On Checking Versus Evaluation of Multiple Queries

William I. Gasarch[*] Lane A. Hemachandra[†] Albrecht Hoene[‡]

April 30, 1991

Abstract

The plausibility of *computing* the answers to many membership queries to a hard set with few queries is the subject of the theory of terseness. In this paper, we develop companion theories—both complexity-theoretic and recursion-theoretic—of *characteristic vector terseness.* These theories ask whether the answers to many membership queries to a hard set can be *checked* with fewer queries.

1 Introduction

In recursive function theory there is often no difference between checking and evaluating a single instance of a function. However, the distinction between checking and evaluating is fundamental in computational complexity theory. In its most popular version, this distinction emerges as the P = NP question, but it also reappears in other problems of the field [Val76,IT89]. We introduce the distinction between checking and evaluating to the theory of terseness.

The theory of terseness studies whether it is possible—using only a few questions—to compute the answers to many questions to an oracle. Pioneering research in this area was done by Bārzdiņš [Bār68]—and, more recently, by Beigel, Gasarch, Gill, and Owings [BGGO]—for a recursion-theoretic context, and by Amir and Gasarch [AG88] and

[*]Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported in part by the National Science Foundation under research grants CCR-8803641 and CCR-9020079.

[†]University of Rochester, Department of Computer Science, Rochester, NY 14627. Research supported in part by the National Science Foundation under research grants CCR-8809174, CCR-8996198, and CCR-8957604.

[‡]Technische Universität Berlin, Fachbereich Informatik, D-1000 Berlin 10. Research supported in part by a Deutsche Forschungsgemeinschaft Postdoktorandenstipendium.

Beigel [Bei87] for a polynomial-time framework. These papers study the complexity of *evaluating* the values of (many) queries—that is, of *computing* the answers to the queries.

This paper develops a theory of characteristic vector terseness—the 'checking' counterpart to the theory of terseness. Given a vector of queries to an oracle and proposed answers to the queries, we want to know whether all the answers are correct; that is, we wish to determine whether the vector of answers is the characteristic vector of the queries relative to the considered language. The minimal number of queries that suffice for this purpose is called the *characteristic vector cost* of the language considered.

Section 2 studies the problem for nonrecursive sets. In this context the appropriate question for characteristic vector cost is: What is the minimum number j such that the answers to n queries to A can be checked with j queries to A? We show that from a result of [Rog67] it follows that the characteristic vector cost of K is exactly two. This strengthens the result of [BGGO] (see also [Bār68]) that *computing* the solutions to two queries to K requires two queries to K. Other results on the characteristic vector cost of nonrecursive languages are also proven; in particular, for each k we show that there are sets whose characteristic vectors can be checked with k queries but not with $k-1$.

In the resource-bounded world of feasible computations, Section 3 explores the analogous question for polynomial-time oracle machines, and finds that the characteristic vector cost of a language is closely related to its basic set-theoretic properties. Our results show a connection between characteristic vector costs and boolean hierarchies [Hau14,CGH$^+$88, CGH$^+$89]. For reasonably well-behaved classes C, the characteristic vectors of sets in C can be checked with one query if and only if the boolean hierarchy over C collapses in a certain way.

We next consider the characteristic vector cost of NP. If characteristic vectors of NP languages are recognizable *uniformly* (that is, via a single polynomial-time oracle machine) with one query to a language of the same class, then we can derive a strong conclusion: all languages polynomial-time truth-table reducible to a language L of the considered class are in fact 1-truth-table reducible to L. Finally, we investigate some specific classes within the boolean hierarchy over NP and show how their characteristic vector costs relate to the existence of a proper boolean hierarchy over NP.

Before presenting our results, let us introduce some notation. $\langle \cdot, \cdot, \ldots, \cdot \rangle$ applied to k arguments ($k \geq 2$) denotes the polynomial-time computable k-ary pairing function from $N^k \rightarrow N$ used in [Rog67]. We make use of these functions for strings as well by identifying $i \in N$ with $string(i)$, which stands for the i-th binary string in lexicographical order;

henceforth, when we use the pairing functions on strings, we tacitly assume that the function *string* and its inverse are being used to convert the strings to integers and to convert the output of the pairing function back to a string. As to singleton arguments, by convention we'll consider $\langle x\rangle$ to represent x. For any language $A \subseteq \Sigma^*$, we denote by $\overline{A}$ its complement and by A^k the set $\{\langle x_1, \ldots, x_k\rangle \mid x_i \in A \text{ for } 1 \leq i \leq k\}$. For sets $A_i \subseteq \Sigma^*$ the k-fold disjoint union $A_1 \oplus A_2 \oplus \ldots \oplus A_k$ is defined by $\cup_{i=1}^{k} \{\langle string(i), x\rangle \mid x \in A_i\}$.

P (NP) denotes the class of languages that can be accepted by deterministic (nondeterministic) polynomial-time Turing machines [HU79]. P^C and NP^C refer to the analogous classes relative to an oracle from the class C [BGS75]. If during such a computation only a limited number of oracle queries are permitted (say k), we indicate that by the additional superscript $[k]$, for example, $\mathrm{P}^{\mathrm{NP}[k]}$ [PZ83]. In general, the classes dealt with are either well-known or will be defined explicitly as needed.

We now propose and study an "answer checking" analog of the [Bār68,BGGO,AG88] theory of terseness.

2 Recursion-Theoretic Results on Characteristic Vectors

We study the number of queries required to verify whether a given string is a characteristic vector of a nonrecursive set.

Definition 2.1 If $A \subseteq \Sigma^*$ and $x_1, \ldots, x_i \in \Sigma^*$ then let

$$F_i^A(x_1, \ldots, x_i) = \langle \chi_A(x_1), \ldots, \chi_A(x_i)\rangle,$$

$$V_i^A = \{\langle x_1, \ldots, x_i, b_1, \ldots, b_i\rangle \mid (\forall j)[\chi_A(x_j) = b_j]\},$$

and

$$V_\omega^A = \bigcup_{i=1}^{\infty} \{\langle i, v\rangle \mid v \in V_i^A\}.$$

where χ_A is the characteristic function of A.

Definition 2.2 [BGGO] Let $i \in N$ and let A be a set. $Q(i, A)$ is the collection of all sets B such that $B \leq_T A$ and this Turing-reduction can be performed by a machine that makes at most i serial queries to A. $Q_{||}(i, A)$ is the collection of all sets B such that $B \leq_T A$ and this Turing-reduction can be performed by a machine that makes at most i parallel (i.e., truth-table) queries to A.

A language L is said to be *k-pterse* [AG88] if a polynomial-time oracle machine cannot in general *compute* the correct answers to an input set of k queries with fewer than k queries to the oracle L; if L is k-pterse for each $k \geq 2$ then it is *pterse* [AG88]. We now introduce a notion—*vterseness*—original to this paper, that allows us to define and study the "checking" analog of pterseness.

Definition 2.3 A set A is *vterse* if for all $i \geq 1$, $V_i^A \notin Q(i-1, A)$.

Definition 2.3 defines vterseness in terms of sequential (adaptive) oracle queries. It might also be natural to define vterseness in terms of parallel queries; that is, a set might be said to be $vterse_{||}$ if for all $i \geq 1$, $V_i^A \notin Q_{||}(i-1, A)$. However, we are concerned with proving that certain sets *are* vterse; thus, the definition we adopt is more demanding, and results about parallel vterseness follow implicitly.

We are interested in how many queries to A are required to decide the set V_i^A (and V_ω^A). Naively it appears that for sets A that are nonrecursive, i queries might be required.

We first examine r.e. sets. How many queries are required to compute V_i^A if queries to an r.e. set other than A are allowed? Let K be the halting set: $K = \{i \mid M_i(i) \text{ halts}\}$.

Theorem 2.4 If A is r.e. and $i \in N$, then $V_i^A \in Q_{||}(2, K)$.

Proof: Given $\langle x_1, \ldots, x_i, b_1, \ldots, b_i \rangle$ create machines M_{y_1} and M_{y_2} such that M_{y_1} enumerates A and stops only if all the elements of $\{x_j \mid b_j = 1\}$ appear, and M_{y_2} enumerates A and stops only if some element of $\{x_j \mid b_j = 0\}$ appears. It is easy to see that $\langle x_1, \ldots, x_i, b_1, \ldots, b_i \rangle \in V_i^A$ if and only if $(y_1 \in K) \wedge (y_2 \notin K)$. ▮

The above proof in fact shows that V_i^A is 2-r.e. (see [EHK81] for a definition of 2-r.e.). Theorem 2.4 is optimal for $A = K$ in that the set V_2^K requires two queries to K. The following proof of Theorem 2.5, pointed out to us by Richard Chang, is more direct than that found in earlier versions of this paper.

Theorem 2.5 $V_2^K \notin Q(1, K)$.

Proof: It is known (see [Rog67]) that the m-degree of $K \times \overline{K}$ contains the m-degree of $K \oplus \overline{K}$ properly. By the fact that $K \times \overline{K}$ many-one reduces to V_2^K, it follows that every language in the m-degree of $K \times \overline{K}$ many-one reduces to V_2^K. Since every language in $Q(1, K)$ many-one reduces to $K \oplus \overline{K}$ it follows that $V_2^K \notin Q(1, K)$.

▮

It is easy to see that certain sets are necessarily vterse. For example, a class of sets for which $(\forall i)[A^{i+1} \notin Q(i, A)]$—namely the 1-generic sets (see [Joc79])—was identified in [BGGO]; thus these sets are all vterse. Since the class of 1-generic sets is comeager [Joc79], the class of vterse sets is also comeager.

The following theorem shows that non-vterse sets are ubiquitous.

Theorem 2.6 Every truth-table degree (see [Rog67]) contains a set A such that $V_\omega^A \in Q(1, A)$.

Proof: Let D be any set. We construct $A \equiv_{tt} D$ such that $V_\omega^A \in Q(1, A)$. Let $A_0 = D$. For $j \geq 0$ let $A_{j+1} = \{\langle y_1, b_1, m_1, \ldots, y_w, b_w, m_w, w, j+1\rangle \mid w \geq 1$ and $(\forall h)[1 \leq h \leq w \Rightarrow [(0 \leq m_h \leq j)$ and $(F_1^{A_{m_h}}(y_h) = b_h)]]$. Set $A = \cup_{j=1}^{\infty} A_j$ (note that A_0 is excluded). It is easy to see that $A \equiv_{tt} D$ and that

$$\langle y_1, \ldots, y_w, b_1, \ldots, b_w\rangle \in V_\omega^A \text{ iff } \langle y_1,\, b_1,\, q_1,\, \cdots,\, y_w,\, b_w,\, q_w,\, w,\, q+1\rangle \in A,$$

where the q_ℓ are such that $y_\ell = \langle r_\ell, q_\ell\rangle$ and q is the maximum value of the last component of any y_j (since we are adopting the recursively defined pairing functions of Rogers, this can be written $q = \max_{1 \leq h \leq j}\{q_h \mid y_h = \langle r_h, q_h\rangle\}$). ∎

Combining a diagonalization scheme with an extension of the coding technique of Theorem 2.6, we now generalize and strengthen the claim of Theorem 2.5. In particular, for each k, we show that there are sets A whose (unbounded) characteristic vectors can be checked with k parallel queries to A, but cannot be checked with $k-1$ parallel queries—or even $k-1$ sequential queries—to A. That is, we show that, for each k, there exists a set A such that $V_\omega^A \in Q_{||}(k, A) - Q(k-1, A)$. In a certain sense, this is a strongest possible separation.

To ensure that $V_\omega^A \in Q_{||}(k, A)$, our proof constructs A so that A codes information about itself. Our construction will rely on the following machinery.

Notation 2.7 For each $k \in N$, fix partition schemes (based on k) and a function c_k, as specified below.

1. Recursively partition Σ^* into k infinite parts $Z_1, \ldots, Z_k$ as follows (note that we'll tacitly use the correspondence between strings and natural numbers). For each $1 \leq i \leq k-1$, let $Z_i = \{\langle i, a, b\rangle \mid a, b \in N\}$. Also, let $Z_k = \{\langle i, a, b\rangle \mid a, b \in N,\ i \geq k\}$.

2. For each $1 \leq i \leq k$, recursively partition Z_i into an infinite number of infinite parts $Z_{i1}, Z_{i2}, \ldots$ as follows (we think of each Z_i as an infinite matrix whose hth row consists

of the elements of Z_{ih}). For $1 \leq i \leq k-1$ and $j \geq 1$, let $Z_{ij} = \{\langle i, j, b\rangle \mid b \in N\}$. Also, for $j \geq 1$ let $Z_{kj} = \{\langle i, j, b\rangle \mid b \in N,\ i \geq k\}$.

3. We call the elements of any Z_{ij} *elements of rank* j. For each $j \geq 2$ and $1 \leq i \leq k$, let $\widehat{Q}_{ij} = \{\langle m, z, j\rangle \mid (m \geq 2) \wedge (\exists\, y_1, \ldots, y_m \in \bigcup_{\widehat{d}<j} Z_{i\widehat{d}})\,(\exists\, b_1, \ldots, b_m \in \{0,1\})\,[z = \langle y_1, \ldots, y_m, b_1, \ldots, b_m\rangle\,]\,\}$. Intuitively, $\widehat{Q}_{ij}$ is a "tag" of j together with all tuples of queries/purported-answers such that each query name is a string of Z_i and of rank less that j.

   We'll now define $c_k(x)$ so that it map each Z_{ij} $(1 \leq i \leq k,\ j \geq 2)$ one-to-one onto $\widehat{Q}_{ij}$, so that the elements of Z_{ij} can be interpreted as vectors of queries/purported-answers. Eventually, this will allow us—in certain cases—to compress many membership queries/purported-answers about $Z_i \bigcap A$ into a single query to some element of Z_i. For $j \geq 2$, $1 \leq i \leq k$, and $x \in Z_{ij}$, define $c_k(x)$ to be the lexicographically uth smallest string in $\widehat{Q}_{ij}$, where x is the uth smallest string lexicographically in Z_{ij}.

   Note that c_k is a recursive bijection from domain $\Sigma^* - \bigcup_{1 \leq i \leq k} Z_{i1}$ onto the range $\bigcup_{1 \geq i \leq k,\ j \geq 2} \widehat{Q}_{ij}$. This bijection has the following "sub-bijection" property: for each $j \geq 2$ and $1 \leq i \leq k$, it holds that c_k restricted to Z_{ij} is a one-to-one onto map between Z_{ij} and $\widehat{Q}_{ij}$.

Since the functions c_k defined above are recursive bijections (relative to the range and domain specified), they have recursive inverses (c_k^{-1}). We re-emphasize the fact that c_k always maps elements of rank j to strings of the form $\langle m, \langle y_1, \ldots, y_m, b_1, \ldots, b_m\rangle, j\rangle$ where the y_i are elements of rank strictly less than j (elements of rank 1 are not within the domain of c_k).

Definition 2.8 Let $k \in N$. Let c_k be as defined above. A set A is *k-self-coding* if, for every x of rank $j \geq 2$,

$$x \in A \text{ if and only if } (c_k(x) = \langle m, z, j\rangle \text{ and } \langle m,\ z\rangle \in V_\omega^A).$$

(Intuitively, membership information in A for elements in the jth row of Z_i can be obtained from the first $j-1$ rows of $A \cap Z_i$.)

Lemma 2.9 Let $k \in N$. If A is k-self-coding then $V_\omega^A \in Q_{||}(k, A)$.

Proof: Given $\langle y_1, \ldots, y_m, b_1, \ldots, b_m\rangle$—with $y_i \in \Sigma^*$ and $b_i \in \{0,1\}$ for $1 \leq i \leq m$—we may assume, by implicitly shuffling the labels of both the y_js and b_js, that there exists an integer l, $1 \leq l \leq k$, and integers $m_1, m_2, \ldots, m_{l-1}$, $1 \leq i_1 < i_2 < \cdots < i_l \leq k$ such that, in the case $l \geq 2$ (the $l = 1$ case is immediate):

$y_1, \ldots, y_{m_1} \in Z_{i_1}$,

$y_{m_1+1}, \ldots, y_{m_2} \in Z_{i_2}$,

$\vdots$

$y_{m_{l-1}+1}, \ldots, y_m \in Z_{i_l}$.

For each $1 \leq p \leq l$, let r_p be the maximum rank of any of the y_j in Z_{i_p} above. Note that $\langle m, \langle y_1, \ldots, y_m, b_1, \ldots, b_m \rangle\rangle \in V_\omega^A$ if and only if

$c_k^{-1}(\langle m_1, \langle y_1, \ldots, y_{m_1}, b_1, \ldots, b_{m_1}\rangle, 1 + r_1\rangle) \in A \wedge$

$c_k^{-1}(\langle m_2 - m_1, \langle y_{m_1+1}, \ldots, y_{m_2}, b_{m_1+1}, \ldots, b_{m_2}\rangle, 1 + r_2\rangle) \in A \wedge$

$\vdots$

$c_k^{-1}(\langle m - m_{l-1}, \langle y_{m_{l-1}+1}, \ldots, y_m, b_{m_{l-1}+1}, \ldots, b_m\rangle, 1 + r_l\rangle) \in A$.

Hence $V_\omega^A \in Q_{||}(k, A)$. ∎

Lemma 2.10 If A is k-self-coding then $A \leq_{tt} A \bigcap (\bigcup_{i=1}^{k} Z_{i1})$ by a specific reduction f such that if $x \in Z_{i_0}$ then all strings mentioned in $f(x)$ are in $Z_{i_0 1}$. The reduction f below works for any k-self-coding A. (Intuitively, A is truth-table reducible to the elements of A of rank 1.)

Proof:

Let x be any string. If x is of rank 1 then $x \in A$ if and only if $x \in A \bigcap (\bigcup_{i=1}^{k} Z_{i1})$. If x is of rank $j > 1$, and $x \in Z_{i_0}$, then x is of the form $c_k^{-1}(\langle m, \langle y_1, \ldots, y_m, b_1, \ldots, b_m\rangle, j\rangle)$, where all the y_i are in $Z_{i_0 1} \bigcup Z_{i_0 2} \bigcup \cdots \bigcup Z_{i_0 (j-1)}$. Since A is k-self-coding,

$$x \in A \text{ if and only if } (\forall i : 1 \leq i \leq m)[\chi_A(y_i) = b_i].$$

For each y_i of rank greater than 1, each of the statements '$\chi_A(y_i) = b_i$' can be similarly reduced. Eventually, the statement '$x \in A$' is seen to be equivalent to a boolean combination of statements of the form '$y \in A$' where every y is in $Z_{i_0 1}$. Each such '$y \in A$' is equivalent to '$y \in A \bigcap Z_{i_0 1}$.' Hence $A \leq_{tt} A \bigcap (\bigcup_{i=1}^{k} Z_{i1})$ by a reduction f such that if x is in Z_{i_0}, then all strings mentioned in $f(x)$ are in $Z_{i_0 1}$.

Note that the truth-table reduction specified is correct for any k-self-coding set A.

∎

By the above lemma, any k-self-coding set is determined by its rank 1 elements.

Theorem 2.11 For every $k \in N$ there exists a set $A \leq_T K$ such that $V_\omega^A \in Q_{||}(k, A)$, but $A^k \notin Q(k-1, A)$.

Proof: Let $M_0^{()}, M_1^{()}, M_2^{()}, \ldots$ be a standard list of all oracle Turing machines. If X is a set and M_i^X is total and 0-1 valued then let $\mathrm{L}(M_i^X)$ be the language decided by M_i^X.

We construct a k-self-coding set A that satisfies the requirements:

R_e: $(\forall x)[M_e^A(x)$ makes $\leq k-1$ queries and converges$]\Rightarrow \mathrm{L}(M_e^A) \neq A^k$.

At the end of every stage e we will have a pair of disjoint finite sets $(A_e^1, \widehat{A}_e^1)$ such that A_e^1 is the set of rank 1 elements we have determined are in A, and $\widehat{A}_e^1$ is the set of rank 1 elements we have determined are not in A. The construction determines the membership in A of all strings of rank 1, hence by Lemma 2.10 the membership in A of all strings has been determined.

CONSTRUCTION OF A

Stage 0: $A_0^1 = \emptyset$, $\widehat{A}_0^1 = \emptyset$.

Stage e+1: (We satisfy R_e during this stage.) For $1 \leq i \leq k$, let x_i denote the least element of Z_{i1} such that $x_i \notin A_e^1 \bigcup \widehat{A}_e^1$. We ensure that either:

1. $M_e^A(\langle x_1, \ldots, x_k \rangle)$ diverges, or
2. the computation of $M_e^A(\langle x_1, \ldots, x_k \rangle)$ asks more than $k-1$ questions, or
3. $M_e^A(\langle x_1, \ldots, x_k \rangle) = 1$ and some $x_i \notin A$, or
4. $M_e^A(\langle x_1, \ldots, x_k \rangle) = 0$ and for all i such that $1 \leq i \leq k$, it holds that $x_i \in A$.

It is easy to see that any of these outcomes ensure that R_e is satisfied.

We simulate $M_e^A(\langle x_1, \ldots, x_k \rangle)$. Since we have only incomplete information about A, we may determine $A(x)$ for some values of x during the simulation. To avoid notational difficulties the term A_e^1 $(\widehat{A}_e^1)$ refers to the finite set A_e^1 $(\widehat{A}_e^1)$ at the end of stage e, unioned with all strings that have been placed into A_e^1 $(\widehat{A}_e^1)$ at this stage $(e+1)$ up to the current point.

Since we are merely required to ensure that $A \leq_T K$, we may use K as an oracle during the construction. Thus, we may assume that our simulation of $M_e^A(\langle x_1, \ldots, x_k \rangle)$ converges, outputs either 0 or 1, and never makes more than $k-1$ queries. (That is, initially and after each query is made and answered, we ask K, 'Will the computation ask any more queries?' and 'Will the computation converge from the current configuration?' If we ever discover that either a kth query is to be asked, or that the computation does not converge, then the simulation halts and outputs 0 (by convention). If the computation runs normally to completion, then if the output is nonzero, change it to 1.) Our simulation will not mention this explicitly.

We simulate $M_e^A(\langle x_1, \ldots, x_k \rangle)$ as follows. Run the computation. Whenever a query q is encountered we do the following. Let i_0 be such that $q \in Z_{i_0}$. Since we intend that A be k-self-coding we want, by Lemma 2.10,

$$q \in A \text{ if and only if } \psi(z_1, \ldots, z_m)$$

where $\psi(z_1, \ldots, z_m)$ is a boolean combination of statements (by Lemma 2.10's "unrolling" of applications of c_k) of the form '$z_i \in A$,' and each z_i is in $Z_{i_0 1}$. Put all elements of $\{z_1, \ldots, z_m\}$ that are not in $\widehat{A}_e^1$ into A_e^1. (Note that since all the z_i are in $Z_{i_0 1}$, at most one of $\{x_1, \ldots, x_k\}$ was put into A_e^1, namely x_{i_0}. This is because, by the definition above of the x_i, at most one x_i is in each Z_h.) Now the membership of q in A has been determined and the query can be answered.

Since at most $k-1$ queries are made, at most $k-1$ of the x_i are placed into A_e^1 (none are placed into $\widehat{A}_e^1$) during the simulation. Hence there exists a j such that the status of x_j in A is not determined.

At the end of the simulation, if $M_e^A(\langle x_1, \ldots, x_k \rangle) = 0$ then let A_{e+1}^1 be $A_e^1 \bigcup \{x_1, \ldots, x_k\}$, and $\widehat{A}_{e+1}^1$ be $\widehat{A}_e^1$ If $M_e^A(\langle x_1, \ldots, x_k \rangle) = 1$ then let $\widehat{A}_{e+1}^1$ be $\widehat{A}_e^1 \bigcup \{x_j\}$, and A_{e+1}^1 be A_e^1. (Recall that, in to avoid notational problems, we let A_e^1 ($\widehat{A}_e^1$) be what A_e^1 ($\widehat{A}_e^1$) was at the end of stage e, together with whatever strings we had added up to that point.)
END OF CONSTRUCTION

Let A be the k-self-coding set whose rank 1 elements are $\bigcup_{e \geq 1} A_e^1$. That is, let A be the set that satisfies $(\bigcup_{i \geq 0} Z_{i1}) \bigcap A = \bigcup_{e \geq 1} A_e^1$ and that is k-self-encoded by the fixed c_k and partition scheme of Notation 2.7. By the nature of the construction, it is clear that A is k-self-coding, and that each R_e is satisfied. Since A is k-self-coding it holds that $V_\omega^A \in Q_{||}(k, A)$, but by construction $A^k \notin Q(k-1, A)$. ▮

Theorem 2.12 Let $k \in N$. Every Turing degree above (or equal) to that of K contains a set A such that $V_\omega^A \in Q_{||}(k, A)$, but $A^k \notin Q(k-1, A)$. Thus, in particular, $V_\omega^A \in Q_{||}(k, A) - Q(k-1, A)$. ▮

Proof:

Let $\mathbf{b}$ be a Turing degree above (or equal) to K, and let $B \in \mathbf{b}$. We modify the proof of Theorem 2.11 by initially recursively coding B into a recursive subset of Z_{11}. Assume, without loss of generality, that there exists infinite recursive sets U and V such that

$$Z_{11} = \{x \mid (\exists u \in U)\ [x = 0u]\} \bigcup \{x \mid (\exists v \in V)\ [x = 1v]\}.$$

Let d be a recursive bijection from U to Σ^*. During stage 0 of the construction let $A_0^1 = \{0u \mid d(u) \in B\}$ and $\widehat{A}_0^1 = \{0u \mid d(u) \notin B\}$. For the rest of the construction we think of A_e^1 ($\widehat{A}_e^1$) as being a finite list of rank 1 elements, together with A_0^1 ($\widehat{A}_0^1$). Whenever we need to know if some z is in A_e^1 ($\widehat{A}_e^1$), if $z \notin 0U$ then we use the finite list; if $z \in 0U$ then we use the oracle B (i.e., if $z = 0u$, $u \in U$, then (1) $z \in A_e^1$ if and only if $d(u) \in B$, and (2) $z \in \widehat{A}_e^1$ iff $d(u) \notin B$). Hence the construction is recursive in $K \oplus B \leq_T B$, so $A \leq_T B$.

By the coding at stage 0, $B \leq_m A$: $z \in B$ if and only if $0d^{-1}(z) \in A$. ∎

Corollary 2.13 Let $k \in N$. For every Turing degree $\mathbf{b}$ such that $\mathbf{K} \leq_T \mathbf{b}$ there exists a set $A \in \mathbf{b}$ such that $V_\omega^A \in Q(k, A) - Q(k-1, A)$.

We now examine vterseness for r.e. sets.

Theorem 2.14 Every r.e. Turing degree contains r.e. sets A and B such that $V_\omega^A \in Q_{||}(2, A)$ and B is vterse.

Proof: Every r.e. Turing degree contains an r.e. set that is semirecursive, i.e., a lower cut of a recursive linear ordering [Joc68]. Let A be that set, and let $<$ denote the recursive linear ordering. We show that $V_\omega^A \in Q_{||}(2, A)$.

Given $\langle x_1, \ldots, x'_y, b_1, \ldots, b'_y \rangle$, check to see that for every repeated element we have consistent claims about the element's membership in the set (that is, check that $x_k = x_l \Rightarrow b_k = b_l$). If this is not the case, the given characteristic vector is not in V_ω^A. Otherwise, remove all but one copy of each duplicated element and rename the arguments such that $\hat{x}_1 < \hat{x}_2 < \cdots < \hat{x}_y$, and rename the b_j's via the same permutation. If there is an i such that $\hat{b}_i = 0$ and $\hat{b}_{i+1} = 1$, then the given characteristic vector is not in V_ω^A. If no such i exists, then either (1) there exists a j such that $\hat{b}_1 = \hat{b}_2 = \cdots = \hat{b}_j = 1$ and $\hat{b}_{j+1} = \cdots = \hat{b}_i = 0$ in which case it is easy to see that the given characteristic vector is in V_ω^A if and only if $(\hat{x}_j \in A) \wedge (\hat{x}_{j+1} \notin A)$, or (2) $\hat{b}_1 = \hat{b}_2 = \cdots = \hat{b}_y = 0$ in which case the given characteristic vector is in V_ω^A if and only if $(\hat{x}_1 \notin A)$, or (3) $\hat{b}_1 = \hat{b}_2 = \cdots = \hat{b}_y = 1$ in which case the given characteristic vector is in V_ω^A if and only if $(\hat{x}_y \in A)$.

In [BGGO] it was shown that every r.e. Turing degree contains an r.e. set B such that for all i the set $\mathrm{PARITY}^B_{i+1} = \{\langle x_1, \ldots, x_{i+1}\rangle \mid B \cap \{x_1, \ldots, x_{i+1}\}$ contains an even number of elements$\}$ is not in $Q(i, B)$. A careful examination of the proof reveals that $\overline{B}^{i+1}$ is not in $Q(i, B)$. Hence for all i, $V^B_{i+1} \notin Q(i, B)$, so B is vterse. ∎

By way of contrast, it follows easily from Theorem 2.5 that some m-degrees—e.g., that of the r.e. complete sets—contain no sets A for which $V^A_\omega \in Q(1, A)$. Though it follows from Theorem 2.6 that every r.e. Turing degree contains a set A such that $V^A_\omega \in Q(1, A)$, it remains an open question whether every r.e. Turing degree contains a *recursively enumerable* set A such that $V^A_\omega \in Q(1, A)$.

3 Characteristic Vector Complexity and Boolean Hierarchies

To what extent do the results of the last section hold in a resource-bounded framework? The last section's questions about characteristic vector complexity become, in a time-bounded world: Given a vector of queries and a vector of purported answers to the queries, how many queries does a polynomial-time oracle machine need to check whether all the answers are correct? We are primarily interested in classes that lie above P and are reasonably well-behaved.

Definition 3.1 We call a class C *interesting* if:

1. $\mathrm{P} \subseteq C$,
2. C is closed under disjoint union, i.e., if $L_1, \ldots L_k \in C$ then $L_1 \oplus L_2 \oplus \ldots \oplus L_k \in C$,
3. C is closed under cylindrification,[1] i.e., for any language $L \in C$ the sets $L_1 = \{\langle x, y\rangle \mid x \in L, y \in \Sigma^*\}$ and $L_2 = \{\langle x, y\rangle \mid x \in \Sigma^*, y \in L\}$ are in C, and
4. C is closed downwards under $\leq^p_m$ reductions, i.e., $L' \in C$ and $L \leq^p_m L'$ implies $L \in C$.

We now define a complexity-theoretic notion of characteristic vector cost. Our approach here differs from that of the previous section. That section, dealing with the recursion-theoretic case, defined characteristic vector costs for sets. In contrast, we define complexity-theoretic costs for classes—that is, when trying to test characteristic vectors of an (arbitrary)

[1]This is just for convenience, as closure under cylindrification follows from part 4. (The minor difficulty that a non-onto pairing function would cause the cylindrification of Σ^* not to many-one reduce to Σ^* is not a problem due to our onto pairing functions, or, alternatively, due to part 1's guarantee that the class contains sets other than the empty set and Σ^*.)

set from a class, we allow as our oracle any set from that class (and in particular, we allow oracles other than the set actually being tested).

This has an effect on the form of our results. For example, though many queries to any NP set can be checked with two queries to SAT, it is easy to construct relativized worlds A in which specific NP^A sets are relativized vterse. Our formulation blurs the latter case—which reflects the complexity of computing with specific sets whose information is not well-organized—in order to focus on the former case—which reflects the striking ability of complete sets to provide a uniform and structured approach to the classes for which they are complete.

Definition 3.2 Let C be any class of sets.

1. The *characteristic vector cost* of a class C is:

   $CV_C(k) \;=\; \min_{i\geq 0}\{\, i : \text{for all}\, L \,\in\, C,\, V_k^L \,\in\, \mathrm{P}^{C[i]}\,\}.$

2. $CV_C^+(k) = \min_{i\geq 0}\{\, i\text{: for all } L \,\in\, C,\; L^k \,\in\, \mathrm{P}^{C[i]}\,\}.$

3. $CV_C^-(k) = \min_{i\geq 0}\{\, i\text{: for all } L \,\in\, C,\; \overline{L}^k \,\in\, \mathrm{P}^{C[i]}\,\}.$

4. $CV_C = \min_{i\geq 0}\{\, i\text{: for all } L \in C,\; V_\omega^L \in \mathrm{P}^{C[i]}\,\}.$

We observe that the behavior of a class under basic set-theoretic operations is related to its characteristic vector cost. Note that, in contrast with Section 2, the machines accessing C here are constrained to run in polynomial time.

Proposition 3.3 Let C be an interesting class other than P. If C is closed under:

1. union (intersection) then for all k: $CV_C^-(k) = 1$ ($CV_C^+(k) = 1$).

2. complement then for all k: $CV_C^+(k) = CV_C^-(k) = CV_C(k)$.

3. union and complement (equivalently, intersection and complement) then for all k: $CV_C(k) = 1$.

4. union and intersection then for all k: $CV_C(k) \leq 2$.

Proof:
1. By the assumption that C is different from P, we know that $CV_C^+(k)$ and $CV_C^-(k)$ are greater than 0 if $k \geq 1$. Let L be a language in C. Since C is closed under cylindrification,

$L' = \{\langle x, y\rangle \,|\, x \in L, y \in \Sigma^*\}$ and $L'' = \{\langle x, y\rangle \,|\, x \in \Sigma^*, y \in L\}$ are in C. By the assumption that C is closed under union the set $L' \cup L'' = \{\langle x, y\rangle \,|\, x \in L \text{ or } y \in L\}$ belongs to C. On an input $\langle x, y\rangle$ an oracle machine can check with one query to this oracle if both x and y do not belong to L. This is easily seen to work for arbitrary k. For intersection the proof is analogous.

2. It suffices to show that $CV_C(k) \leq CV_C^+(k)$. Let $L \in C$, C closed under complement. Clearly $V_k^L \leq_m^p (L \oplus \overline{L})^k \in \mathrm{P}^{C[CV_C^+(k)]}$. Thus $CV_C(k) \leq CV_C^+(k)$.

3. and 4. are immediate by 1. and 2. In particular, part 4 holds because in this case $CV_C(k) \leq CV_C^+(k) + CV_C^-(k) \leq 1 + 1$. ∎

The proposition indicates that the characteristic vector costs of a class reflect its closure properties under the corresponding operations. Another way to view results such as Proposition 3.3 is as a study of which assumptions about C are needed to make $CV_C(k)$ coincide for different k (right down to $k = 1$ or $k = 2$). For example, part 4 of Proposition 3.3 says that $CV_C(k) = CV_C(2)$ for all $k \geq 2$, when C is closed under union and intersection. Similarly, if C is closed under the constructions:

$$L \rightarrow \{x_1\#x_2\#\cdots\#x_z \,|\, (\exists i)[x_i \in L]\}$$

$$L \rightarrow \{x_1\#x_2\#\cdots\#x_z \,|\, (\forall i)[x_i \in L]\}$$

then $CV_C(k) = CV_C(2) = CV_C$ for all $k \geq 2$.

Definition 3.4 We say that an interesting class C is *cv-wee*[2] *under complement* if for all $k \geq 1 : CV_C^+(k) = CV_C^-(k) = CV_C(k)$.

Definition 3.5 A class C has *wee* characteristic vector cost if, for all $k \geq 1$, $CV_C(k) = 1$.

From Proposition 3.3 it follows that among the classes with wee characteristic vector cost are all interesting deterministic classes that are defined by time or space bounds, parity polynomial time ($\oplus$P, defined in [PZ83,GP86]), ZPP [Gil77], the Δ_k^p-classes of the polynomial hierarchy, and so on, because they are each closed under complement and union.

Also by Proposition 3.3 the characteristic vector cost is not higher than two for classes such as NP, the Σ_k^p and Π_k^p levels of the polynomial hierarchy [Sto77], and FewP ([AR88],

[2]We use the term "wee," rather than "small," to avoid confusion with the technical meaning sometimes assigned to the word "small" (for example, in Rubinstein's recent work on "small generalized Kolmogorov complexity" [Rub90]).

see also [Rub88,CH90]), since they are closed under union and intersection. Among the classes for which the number of queries can not be reduced straightforwardly are US (which tests for unique solutions, [BG82], see also [GW87]) and the classes of the boolean hierarchy over NP [CGH$^+$88,CGH$^+$89]. Their vector checking cost will be studied in Theorem 3.15 and its corollaries.

We've seen for many well-known classes $\mathcal{C}$ that $CV_{\mathcal{C}}(k) \leq 2$. We now begin a series of results, leading to the conclusion that $CV_{\mathcal{C}}(k) \leq 1$ would imply surprising structural consequences. For example, if $CV_{\mathrm{NP}}(2) \leq 1$, then the polynomial hierarchy would collapse.

We now turn to boolean hierarchies over complexity classes. Among them, the boolean hierarchy over NP has received the most attention [CGH$^+$88,CGH$^+$89]. However, the same definitions can be used to define boolean hierarchies over arbitrary classes [Hau14,Wag88, BBJ$^+$]:

Definition 3.6 For a class C, we define:

1. $\mathrm{BH}_C(1) = C$.
2. $\mathrm{BH}_C(k) = \{ L_1 - L_2 \,|\, L_1 \in C,\, L_2 \in \mathrm{BH}_C(k-1)\}$ for $k > 1$.
3. BH_C represents the boolean closure of C; that is, the closure of C under boolean operations ($\bigcup$, $\bigcap$, complementation).

For the case $C = \mathrm{NP}$, it holds that $\mathrm{BH}_{\mathrm{NP}} = \bigcup_k \mathrm{BH}_{\mathrm{NP}}(k)$, and the definition of the hierarchy's levels given in Definition 3.6 is equivalent to many alternate definitions [CGH$^+$88, Sections 2.1 and 2.2]; however, this is not necessarily true for arbitrary classes C.

We state the following folk theorem related to the work of Beigel [Bei], Bertoni *et. al.* [BBJ$^+$], Cai *et. al.* [CGH$^+$88], and Kobler *et. al.* [KSW87]; there has been much confusion in the literature, due to the fact that the various definitions of the boolean hierarchy are not interchangeable for arbitrary classes.

Proposition 3.7 Let C be any class of sets such that $C - \{\emptyset, \Sigma^*\}$ is non-empty and closed downwards under many-one reductions. For all $k \geq 1 : \mathrm{P}^{C[k]} \subseteq \mathrm{BH}_C$.

Proposition 3.7 holds since we may consider, by brute-force, all 2^k possible sets of query answers.

We observe that if there exists any $k > 1$ such that one can recognize characteristic vectors of length k with one query then one can do the same for every length.

Proposition 3.8 Let C be an interesting class. If there is a $k > 1$ with $CV_C(k) = 1$, then for all $i \geq 1$ it holds that $CV_C(i) = 1$.

Proof: $CV_C(j) = 1$ for $j \leq k$, via repeating values. We proceed by induction. Suppose $CV_C(j) = 1$ for some $j \geq k$. Let M be the machine that checks characteristic vectors of length j for a language $L \in C$ with one query to an oracle $L' \in C$. Now vectors $(x_1, \ldots, x_{j+1}, b_1, \ldots, b_{j+1})$ of L can be checked the following way: Run M on the vector $v = (x_1, \ldots, x_j, b_1, \ldots, b_j)$ pursuing both possible outcomes of the query but without actually asking the query to the oracle. If both outcomes of the query $q(v)$ are different (otherwise the case is trivial), we know that v is a characteristic vector if and only if M accepts on the 'yes' path and $q(v) \in L'$ or M accepts on the 'no' path and $q(v) \notin L'$. Since C is closed under disjoint union there is a machine M' checking membership in $V_2^{L \oplus L'}$ with one query to a language $L'' \in C$. We run M' on $(1q(v), 0x_{j+1}, 1, b_{j+1})$ if M accepts v on the 'yes' path, and on $(1q(v), 0x_{j+1}, 0, b_{j+1})$ if it accepts v on the 'no' path. Thus one query to L'' suffices to check vectors of length $j + 1$, and the claim follows. ∎

The following theorem explores the consequences for these boolean hierarchies of wee characteristic vector costs of their basic classes.

Theorem 3.9 Let C be an interesting class that contains P properly: $(\exists k > 1)[CV_C(k) = 1]$ if and only if $\mathrm{BH}_C = \mathrm{P}^{C[1]}$.

Proof:
If $\mathrm{BH}_C = \mathrm{P}^{C[1]}$ then by Proposition 3.7 we get $V_k^L \in P^{C[1]}$, which means that we can check characteristic vectors of length k with one query to a language in C.
Now assume the left-hand side of the equivalence holds.
'$\mathrm{P}^{C[1]} \subseteq \mathrm{BH}_C$': By Proposition 3.7.
'$\mathrm{BH}_C \subseteq \mathrm{P}^{C[1]}$': Fix $S \in \mathrm{BH}_C$. We will show that $S \in \mathrm{P}^{C[1]}$. By definition there is a k such that $S = S_1 - (S_2 - (\cdots S_{k-1} - S_k) \cdots)$, with $S_i \in C$ for all $i \leq k$. This gives rise to a boolean formula with

$$x \in S \iff S_1(x) \wedge \neg(S_2(x) \wedge \neg(\cdots \neg(S_{k-1}(x) \wedge \neg S_k(x)) \cdots)),$$

where $S_i(x)$ stands for $x \in S_i$. The conjunctive normal form of this formula is of the form $cl_1 \wedge cl_2 \wedge \cdots \wedge cl_j$, where j and the number of literals in each clause cl_i are constant, i.e., they depend only on S but not on the individual input string x. Thus it is clear that there is a machine that evaluates each clause $cl_i = (S_{i_1}(x), \ldots, S_{i_s}(x), \neg S_{i_{s+1}}(x), \ldots, \neg S_{i_r}(x))$

with one query to a C language: Since C is interesting $S_1 \oplus \cdots \oplus S_k \in C$ and by Proposition 3.8 there is machine M that checks $S_1 \oplus \cdots \oplus S_k$-vectors of the maximal length of all clauses cl_i with one query to a language $L' \in C$. We run M on the vector $v_i = (\langle string(i_1), S_{i_1}(x)\rangle, \ldots, \langle string(i_r), S_{i_r}(x)\rangle, 0, \ldots, 0, 1, \ldots, 1)$ with 0 repeated s times and 1 repeated $r - s$ times—exploiting the fact that v_i is a characteristic vector of $S_1 \oplus \cdots \oplus S_k$ if and only if the value of the clause cl_i is false.

Without loss of generality we can assume that for any vector v_i this machine queries $q(v_i)$, accepts on one outcome of the query, and rejects on the other one. Thus the value of cl_i is true if and only if either M accepts on the 'yes' path and $q(v_i) \in L'$, or M accepts on the 'no' path and $q(v_i) \notin L'$. In polynomial time we can precompute which queries M asks for the different clauses, and which of the 'yes' or 'no' paths it accepts on.

This gives rise to the following procedure:

On input x with $F(x) = cl_1 \wedge cl_2 \wedge \cdots \wedge cl_j$ being the corresponding CNF formula (which is fixed for each S).

- For each clause cl_i compute the vector v_i and the query $q(v_i)$ the machine M asks. If M accepts v_i on the 'yes' path then $b_i := 0$ else $b_i := 1$.

- Check the vector $(q(v_1), q(v_2), \ldots, q(v_j), b_1, \ldots, b_j)$.

By assumption the last part is possible by running a polynomial-time oracle machine, with an oracle $L \in C$ that is queried only once. From the construction it follows that $x \in S$ if and only if $F(x) = cl_1 \wedge cl_2 \wedge \cdots \wedge cl_j$ is true if and only if for all $i \leq j$ it holds that $\chi_L(q(v_i)) = b_i$.

∎

It follows that for classes with wee characteristic vector cost it is possible for deterministic oracle machines to query the oracle only once, instead of a constant number of times, without loss of power.

Corollary 3.10 Let C be an interesting class.

1. $(\exists k > 1)[CV_C(k) = 1]$ if and only if $\mathrm{P}^{C[i]} = \mathrm{P}^{C[1]}$ for all $i \geq 1$.

2. $(\exists k > 1)[CV_C(k) = 1]$ if and only if every language that is polynomial-time bounded-truth-table reducible to a set in C is indeed polynomial-time one-truth-table reducible to a set in C.

By a result of Papadimitriou and Zachos ([PZ83], see also [KSW87,AG88,CGH$^+$88]), for each i, $\mathrm{P}^{\mathrm{NP}[i]}$ is contained in the boolean hierarchy. This also holds for arbitrary interesting classes (Proposition 3.7), and the same applies for bounded-truth-table degrees of interesting classes. Thus Corollary 3.10 follows from Theorem 3.9.

Corollary 3.10 contrasts with a result of Chang and Kadin ([CK90a], see also [Kad88]) and Beigel [Bei88]: They gave a criterion that is equivalent to the collapse of the boolean hierarchy of a class to its closure under polynomial-time many-one reductions. For $C = \mathrm{NP}$, Corollary 3.13 will provide a criterion that is equivalent to the coincidence of the closure of NP under polynomial-time truth-table reductions with its closure under one-truth-table reductions.

Theorem 3.9 immediately yields many results about well-known interesting complexity classes (in the technical sense defined previously). We will illustrate and extend them in the case of NP and some classes that are in its boolean hierarchy. The boolean hierarchy over NP extends the study of sets that are the difference of two NP languages [PZ83,PY84], and has been extensively investigated (see [CGH$^+$88,CGH$^+$89]). For NP itself we obtain Corollary 3.11, which follows from the fact that NP is closed under union and intersection, Proposition 3.3, Theorem 3.9, and the result of Chang and Kadin result that a collapse of the boolean hierarchy implies one of the polynomial hierarchy ([CK90a], see also [Kad88]).

Corollary 3.11

1. $CV_{\mathrm{NP}} \leq 2$. Indeed, the two queries can be parallel rather than sequential.
2. $(\exists k > 1)[CV_{\mathrm{NP}}(k) = 1]$ if and only if $\mathrm{BH}_{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}[1]}$.
3. If there is a $k > 1$ such that $CV_{\mathrm{NP}}(k) = 1$, then the polynomial hierarchy collapses.

Up to here we studied the consequences of being able to recognize characteristic vectors of a fixed constant length with one query to an oracle of the same class. The following theorem explores the consequences of the stronger assumption that characteristic vectors of variable length are uniformly checkable with one query.

Theorem 3.12 $CV_{\mathrm{NP}} = 1$ if and only if $\mathrm{P}^{\mathrm{NP}[\log n]} = \mathrm{P}^{\mathrm{NP}[1]}$.

Proof: The idea of the proof (left to right, the other direction is immediate since $V_\omega^{SAT} \in \mathrm{P}^{SAT[2]}$) is similar to that of Theorem 3.9: Run the machine M, which queries its oracle $L \in \mathrm{NP}$ at most $\log n$ times, along every possible path arising from all answers that might be given by the oracle (without querying the oracle). Each accepting path corresponds to a candidate for a characteristic vector of at most $\log n$ components. M accepts the input if and only if one of them is a characteristic vector. By assumption we can recognize these vectors with a machine M' with one query to an NP language.

Now we run M' on each of those vectors without querying the oracle. M' asks a query q for such a vector, and accepts it on the 'yes' path and rejects on the 'no' path or vice versa (otherwise the case is trivial). The vector is a characteristic vector if and only if q is in the oracle and the 'yes' path of M' accepts or q is not in the oracle if the 'no' path accepts. Thus by checking the outcomes of M' on the different input vectors we obtain a vector of new queries and new answers, knowing that M accepts if and only if at least one of the answers is correct. Thus it follows that M accepts if and only if the vector consisting of the same queries, and having exchanged 'yes' and 'no' answers, is *not* a characteristic vector, which again by assumption we can check with one query. ∎

The fact that the class of languages polynomial-time truth-table reducible to an NP language equals $\mathrm{P}^{\mathrm{NP}[\log n]}$ [Hem89,Wag90] yields the following corollary, which is interesting when compared with Corollary 3.10, part 2.

Corollary 3.13 $CV_{\mathrm{NP}} = 1$ if and only if all languages that are polynomial-time truth-table reducible to an NP language are indeed polynomial-time one-truth-table reducible to an NP language.

Indeed, combining Theorem 3.12, the discussion following Proposition 3.3 (i.e., that $CV_{\mathrm{NP}} = CV_{\mathrm{NP}}(2)$), and Corollary 3.11 part 2, one arrives at Corollary 3.14 below, recently proven (by other means) by Chang and Kadin.

Corollary 3.14 [CK90b] $\mathrm{BH}_{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}[1]}$ if and only if $\mathrm{P}^{\mathrm{NP}[\log n]} = \mathrm{P}^{\mathrm{NP}[1]}$.

Now let's turn our attention to some classes that are located within the boolean hierarchy over NP. Its levels are neither known nor believed to be closed under complement, intersection, or union. In fact, by a result of Chang and Kadin, [CK90a], for any $k \geq 3$ the closure of $\mathrm{BH}_{\mathrm{NP}}(k)$ under one of those operations implies its collapse on the same level.

What are the consequences if vectors of languages of some level can be checked with one query? The following two results display a connection between vector checking and the more general structure of the boolean hierarchy. (Note that, since the collapse of the boolean hierarchy implies the collapse of the polynomial hierarchy, the first part is related to the structure of the polynomial hierarchy.)

Theorem 3.15 1. For all $k > 1$ it holds that: $(\exists i > 1)[CV_{\mathrm{BH}_{\mathrm{NP}}(2^k-1)}(i) = 1] \Longrightarrow \mathrm{BH}_{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}[k]}$.

2. Let COMP be any set that is $\leq_m^p$-complete for BH(2) (see, e.g., [PY84,CM87]) and let $\mathcal{C}$ be an arbitrary class of sets. $(\forall i)[V_i^{\mathrm{COMP}} \in \mathrm{P}^{\mathcal{C}[1]}] \iff \mathrm{BH} \subseteq \mathrm{P}^{\mathcal{C}[1]}$.

Proof: For the first part, let $L \in \mathrm{BH}_{\mathrm{NP}}(2^k)$ and $L = L_1 - L_2$ with $L_1 \in \mathrm{NP}$ and $L_2 \in \mathrm{BH}_{\mathrm{NP}}(2^k - 1)$. Furthermore let $\hat{L}$ be any $\mathrm{BH}_{\mathrm{NP}}(2^k - 1)$-complete problem (in [CGH$^+$88] it was shown that such problems exist) and f_1, f_2 be the corresponding reductions from L_1 and L_2 to it. Then $x \in L$ if and only if $(f_1(x), f_2(x), 1, 0)$ is a characteristic vector of $\hat{L}$, which can be checked with one query to $\hat{L}$ itself, since $\hat{L}$ is complete in $\mathrm{BH}_{\mathrm{NP}}(2^k - 1)$. Using the result of [AG88] that $\mathrm{BH}_{\mathrm{NP}}(2^k - 1) \subseteq \mathrm{P}^{\mathrm{NP}[k]}$ and thus $\mathrm{P}^{\mathrm{BH}_{\mathrm{NP}}(2^k-1)[1]} \subseteq \mathrm{P}^{\mathrm{NP}[k]}$ it follows that $\mathrm{BH}_{\mathrm{NP}}(2^k) \subseteq \mathrm{P}^{\mathrm{NP}[k]}$. Since $\mathrm{BH}_{\mathrm{NP}}(2^k) \supseteq \mathrm{P}^{\mathrm{NP}[k]}$ it follows that $\mathrm{BH}_{\mathrm{NP}}(2^k) = \mathrm{P}^{\mathrm{NP}[k]}$ and the boolean hierarchy collapses to that class, since in that case $\mathrm{BH}_{\mathrm{NP}}(2^k)$ is closed under complement.

The second part is immediate from the fact all sets of the boolean hierarchy can be expressed as unions of BH(2) sets [CGH$^+$88]. ∎

We note that for almost all oracles, characteristic vectors of any class of the boolean hierarchy cannot be checked with one query to an oracle of the same power. In particular, it follows that with respect to a random oracle the bound of part 1 of Corollary 3.11 is optimal.

Corollary 3.16 With probability one relative to a random oracle A,

$$(\forall k \geq 1)[CV_{\mathrm{BH}_{\mathrm{NP}^A}(k)} \neq 1].$$

This follows from the facts that the proof of Theorem 3.15 relativizes and the boolean hierarchy over NP is infinite with respect to a random oracle [Cai89].

Another class located in the boolean hierarchy over NP is US (Unique Solutions): A language belongs to US if it is accepted in polynomial time by a nondeterministic machine that, by definition, accepts if and only if it has exactly one accepting path [BG82,GW87]. US is known to be closed under intersection, but does not seem to be closed under union or complement. For this class Theorem 3.9 yields the following corollary (recall Definition 3.4):

Corollary 3.17

1. US is cv-wee under complement if and only if $\mathrm{BH}_{\mathrm{NP}} = \mathrm{P}^{\mathrm{US}[1]}$.

2. If US is cv-wee under complement then the polynomial hierarchy collapses.

Proof: First note that $\mathrm{BH}_{\mathrm{NP}} = \mathrm{BH}_{\mathrm{US}}$, since US contains coNP [BG82], so $\mathrm{BH}_{\mathrm{NP}} \subseteq \mathrm{BH}_{\mathrm{US}}$, and thus, since $\mathrm{US} \subseteq \mathrm{BH}_{\mathrm{NP}}(2)$ the two hierarchies are equal. Part 1 follows from this observation and the fact that US is closed under intersection. Part 2 is a consequence of the result of Chang and Kadin that a collapse of the boolean hierarchy over NP implies a collapse of the polynomial hierarchy ([CK90a], see also [Kad88]). ∎

Finally we observe that wee characteristic vector cost for a class within NP implies that it is low. Low classes, intuitively, are sets carrying far less information than NP-complete sets. In particular, a set $L \in \mathrm{NP}$ is in $\widehat{low}_3$ if $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}^L}} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ ($\widehat{low}_3$ was first defined in [KS85] as a generalization of the work of [Sch83]). Similarly, a class $C \subseteq \mathrm{NP}$ is said to be $\widehat{low}_3$ if every $L \in C$ is in $\widehat{low}_3$. Following Schöning's seminal paper on the low hierarchy, a number of papers have explored and refined its structure [KS85,BBS86,K$\ddot{8}$8], culminating in the essentially optimal placement of classes within the low hierarchy [AH]. The following result shows that classes with wee characteristic vector cost are simple in the sense of lowness.

Theorem 3.18 All classes $C \subseteq \mathrm{NP}$ with wee characteristic vector cost are $\widehat{low}_3$.

Proof: A sufficient condition for C to be $\widehat{low}_3$ is that for any $L \in C$ there is an $L' \in C$ such that $\{(x,y) | x \in L \wedge y \notin L\} \leq_m^p \{(x,y) | x \notin L' \vee y \in L'\}$ [Cha89]. For $L \in C$ with wee characteristic vector cost it holds that $\{(x,y) | x \in L \wedge y \notin L\} \in \mathrm{P}^{L'[1]}$ for some $L' \in C$. As in the proof of Theorem 3.9, one can easily show that $\mathrm{P}^{L'[1]} \leq_m^p \{(x,y) | x \notin L' \vee y \in L'\}$, and thus the condition is fulfilled. ∎

Acknowledgments
We wish to thank Gerd Wechsung, Garry Benson, Gerhard Buntrock, Katerina Chronaki, Martin Farach, Sudhir Jha, Kathleen Romanik, Dirk Siefkes, and Hubert Wagener for critically reading earlier versions of the paper, Richard Beigel, James Kadin, Klaus-Jörn Lange, Leen Torenvliet and Paul Vitanyi for helpful conversations, Gerd Wechsung for hosting the workshop at which this collaboration was started, and Richard Chang for pointing out the modified proof of Theorem 2.5. We are particularly grateful to Peter van Emde Boas and two anonymous referees for many helpful suggestions.

References

[AG88] A. Amir and W. Gasarch. Polynomial terse sets. *Information and Computation*, 77:37–56, 1988.

[AH] E. Allender and L. Hemachandra. Lower bounds for the low hierarchy. *Journal of the ACM*. To appear.

[AR88] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.

[Bār68] J. Bārzdiņš. Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. *Soviet Math. Dokl.*, 9:1251–1254, 1968.

[BBJ$^+$] A. Bertoni, D. Bruschi, D. Joseph, M. Sitharam, and P. Young. Generalized boolean hierarchies and boolean hierarchy over RP. Technical Report 809, University of Wisc.–Madison Department of Computer Science, 1989. Preliminary version appears in *Proceedings Fundamentals of Computation Theory*, Springer-Verlag *Lecture Notes in Computer Science #380*, pp. 35–46.

[BBS86] J. Balcázar, R. Book, and U. Schöning. Sparse sets, lowness and highness. *SIAM Journal on Computing*, 15(3):739–746, 1986.

[Bei] R. Beigel. Bounded queries to SAT and the boolean hierarchy. *Theoretical Computer Science*. To appear.

[Bei87] R. Beigel. A structural theorem that depends quantitatively on the complexity of SAT. In *Proceedings of the 2nd Structure in Complexity Theory Conference*, pages 28–32. IEEE Computer Society Press, June 1987.

[Bei88] R. Beigel. NP-hard sets are P-superterse unless R=NP. Technical Report 88-04, Johns Hopkins Department of Computer Science, August 1988.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55:80–88, 1982.

[BGGO] R. Beigel, W. Gasarch, J. Gill, and J. Owings. Terse, superterse, and verbose sets. *Information and Computation*. To appear.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[Cai89] J. Cai. With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy. *Journal of Computer and System Sciences*, 38(1):68–85, 1989.

[CGH$^+$88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[CGH$^+$89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

[CH90] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23:95–106, 1990.

[Cha89] R. Chang. On the structure of bounded queries to arbitrary NP sets. In *Proceedings of the 4th Structure in Complexity Theory Conference*, pages 250–258. IEEE Computer Science Press, June 1989.

[CK90a] R. Chang and J. Kadin. The boolean hierarchy and the polynomial hierarchy: A closer connection. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 169–178. IEEE Computer Society Press, July 1990.

[CK90b] R. Chang and J. Kadin. On computing boolean connectives of characteristic functions. Technical Report TR 90-1118, Department of Computer Science, Cornell University, Ithaca, NY, May 1990.

[CM87] J. Cai and G. Meyer. Graph minimal uncolorability is $\mathrm{D^P}$-complete. *SIAM Journal on Computing*, 16(2), 1987.

[EHK81] R. Epstein, R. Haas, and R. Kramer. Hierarchies of sets and degrees below $0'$. In *Logic Year 1979-80, the University of Connecticut, Lecture Notes in Mathematics #859*, pages 32–47. Springer Verlag, Berlin, 1981.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[GP86] L. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theoretical Computer Science*, 43:43–58, 1986.

[GW87] T. Gundermann and G. Wechsung. Counting classes with finite acceptance types. *Computers and Artificial Intelligence*, 6(5):395–409, 1987.

[Hau14] F. Hausdorff. *Grundzüge der Mengenlehre*. Leipzig, 1914.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[IT89] R. Impagliazzo and G. Tardos. Decision versus search problems in super-polynomial time. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 222–227. IEEE Computer Society Press, October/November 1989.

[Joc68] C. Jockusch. Semirecursive sets and positive reducibility. *Transactions of the AMS*, 131(2):420–436, 1968.

[Joc79] C. Jockusch. Recursion theory: Its generalizations and applications. In *Proccedings of the Logic Colloquium, Leeds*, pages 140–157. Cambridge University Press, 1979.

[K8̈8] J. Kämper. Non-uniform proof systems: a new framework to describe non-uniform and probabilistic complexity classes. In *Proceeings of the 8th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 193–210. Springer-Verlag *Lecture Notes in Computer Science #338*, December 1988.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988.

[KS85] K. Ko and U. Schöning. On circuit-size complexity and the low hierarchy in NP. *SIAM Journal on Computing*, 14(1):41–51, 1985.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *R.A.I.R.O. Informatique théorique et Applications*, 21:419–435, 1987.

[PY84] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings 6th GI Conference on Theoretical Computer Science*, pages 269–276. Springer-Verlag *Lecture Notes in Computer Science #145*, 1983.

[Rog67] H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.

[Rub88] R. Rubinstein. *Structural Complexity Classes of Sparse Sets: Intractability, Data Compression and Printability.* PhD thesis, Northeastern University, Boston, MA, August 1988.

[Rub90] R. Rubinstein. Relativizations of the P-printable sets and the sets with small generalized Kolmogorov complexity. Technical Report WPI-CS-TR-90-3, Worcester Polytechnic Institute, Worcester, MA, March 1990.

[Sch83] U. Schöning. A low and a high hierarchy in NP. *Journal of Computer and System Sciences.*, 27:14–28, 1983.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[Wag88] K. Wagner. Bounded query computation. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 260–277. IEEE Computer Society Press, June 1988.

[Wag90] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.