x \in \bigcap_{j=a+1}^{a+3(b-a)/(p+1)} L_j?$$

$$\vdots$$

$$x \in \bigcap_{j=a+1}^{a+p(b-a)/(p+1)} L_j?$$

(We will see later that $b-a$ is always divisible by $p+1$.)

(4) Let k_0 be the largest k such that

$$x \in \bigcap_{j=a+1}^{a+k_0(b-a)/(p+1)} L_j.$$

Set

$a \leftarrow a +$

$b \leftarrow a +$

(Note: the a and b on the right and b.)

(5) If $a+1 < a+(b-a)/(p+1)$ (i.e., $a+1 = a+(b-a)/(p+1)$ (i.e., simultaneously $x \in L_{a+1}$, $x \in L_{a+}$ number in $\{a+1, a+2, ..., a+p$ that $b-a$ will eventually be $p+$ happen.)

(6) If i is even then output(YES), els

By induction one can show that $b-a = (p+1)^{m-q}$, and we know that interval $[a+1, b]$

The rest of this proof is analogous t

THEOREM 24. *If A is in $Q(k, F_p^{\mathrm{NPC}}$ such that $A = \bigcup_{i=1}^{2^{pk}} B_i$. This implies tha*

Proof. This is proven in a manne except that the w_i are sequences of elem formulas on each path to be concerne

8. NONSPARSE VERBOSE SE

The sets A constructed (in Theore p-close. This leads us to believe that question now arises, Are there some *some* savings, though perhaps not quite a savings on queries is possible for s p-close.

The following theorem is based o presented in [15], put into a polynom used in recursion-theoretic terseness b [6]. We construct verbose sets that cannot prove this.

Set

$$a \leftarrow a + \frac{k_0(b-a)}{p+1}$$

$$b \leftarrow a + \frac{(k_0+1)(b-a)}{p+1}.$$

(Note: the a and b on the right hand side are the old values of a and b.)

(5) If $a+1 < a+(b-a)/(p+1)$ (i.e., $b-a > p+1$) then go to step 3. If $a+1 = a+(b-a)/(p+1)$ (i.e., $b-a = p+1$) then ask F_p^{NPC} simultaneously $x \in L_{a+1}$, $x \in L_{a+2}$, ..., $x \in L_{a+p}$. Set i to be the least number in $\{a+1, a+2, ..., a+p\}$ such that $x \notin L_i$. (We will see later that $b-a$ will eventually be $p+1$; hence no other possibility can happen.)

(6) If i is even then output(YES), else output(NO).

By induction one can show that after step 3 is executed q times, $b-a = (p+1)^{m-q}$, and we know that the least i such that $x \notin L_i$ is in the interval $[a+1, b]$

The rest of this proof is analogous to that for Theorem 20. ∎

THEOREM 24. *If A is in $Q(k, F_p^{\mathrm{NPC}})$ then there exist D_2 sets B_1, ..., $B_{2^{pk}}$ such that $A = \bigcup_{i=1}^{2^{pk}} B_i$. This implies that $Q(k, F_p^{\mathrm{NPC}}) \subseteq D_{2^{p+1}}$.*

Proof. This is proven in a manner similar to that for Theorem 21, except that the w_i are sequences of elements from $\{0, 1\}^p$, and there are pk formulas on each path to be concerned with. ∎

8. Nonsparse Verbose Sets and Superterse Sets

The sets A constructed (in Theorem 2) such that $F_k^A \in FQ(1, A)$ are p-close. This leads us to believe that all such sets must be p-close. The question now arises, Are there some non-p-close sets where you can get *some* savings, though perhaps not quite as drastic as k for 1? We show that a savings on queries is possible for some sets that do not appear to be p-close.

The following theorem is based on ideas of McLauglin and Martin presented in [15], put into a polynomial framework by Selman [25], and used in recursion-theoretic terseness by Beigel, Gasarch, Gill, and Owings [6]. We construct verbose sets that do not seem to be p-close but we cannot prove this.

THEOREM 25. *If A is a tally set then there exists a nonsparse set B, $B \equiv_{tt}^{p} A$, such that for all n, $F_{2^n-1}^{B} \in FQ(n, B)$.*

Proof. Let $a_i = \chi_A(1^i)$. Let **a** be the infinite string $a_1 a_2 a_3 \ldots$. Let

$$B = \{w \mid w \text{ is lexicographically less than } \mathbf{a}\}.$$

It is easy to see that $A \equiv_{tt}^{p} B$ and that B is not sparse. $F_{2^n-1}^{B}(x_1, \ldots, x_{2^n-1})$ can be found by a binary search which determines the largest element that is NOT in B; all larger elements are in B, all smaller ones are not. This only needs n queries. ∎

Note that the set B can be looked at as the left cut of a real [17], where that real is in the same polynomial 1–1 degree as A. The set B in the above theorem does not appear to be p-close, but we are currently unable to prove this.

9. OPEN PROBLEMS AND FURTHER WORK

We would like to know more about which sets are and are not pterse. We informally conjecture that all sets A such that $F_k^A \in FQ(1, A)$ are "unnatural." Along these lines, in a forthcoming article [3] we show that the hard cores of the non-2-pterse sets (informally) have large "gaps."

The notions of 2-pterseness and p-separability seem linked. For NP-complete sets the notions are similar: non-2-pterseness of A is equivalent to $\bar{A} \times A$ and $A \times \bar{A}$ being p-separable. However, in a forthcoming article [3] we will show that these notions are not equivalent.

It is open if $P \neq NP$ implies $Q(k, SAT) \neq Q(k+1, SAT)$. Krentel [19] has shown that there exists an oracle A such that makes $Q(1, SAT)^A = Q(2, SAT)^A$ and $P^A \neq NP^A$. Since the proof of "$P \neq NP$ implies $FQ(k, SAT) \neq FQ(k+1, SAT)$" relatives, the techniques used in that proof will not suffice to show that $P \neq NP$ implies $Q(k, SAT) \neq Q(k+1, SAT)$.

The class $Q(1, F_k^A)$ appears to be weaker than $Q(k, A)$, but this seems hard to prove. Obtaining various values of A that make these two classes equal or unequal would be of interest.

10. ACKNOWLEDGMENTS

We thank Richard Beigel for helpful discussions and Clyde Kruskal for proofreading. We also thank an anonymous referee for suggestions that culminated in Theorem 13.

RECEIVED December 5, 1987; ACCEPTED July 1987

REFERE

1. AMBOS-SPIES, K. (March 1986), Inhomoger degrees of super sparse sets, *Informat. Proce*
2. AMBOS-SPIES, K., FLEISCHHACK, H., AND H time computable sets, *in* "Proceedings of IC
3. AMIR, A., GASARCH, W. I., AND BEIGEL, manuscript.
4. BEIGEL, R. J. (1987), "Bounded Queries to S
5. BEIGEL, R. J., AND GASARCH, W. I. (1987), Graph Theory," University of Maryland Science, TR-1804.
6. BEIGEL, R. J., GASARCH, W. I., GILL, J. T., Superterse Sets," University of Maryland a Science, TR-1806.
7. BEIGEL, R. J., GASARCH, W. I., AND HAY, L. Hierarchy," manuscript.
8. BLASS, A., AND GUREVICH, Y. (1982), On th *Control* **55**, 80–88.
9. CAI, J., HARTMANIS, J., HEMACHANDRA, Hierarchy I: Structural Properties," manuscri
10. CAI, J., AND HEMACHANDRA, L. (June 1984), *in* "Proceedings of Structure in Complexity T Lecture Notes in Computer Science Vol. 223
11. GASARCH, W. I. (1986), "The Complexity Maryland, Department of Computer Science
12. GESKE, J., HUYNH., AND SELMAN, A. (1987) hard sets, *in* "STACS 1987 Proceedings," Le Verlag, Berlin/New York, in press.
13. GROLLMAN, J., AND SELMAN, A. L. (1984), systems, *in* "Proceedings of the 25th FOCS.
14. HOPCROFT, J., AND ULLMAN, J. (1979), "Int and Computation," Addison–Wesley, Readin
15. JOCKUSCH, C. G. (May 1968a), Semirecursiv *Math. Soc.* **131**, 420–436.
16. JOCKUSCH, C. G. (August 1979), Degree generalisations and applications, *in* "Proce 140–157.
17. KO, K. (1982), On the definition of some *Systems Theory.*
18. KOBLER, J., SCHONING, U., AND WAGNER, K hierarchies for NP, *RAIRO*, in press.
19. KRENTEL, M. W., The complexity of optimi press.
20. KRUSKAL, C. P. (October 1983), Searching, r *IEEE Trans. Comput.* **C-32**, 942–946.
21. LADNER, R. E., LYNCH, N. A., AND SELMAN time reducibilities, *Theoret. Comput. Sci.* **1**,
22. PAPADIMITRIOU, C. H., AND WOLFE, D. (Octo *in* "26th Annual FOCS," pp. 74–78.

REFERENCES

1. AMBOS-SPIES, K. (March 1986), Inhomogeneities in the polynomial-time degrees: The degrees of super sparse sets, *Informat. Process. Lett.* **22**, 113–118.
2. AMBOS-SPIES, K., FLEISCHHACK, H., AND HUWIG, H. Diagonalization over polynomial time computable sets, *in* "Proceedings of ICALP, 1984."
3. AMIR, A., GASARCH, W. I., AND BEIGEL, R. J. (1987), "Polynomial Terse Sets II," manuscript.
4. BEIGEL, R. J. (1987), "Bounded Queries to SAT," manuscript.
5. BEIGEL, R. J., AND GASARCH, W. I. (1987), "Applications of Binary Search to Recursive Graph Theory," University of Maryland at College Park, Department of Computer Science, TR-1804.
6. BEIGEL, R. J., GASARCH, W. I., GILL, J. T., AND OWINGS, J. (1987), "Verbose, Terse, and Superterse Sets," University of Maryland at Colledge Park, Department of Computer Science, TR-1806.
7. BEIGEL, R. J., GASARCH, W. I., AND HAY, L. (1987), "Bounded Queries and the Difference Hierarchy," manuscript.
8. BLASS, A., AND GUREVICH, Y. (1982), On the unique satisfiability problem, *Inform. and Control* **55**, 80–88.
9. CAI, J., HARTMANIS, J., HEMACHANDRA, L., AND SEWELSON, V., "The Boolean Hierarchy I: Structural Properties," manuscript.
10. CAI, J., AND HEMACHANDRA, L. (June 1984), The boolean hierarchy: Hardware over NP, *in* "Proceedings of Structure in Complexity Theory, University of California at Berkeley," Lecture Notes in Computer Science Vol. 223, Springer-Verlag, Berlin/New York.
11. GASARCH, W. I. (1986), "The Complexity of Optimization Functions," University of Maryland, Department of Computer Science, TR-1652.
12. GESKE, J., HUYNH., AND SELMAN, A. (1987), Hierarchy theorems for almost everywhere hard sets, *in* "STACS 1987 Proceedings," Lecture Notes in Computer Science, Springer-Verlag, Berlin/New York, in press.
13. GROLLMAN, J., AND SELMAN, A. L. (1984), Complexity measures for public-key cryptosystems, *in* "Proceedings of the 25th FOCS. [Manuscript]
14. HOPCROFT, J., AND ULLMAN, J. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.
15. JOCKUSCH, C. G. (May 1968a), Semirecursive sets and positive reducibility, *Trans. Amer. Math. Soc.* **131**, 420–436.
16. JOCKUSCH, C. G. (August 1979), Degrees of generic sets, Recursion theory: Its generalisations and applications, *in* "Proceedings of Logic Colloquium 1979, Leeds, 140–157.
17. KO, K. (1982), On the definition of some complexity classes of real numbers, *Math. Systems Theory.*
18. KOBLER, J., SCHONING, U., AND WAGNER, K. W. (1987), The difference and truth-table hierarchies for NP, *RAIRO*, in press.
19. KRENTEL, M. W., The complexity of optimization problems, *J. Comput. System Sci.*, in press.
20. KRUSKAL, C. P. (October 1983), Searching, merging, and sorting in parallel computation, *IEEE Trans. Comput.* **C-32**, 942–946.
21. LADNER, R. E., LYNCH, N. A., AND SELMAN, A. L. (1975), A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1**, 103–123.
22. PAPADIMITRIOU, C. H., AND WOLFE, D. (October 1985), The complexity of facets resolved, *in* "26th Annual FOCS," pp. 74–78.

23. PAPADIMITRIOU, C. H., AND YANNAKAKIS, M. (1984), The complexity of facets (and some facets of complexity), *J. Comput. System Sci.* **28**, 244–259.
24. SCHONING, U. (September 1985), Complexity and structure, "Lecture Notes in Computer Science," Vol. 211, Springer-Verlag, Berlin/New York.
25. SELMAN, A. L. (January 1982), Analogues of semirecursive sets and effective reducibilities to the study of NP complexity, *Inform. and Control* **52**(1), 36–51.
26. SNIR, M. (August 1985), On parallel searching, *SIAM J. Comput.* **14**, 688–708.
27. VALIANT, L. G., AND VAZIRANI, V. V. (1985), NP is as easy as detecting unique solutions, *in* "Proceedings of the 17th Annual STOC," 458–463.
28. WAGNER, K. (1986), More complicated questions about maxima and minima and some closures of NP. *in* "Proceedings of the Thirteenth ICALP, Rennes, France," pp. 434–443, Lecture Notes in Computer Science Vol. 226.
29. WECHSUNG, G. (1986), On the boolean closure of NP, *in* "Proceedings of the 1985 FCT," pp. 485–493, Lecture Notes in Computer Science Vol. 199. [This paper was actually coauthored by K. Wagner]
30. YESHA, Y. (1983), On certain polynomial-time truth table reducibilities and set in NP, *SIAM J. Comput.* **12**, 411–425.

INFORMATION AND COMPUTATION **77**, 57–76 (1988)

Mathematical Optim
Synchronizing Proper

LARRY E. S

Department of Quantitative Business An
Baton Rouge, Lou

Methods for constructing binary exhaust
nizing properties are given. In particular it i
ment that a code be anagrammatic and the
delay as linear integer programming proble
given and the program optimizes over all
examples, solved by a commercially availabl
are included.

I. PRELIMI

If E is a set of symbols then let E^i, $i =$
bol strings of length i over E. The set of
often denoted E^+. By an *encoding* over
ping of S into E^+, and the mapping ma
E^+ elements and the association wit
$E = \{0, 1\}$ an encoding of $\{s_1, s_2, s_3, \ldots,$
$110011\}$, where s_1 is represented by 1,

Elements of W are called code words
are messages. A message from the exam
not *uniquely decipherable* (ud), since
decoded validly in more than one wa
"code" for the ud case.

If a string $e \in E^+$ may be expressed
empty strings α, β, $e = \alpha\beta$, then α is a *pr*
well known if an encoding W has the pr
prefix of another word, the encoding is

A *proper prefix* of W is a non-null pr

A convenient representation of a prefi
tree, which may be drawn as an oriented
code words are described by the sequ
recognized root to all terminal (degree