数理解析研究所講究録
第833巻 1993年 31-41

On Using Oracles That Compute Values

関数のオラクルを用いた計算

Stephen Fenner(南メイン大)* Steve Homer(ボストン大)†
Mitsunori Ogiwara(荻原光徳 電気通信大)‡ Alan L. Selman(ニューヨーク州立大)§

Abstract

This paper focuses on complexity classes of partial functions that are computed in polynomial time with oracles in NPMV, the class of all multivalued partial functions that are computable nondeterministically in polynomial time. Concerning deterministic polynomial-time reducibilities, it is shown that

1. A multivalued partial function is polynomial-time computable with k adaptive queries to NPMV if and only if it is polynomial-time computable via $2^k - 1$ nonadaptive queries to NPMV.
2. A characteristic function is polynomial-time computable with k adaptive queries to NPMV if and only if it is polynomial-time computable with k adaptive queries to NP.
3. Unless the Boolean hierarchy collapses, k adaptive (nonadaptive) queries to NPMV is different than $k+1$ adaptive (nonadaptive) queries to NPMV for every k.

Nondeterministic reducibilities, lowness and the difference hierarchy over NPMV are also studied. The difference hierarchy for partial functions does not collapse unless the Boolean hierarchy collapses, but, surprisingly, the levels of the difference and bounded query hierarchies do not interleave (as is the case for sets) unless the polynomial hierarchy collapses.

*Dept. of Computer Science, Portland, ME 04103. Research partially supported by the National Science Foundation under grant no. CCR-9209833.

†Dept. of Computer Science, Boston, MA 01003. Research partially supported by the National Science Foundation under grant no. CCR-9103055

‡Dept. of Computer Science, Chofu-shi, Tokyo 182, Japan. Research partially supported by the National Science Foundation under grant no. CCR-9002292.

§Dept. of Computer Science, Buffalo, NY 14260. Research partially supported by the National Science Foundation under grant no. CCR-9002292.

1 Introduction

In this paper we study classes of partial functions that can be computed in polynomial time with oracles in NPMV and NPSV; namely, we study the classes $\mathrm{PF}^{\mathrm{NPMV}}$ and $\mathrm{PF}^{\mathrm{NPSV}}$.

NPMV is the set of all partial multivalued functions that are computed nondeterministically in polynomial time, and NPSV is the set of all partial functions in this class that are single-valued. NPMV captures the complexity of computing witnesses to problems in NP. For example, let *sat* denote the partial function defined by $sat(x)$ maps to a value y if and only if x encodes a formula of propositional logic and y encodes a satisfying assignment of x. Then, *sat* belongs to NPMV, and the domain of *sat* (i.e., the set of all words x for which the output of $sat(x)$ is non-empty) is the NP-complete satisfiability problem, SAT. Also, NPMV captures the complexity of inverting polynomial time honest functions. To wit, the inverse of every polynomial time honest function belongs to NPMV, and the inverse of every one-one polynomial time honest function belongs to NPSV.

The class of partial functions with oracles in NP, namely, $\mathrm{PF}^{\mathrm{NP}}$ has been well-studied [Kre88], as have been the corresponding class of partial functions that can be computed nonadaptively with oracles in NP, viz. $\mathrm{PF}^{\mathrm{NP}}_{tt}$ [Sel92], and the classes of partial functions that are obtained by limiting the number of queries to some value $k \geq 1$, namely, $\mathrm{PF}^{\mathrm{NP}[k]}$ and $\mathrm{PF}^{\mathrm{NP}[k]}_{tt}$ [Bei91]. A rich body of results is known about these classes.

Here we raise the question, "what is the difference between computing with an oracle in NPMV versus an oracle in NP." The answer is not obvious. If the partial function *sat* is provided as an oracle to some polynomial-time computation M, then on a query x, where x encodes a satisfiable formula of propositional logic, the oracle will return some satisfying assignment y. However, if the oracle to M is the NP-compete set SAT, then to this query x, the oracle will only return a Boolean value "yes." On the other hand, by the well-known self-reducibility of SAT, M could compute y for itself by judicious application of several adaptive queries to SAT. Indeed Theorem 1.1 states that unbounded access to an oracle in NPMV is no more powerful than such an access to an oracle in NP. However, in Section 3 we will see that the situation for bounded query classes is much more subtle. In general, function oracles cannot be replaced by set oracles – but set oracles are still useful. We will show that every partial function in $\mathrm{PF}^{\mathrm{NPMV}[k]}$ can be computed by a partial function of the form $f \circ g$, where f is in NPMV and g belongs to $\mathrm{PF}^{\mathrm{NP}[k]}$. Moreover, most surprisingly, the relationship between access to an oracle in NPMV and access to an oracle in NP is tight regarding set recognition; that is, $\mathrm{P}^{\mathrm{NPMV}[k]} = \mathrm{P}^{\mathrm{NP}[k]}$. This means that when we are computing characteristic functions, k bounded queries to an oracle in NPMV give no more information than the same number of queries to an oracle in NP.

We will show that the levels of the nonadaptive and adaptive bounded query hierarchies interleave (for example, k adaptive queries to a partial function in NPMV is equivalent to $2^k - 1$ nonadaptive queries to a partial function in NPMV), and we will show that these bounded query hierarchies collapse only if the Boolean hierarchy collapses.

In Section 4 we study nondeterministic polynomial time reductions to partial functions in NPMV. Unlike the case for deterministic functions, we will see that just one query to an NP oracle can substitute for an unbounded number of queries to any partial function in NPMV. The hierarchy that is formed by iteratively applying NP reductions is an analogue of the polynomial hierarchy, and we will show that this hierarchy collapses if and only if the polynomial hierarchy collapses.

In Section 5 we will study the difference hierarchy over NPMV. We define $f - g$ to be a partial function that maps x to y if and only if f maps x to y and g does *not* map x to y, and we define $\mathrm{NPMV}(k) = \{f_1 - (f_2 - (\cdots - f_k)) : f_1, \cdots, f_k \in \mathrm{NPMV}\}$. Since the properties of the bounded query hierarchies over NPMV are largely similar to those over NP, one might hope that the same thing happens here – that the query hierarchy over NPMV and the query hierarchy over NP are similar. However, the contour of this hierarchy is, to our astonishment, totally different than its analogy for NP. Although $\mathrm{BH} = \bigcup_k \mathrm{NP}(k) \subseteq \mathrm{P}^{\mathrm{NP}}$, with no assumption, we will show that NPMV(2) is included in $\mathrm{PF}^{\mathrm{NPMV}}$ if and only if $\mathrm{PH} = \Delta_2^{\mathrm{P}}$. Also, in this section we will introduce the notion of NPMV-lowness, and we will give a complete characterization of NPMV-lowness.

Consideration of reduction classes with oracles in NPSV, to be studied in Section 6, is motivated in part by a desire to understand how difficult it is to compute satisfying assignments for satisfiable formulas. The following technical notions will help to make this clear. Given partial multivalued functions f and g, define g to be a *refinement* of f if $dom(g) = dom(f)$ and for all $x \in dom(g)$ and all y, if y is a value of $g(x)$, then y is a value of $f(x)$. Let $\mathcal{F}$ and $\mathcal{G}$ be classes of partial multivalued functions. Purely as a convention, if f is a partial multivalued function, we define $f \in_c \mathcal{G}$ if $\mathcal{G}$ contains a refinement g of f, and we define $\mathcal{F} \subseteq_c \mathcal{G}$ if for every $f \in \mathcal{F}$, $f \in_c \mathcal{G}$. Let PF denote the class of partial functions that are computable deterministically in polynomial time. The assertion "$\mathrm{NPMV} \subseteq_c \mathrm{PF}$" would mean that every partial multivalued function in NPMV has a refinement that can be computed efficiently by some deterministic polynomial time transducer. It is well-known that $sat \in_c \mathrm{PF}$ if and only if $\mathrm{NPMV} \subseteq_c \mathrm{PF}$ if and only if $\mathrm{P} = \mathrm{NP}$ [Sel92]. Thus, one does not expect that $sat \in_c \mathrm{PF}$. Is sat computable in some larger single-valued class of partial functions? It is shown in [Sel92] that $\mathrm{PF} \subseteq \mathrm{NPSV} \subseteq \mathrm{PF}_{tt}^{\mathrm{NP}}$, and it is an open question whether $sat \in_c \mathrm{NPSV}$ or whether $sat \in_c \mathrm{PF}_{tt}^{\mathrm{NP}}$. We will consider classes of the form $\mathrm{PF}^{\mathrm{NPSV}[k]}$ and $\mathrm{PF}_{tt}^{\mathrm{NPSV}[k]}$, where $k \geq 1$, and we will show that the adaptive and the

nonadaptive classes form proper hierarchies unless the Boolean hierarchy collapses. Thus, these classes form a finer classification in which to study the central question of whether *sat* has a refinement in some interesting class of single-valued partial functions.

Finally, we note in passing that the complexity theory of decision problems, i.e., of sets, is extremely well developed. Although the computational problems in which we are most interested are naturally thought of as partial multivalued functions, the structural theory to support classification of these problems has been slight. By introducing several natural hierarchies of complexity classes of partial multivalued functions, with strong evidence supporting these claims, we intend this paper to make significant steps in correcting this situation.

2 Preliminaries

We fix Σ to be the finite alphabet $\{0,1\}$. Let $f : \Sigma^* \pfun \Sigma^*$ be a partial multivalued function. We write $f(x) \mapsto y$, if y is a value of f on input string x. Define $graph(f) = \{\langle x,y\rangle \mid f(x) \mapsto y\}$, $dom(f) = \{x \mid \exists y(f(x) \mapsto y)\}$, and $range(f) = \{y \mid \exists x(f(x) \mapsto y)\}$. We will say that f is undefined at x if $x \notin dom(f)$.

A transducer T is a nondeterministic Turing machine with a read-only input tape, a write-only output tape, and accepting states in the usual manner. T computes a value y on an input string x if there is an accepting computation of T on x for which y is the final content of T's output tape. (In this case, we will write $T(x) \mapsto y$.) Such transducers compute partial, multivalued functions. (As transducers do not typically accept all input strings, when we write "function", "partial function" is always intended. If a function f is total, it will always be explicitly noted.)

- NPMV is the set of all partial, multivalued functions computed by nondeterministic polynomial time-bounded transducers;
- NPSV is the set of all $f \in$ NPMV that are single-valued;
- PF is the set of all partial functions computed by deterministic polynomial time-bounded transducers.

$\mathrm{PF}^{\mathrm{NP}}$ is the class of functions computed in polynomial time with oracles in NP. $\mathrm{PF}^{\mathrm{NP}}_{tt}$ is the class of functions that can be computed nonadaptively with oracles in NP; that is, a partial function f is in $\mathrm{PF}^{\mathrm{NP}}_{tt}$ if there is an oracle Turing machine transducer T such that $f \in \mathrm{PF}^{\mathrm{NP}}$ via T with an oracle L in NP and a polynomial time computable function

$f : \{0,1\}^* \mapsto (c\{0,1\}^*)^*$ such that, for each input x to T, T only makes queries to L from the list $f(x)$.

Now we describe oracle Turing machines with oracles that compute partial functions. For the moment, we assume that the oracle is a single-valued partial function. Let $\perp$ be a symbol not belonging to the finite alphabet Σ. In order for M to access a partial function oracle, M contains a write-only input oracle tape, a separate read-only output tape, and a special oracle call state q. When M enters state q, if the string currently on the oracle input tape belongs to the domain of the oracle partial function, then the result of applying the oracle appears on the oracle output tape, and if the string currently on the oracle input tape does not belong to the domain of the oracle partial function, then the symbol $\perp$ appears on the oracle output tape. Thus, if the oracle is some partial function g, given an input x to the oracle, the oracle, if called, returns a value $g(x)$ if one exists, and returns $\perp$ otherwise. The oracle may not provide its own input, so that any change to the oracle input must be made by M. (It is possible that M may read only a portion of the oracle's output if the oracle's output is too long to read with the resources of M.)

If g is a single-valued partial function and M is a deterministic oracle transducer as just described, then we let $M[g]$ denote the single-valued partial function computed by M with oracle g.

Definition 1 *Let f and g be multivalued partial functions. f* is Turing reducible to g in polynomial time, *$f \leq_{\mathrm{T}}^{\mathrm{P}} g$, if for some deterministic oracle transducer M, for every single-valued refinement g' of g, $M[g']$ is a single-valued refinement of f.*[1]

Let $\mathcal{F}$ be a class of partial multivalued functions. $\mathrm{PF}^{\mathcal{F}}$ denotes the class of partial multivalued functions f that are $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to some $g \in \mathcal{F}$. $\mathrm{PF}^{\mathcal{F}[k]}$ (respectively, $\mathrm{PF}^{\mathcal{F}[\log]}$) denotes the class of partial multivalued functions f that are $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to some $g \in \mathcal{F}$ via a machine that, on input x, makes k adaptive queries (respectively, $\mathcal{O}(\log |x|)$ adaptive queries) to its oracle.

$\mathrm{PF}_{tt}^{\mathcal{F}}$ denotes the class of partial multivalued functions f that are $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to some $g \in \mathcal{F}$ via an oracle Turing machine transducer that queries its oracle nonadaptively. $\mathrm{PF}_{tt}^{\mathcal{F}[k]}$

[1] A notion of polynomial-time Turing reducibility between partial functions is defined in [Sel92]. It is important to note that the definition given here is *different* than the one in [Sel92]. Here the oracle "knows" when a query is not in its domain. In the earlier definition, this is not the case. The authors recommend that the reducibility defined in [Sel92] should in the future be denoted as $\leq_{\mathrm{T}}^{\mathrm{PP}}$, which is the common notation for reductions between promise problems. We make this recommendation because conceptually and technically this reducibility between functions is equivalent to a promise problem reduction. Also, we note that the reducibility defined in [Sel92] is not useful for our purposes here. In particular, it is easy to see that iterating reductions between functions in NPMV does not gain anything new unless the oracle is endowed with the ability to know its domain.

denotes the class of partial multivalued functions f that are $\leq^{\mathrm{P}}_{\mathrm{T}}$-reducible to some $g \in \mathcal{F}$ via a machine that makes k nonadaptive queries to its oracle.

$\mathrm{P}^{\mathcal{F}}$, $\mathrm{P}^{\mathcal{F}[k]}$, $\mathrm{P}^{\mathcal{F}[\log]}$, $\mathrm{P}^{\mathcal{F}}_{tt}$ and $\mathrm{P}^{\mathcal{F}[k]}_{tt}$, respectively, denote the classes of all characteristic functions contained in $\mathrm{PF}^{\mathcal{F}}$, $\mathrm{PF}^{\mathcal{F}[k]}$, $\mathrm{PF}^{\mathcal{F}[\log]}$, $\mathrm{PF}^{\mathcal{F}}_{tt}$ and $\mathrm{PF}^{\mathcal{F}[k]}_{tt}$.

For a class of sets $\mathcal{C}$, we may say that $\mathrm{PF}^{\mathcal{C}}$ denotes the class of partial single-valued functions that are $\leq^{\mathrm{P}}_{\mathrm{T}}$-reducible to the characteristic function of some set in $\mathcal{C}$. (Note that functions in $\mathrm{PF}^{\mathcal{C}}$ are single-valued.) $\mathrm{PF}^{\mathcal{C}[k]}$, $\mathrm{PF}^{\mathcal{C}[\log]}$, $\mathrm{PF}^{\mathcal{C}}_{tt}$, $\mathrm{PF}^{\mathcal{C}[k]}_{tt}$, $\mathrm{P}^{\mathcal{C}}$, $\mathrm{P}^{\mathcal{C}[k]}$, $\mathrm{P}^{\mathcal{C}[\log]}$, $\mathrm{P}^{\mathcal{C}}_{tt}$, and $\mathrm{P}^{\mathcal{C}[k]}_{tt}$ are defined similarly.

Obviously $\mathrm{PF}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NPMV}}$. Conversely, for a function $f \in \mathrm{NPMV}$, define f' to be a function such that $f'(x) = \min\{y : f(x) \mapsto y\}$. f' is a single-valued refinement of f and in $\mathrm{PF}^{\mathrm{NP}}$. Therefore, the following theorem holds.

Theorem 1 $\mathrm{PF}^{\mathrm{NPMV}} \subseteq_c \mathrm{PF}^{\mathrm{NP}}$. *That is, every function in* $\mathrm{PF}^{\mathrm{NPMV}}$ *has a single-valued refinement in* $\mathrm{PF}^{\mathrm{NP}}$.

Consider the function *maxclique* that on input a graph G outputs a clique of maximum size, if G has a clique. Then, *maxclique* is a multivalued partial function that belongs to $\mathrm{PF}^{\mathrm{NPMV}}$. Similarly, the following function $maxTsat$ is a multivalued partial function that belongs to $\mathrm{PF}^{\mathrm{NPMV}}$.

$maxTsat(x) \mapsto y$, if y is a satisfying assignment of x with the maximum number of *true*'s.

Let f be a function that maps a pair (x, n) to y if and only if y is a satisfying assignment of x with n *true*'s. Since the number of variables in a formula is bounded by its length, it holds that $maxTsat(x) = f(x, n_x)$, where n_x is the largest $n, 1 \leq n \leq |x|$ such that $(x, n) \in dom(f)$. This implies that $maxTsat \in \mathrm{PF}^{\mathrm{NPMV}}_{tt}$.

We should note that several of the classes we investigate here seem to capture the complexity of finding witnesses to NP-optimization problems. This observation is explored by Chen and Toda [CT92] and by Wareham [War92].

Theorem 1 states that unbounded access to an oracle in NPMV is no more powerful than such an access to an oracle in NP.

3 Bounded Query Classes

Now we state our main results; proofs are given in the full draft paper. General techniques developed in this section are reminiscent of the "mind-change" technique of [Bei91, KSW87].

Theorem 2 *For every* $k \geq 1$, $\mathrm{PF}^{\mathrm{NPMV}[k]} = \mathrm{PF}_{tt}^{\mathrm{NPMV}[2^k-1]} \subseteq_c \mathrm{NPMV} \circ \mathrm{PF}^{\mathrm{NP}[k]} \subseteq \mathrm{NPMV} \circ \mathrm{PF}_{tt}^{\mathrm{NP}[2^k-1]} \subseteq \mathrm{PF}^{\mathrm{NPMV}[k+1]} = \mathrm{PF}_{tt}^{\mathrm{NPMV}[2^{k+1}-1]}$.

For general bounded query classes, it is not known whether $\mathrm{PF}^{\mathrm{NPMV}[k]} \subseteq_c \mathrm{PF}^{\mathrm{NP}[k]}$. But, for reduction classes of sets, this type of equivalence holds.

Theorem 3 *For every* $k \geq 1$, $\mathrm{P}^{\mathrm{NPMV}[k]} = \mathrm{P}^{\mathrm{NP}[k]}$.

Theorem 4 *For every* $k \geq 1$, $\mathrm{P}_{tt}^{\mathrm{NPMV}[k]} = \mathrm{P}_{tt}^{\mathrm{NP}[k]}$.

We denote the k-th level of the Boolean hierarchy as $\mathrm{NP}(k)$. By definition,

- $\mathrm{NP}(1) = \mathrm{NP}$, and
- for every $k \geq 2$, $\mathrm{NP}(k) = \mathrm{NP} - \mathrm{NP}(k-1)$.

The Boolean hierarchy over NP, denoted by BH is the union of all $\mathrm{NP}(k)$, $k \geq 1$.

The following theorems give evidence to show that bounded query hierarchies do not collapse.

Theorem 5 *Let* $k \geq 1$. *If* $\mathrm{PF}^{\mathrm{NPMV}[k+1]} = \mathrm{PF}^{\mathrm{NPMV}[k]}$, *then* BH *collapses to its* 2^k*-th level.*

Theorem 6 *Let* $k \geq 1$. *If* $\mathrm{PF}_{tt}^{\mathrm{NPMV}[k+1]} = \mathrm{PF}_{tt}^{\mathrm{NPMV}[k]}$, *then* BH *collapses to its* $(k+1)$*-st level.*

Analogous to the theorems stated so far, the following theorems hold for reduction classes that make logarithmic many queries to partial functions in NPMV.

Theorem 7 *1.* $\mathrm{PF}^{\mathrm{NPMV}[\log]} = \mathrm{PF}_{tt}^{\mathrm{NPMV}}$.

2. $\mathrm{NPMV} \circ \mathrm{PF}^{\mathrm{NP}[\log]} = \mathrm{NPMV} \circ \mathrm{PF}_{tt}^{\mathrm{NP}}$.

3. $\mathrm{PF}^{\mathrm{NPMV}[\log]} \subseteq_c \mathrm{NPMV} \circ \mathrm{PF}^{\mathrm{NP}[\log]}$.

4. $\mathrm{NPMV} \circ \mathrm{PF}^{\mathrm{NP}[\log]} \subseteq \mathrm{PF}^{\mathrm{NPMV}[\log]}$.

Theorem 8 $\mathrm{P}^{\mathrm{NPMV}[\log]} = \mathrm{P}_{tt}^{\mathrm{NPMV}} = \mathrm{P}^{\mathrm{NP}[\log]} = \mathrm{P}_{tt}^{\mathrm{NP}}$.

4 Nondeterministic Polynomial-Time Reductions

We define nondeterministic reductions between partial functions so that the access mechanism is identical to that for deterministic reductions. Namely, let f be a single-valued partial function and N be a polynomial-time nondeterministic oracle Turing machine. $N[f]$ denotes a multivalued partial function computed by N with oracle f in accordance with the following mechanism:

- when N asks about $y \in dom(f)$, f returns $f(y)$ and
- when N asks about $y \notin dom(f)$, f answers a special symbol $\perp$.

Let f and g be multivalued partial functions. We say that f is nondeterministic polynomial-time Turing reducible to g, denoted by $f \leq_{\mathrm{T}}^{\mathrm{NP}} g$ if there is a polynomial-time nondeterministic Turing machine N satisfying the following conditions: for every x and for every single-valued refinement g' of g,

- $x \in dom(f)$ if and only if $x \in dom(N[g'])$ and
- if $N[g']$ maps x to y, then f maps x to y.

In other words, $N[g']$ is a refinement of f.

Let $\mathcal{F}$ be a class of partial multivalued functions. $\mathrm{NPMV}^{\mathcal{F}}$ denotes the class of partial multivalued functions that are $\leq_{\mathrm{T}}^{\mathrm{NP}}$-reducible to some $g \in \mathcal{F}$. $\mathrm{NPMV}^{\mathcal{F}[k]}$ denotes the class of partial multivalued functions that are $\leq_{\mathrm{T}}^{\mathrm{NP}}$-reducible to some $g \in \mathcal{F}$ via a machine that makes k adaptive queries to its oracle.

$\mathrm{NPMV}_{tt}^{\mathcal{F}}$ denotes the class of partial multivalued functions that are $\leq_{\mathrm{T}}^{\mathrm{NP}}$-reducible to some $g \in \mathcal{F}$ via a machine that makes nonadaptive queries to its oracle. $\mathrm{NPMV}_{tt}^{\mathcal{F}[k]}$ denotes the class of partial multivalued functions that are $\leq_{\mathrm{T}}^{\mathrm{NP}}$-reducible to some $g \in \mathcal{F}$ via a machine that makes k nonadaptive queries to its oracle.

For a class of sets $\mathcal{C}$, we write $\mathrm{NPMV}^{\mathcal{C}}$ to denote the class of multivalued partial functions that are computed by an nondeterministic Turing machine relative to an oracle in $\mathcal{C}$. $\mathrm{NPMV}^{\mathcal{C}[k]}$, $\mathrm{NPMV}_{tt}^{\mathcal{C}}$ and $\mathrm{NPMV}_{tt}^{\mathcal{C}[k]}$ are defined similarly.

For $k \geq 1$, $\Sigma\mathrm{MV}_k$ denotes $\underbrace{\mathrm{NPMV}^{\mathrm{NPMV}^{\cdots^{\mathrm{NPMV}}}}}_{k}$.

Lemma 1 *For every $k \geq 1$, $\Sigma\mathrm{MV}_k = \mathrm{NPMV}^{\Sigma_{k-1}^p[1]}$ and for every $f \in \Sigma\mathrm{MV}_k$, $dom(f) \in \Sigma_k^p$.*

From this lemma we yield the following theorem.

Theorem 9 *Let f be a partial multivalued function. For every $k \geq 1$, the following statements are equivalent:*

(i) *f is in $\Sigma\mathrm{MV}_k$;*

(ii) *f is polynomially length-bounded, $dom(f) \in \Sigma_k^p$, and $graph(f) \in \Sigma_k^p$;*

(iii) *f is polynomially length-bounded and $graph(f) \in \Sigma_k^p$.*

Theorem 10 *For every $k \geq 1$, $\Sigma\mathrm{MV}_{k+1} = \Sigma\mathrm{MV}_k$ if and only if $\Sigma_{k+1}^p = \Sigma_k^p$.*

Thus, these classes form function analogues of the polynomial hierarchy, and, unless the polynomial hierarchy collapses, they form a proper hierarchy.

5 The Difference Hierarchy

Let $\mathcal{F}$ be a class of partial multivalued functions. A partial multivalued function f is in co$\mathcal{F}$ if there exist $g \in \mathcal{F}$ and a polynomial p such that for every x and y

- $f(x)$ maps to y if and only if $|y| \leq p(|x|)$ and $g(x)$ does not map to y.

Let $\mathcal{F}$ and $\mathcal{G}$ be two classes of partial multivalued functions. A partial multivalued function h is in $\mathcal{F} \wedge \mathcal{G}$ if there exist partial multivalued functions $f \in \mathcal{F}$ and $g \in \mathcal{G}$ such that for every x and y,

- $h(x)$ maps to y if and only if $f(x)$ maps to y and $g(x)$ maps to y.

A partial multivalued function h is in $\mathcal{F} \vee \mathcal{G}$ if there exist partial multivalued functions $f \in \mathcal{F}$ and $g \in \mathcal{G}$ such that for every x and y,

- $h(x)$ maps to y if and only if $f(x)$ maps to y or $g(x)$ maps to y.

$\mathcal{F} - \mathcal{G}$ denotes $\mathcal{F} \wedge \mathrm{co}\mathcal{G}$.

NPMV(k) is the class of partial multivalued functions defined in the following way:

1. NPMV(1) = NPMV, and
2. for $k \geq 2$, NPMV(k) = NPMV $-$ NPMV($k-1$).

Lemma 2 *For every $k \geq 1$, $f \in \mathrm{NPMV}(k)$ if and only if f is polynomially length-bounded and $graph(f) \in \mathrm{NP}(k)$.*

This lemma is proved by induction. We use it to obtain the following theorem.

Theorem 11 *For every* $k \geq 1$, $\mathrm{NPMV}(k+1) = \mathrm{NPMV}(k)$ *if and only if* $\mathrm{NP}(k+1) = \mathrm{NP}(k)$.

Theorem 12 NP = co-NP *if and only if* NPMV $\subseteq$ coNPMV *if and only if* coNPMV $\subseteq$ NPMV.

A function f is said to be NPMV-low if $\mathrm{NPMV}^f = \mathrm{NPMV}$.

Theorem 13 *A function* f *is* NPMV*-low if and only if* $f \in \mathrm{NPMV}$ *with* $dom(f) \in \mathrm{NP} \cap$ co-NP.

Theorem 14 $\mathrm{NPMV}(2) \subseteq_c \mathrm{PF}^{\mathrm{NPMV}}$ *if and only if* $\Sigma_2^p = \Delta_2^p$.

Theorem 15 $\mathrm{PF}^{\mathrm{NPMV}[k]} \subseteq \mathrm{NPMV}(2^{k+1} - 1)$.

By Theorem 11, the levels of the difference hierarchy of partial functions are distinct if and only if the same levels of the Boolean hierarchy are distinct. Yet, whereas the Boolean hierarchy resides entirely within P^{NP}, by Theorem 14, this is unlikely to be true of the difference hierarchy of partial functions.

6 Reduction classes to NPSV

In this section, we study the reduction classes to NPSV, $\mathrm{PF}^{\mathrm{NPSV}}$, $\mathrm{PF}^{\mathrm{NPSV}[k]}$, $\mathrm{PF}^{\mathrm{NPSV}[\log]}$, $\mathrm{PF}_{tt}^{\mathrm{NPSV}}$, and $\mathrm{PF}_{tt}^{\mathrm{NPSV}[k]}$. These classes contain only single-valued functions. The following proposition is easy to prove.

Proposition 1 *1.* $\mathrm{PF}^{\mathrm{NP}} = \mathrm{PF}^{\mathrm{NPSV}}$ *and* $\mathrm{PF}_{tt}^{\mathrm{NP}} = \mathrm{PF}_{tt}^{\mathrm{NPSV}}$.

2. $\mathrm{PF}^{\mathrm{NP}[k]} \subseteq \mathrm{PF}^{\mathrm{NPSV}[k]} \subseteq \mathrm{PF}^{\mathrm{NPMV}[k]}$ *and* $\mathrm{PF}^{\mathrm{NP}[\log]} \subseteq \mathrm{PF}^{\mathrm{NPSV}[\log]} \subseteq \mathrm{PF}^{\mathrm{NPMV}[\log]}$.

3. $\mathrm{PF}_{tt}^{\mathrm{NP}[k]} \subseteq \mathrm{PF}_{tt}^{\mathrm{NPSV}[k]} \subseteq \mathrm{PF}_{tt}^{\mathrm{NPMV}[k]}$.

4. $\mathrm{P}_{tt}^{\mathrm{NP}} = \mathrm{P}_{tt}^{\mathrm{NPSV}}$ *and* $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NPSV}}$.

5. $\mathrm{P}^{\mathrm{NP}[k]} \subseteq \mathrm{P}^{\mathrm{NPSV}[k]} \subseteq \mathrm{P}^{\mathrm{NPMV}[k]}$ *and* $\mathrm{P}^{\mathrm{NP}[\log]} \subseteq \mathrm{P}^{\mathrm{NPSV}[\log]} \subseteq \mathrm{P}^{\mathrm{NPMV}[\log]}$.

6. $\mathrm{P}_{tt}^{\mathrm{NP}[k]} \subseteq \mathrm{P}_{tt}^{\mathrm{NPSV}[k]} \subseteq \mathrm{P}_{tt}^{\mathrm{NPMV}[k]}$.

The following theorems follow as corollaries of results proven in the previous sections.

Theorem 16 *For every* $k \geq 1$, $\mathrm{P}^{\mathrm{NPSV}[k]} = \mathrm{P}^{\mathrm{NP}[k]}$.

Theorem 17 *For every* $k \geq 1$, $\mathrm{P}_{tt}^{\mathrm{NPSV}[k]} = \mathrm{P}_{tt}^{\mathrm{NP}[k]}$.

Theorem 18 *If* $\mathrm{PF}_{tt}^{\mathrm{NPSV}[k+1]} = \mathrm{PF}_{tt}^{\mathrm{NPSV}[k]}$ *for some* $k \geq 1$, *then* BH *collapses to its* $(k+1)$*-st level.*

Theorem 19 *If* $\mathrm{PF}^{\mathrm{NPSV}[k+1]} = \mathrm{PF}^{\mathrm{NPSV}[k]}$ *for some* $k \geq 1$, *then* BH *collapses to its* 2^k*-th level.*

参考文献

[Bei91] R. Beigel. Bounded queries to SAT and the boolean hierarchy. *Theor. Computer Science*, 84(2):199–223, 1991.

[CT92] Z. Chen and S. Toda. On the complexity of computing optimal solutions. Department of Computer Science and Information Mathematics, University of Electro-Communications, Chufo-shi, Tokyo 182, Japan, 1992.

[Kre88] M. Krentel. The complexity of optimization problems. *J. Computer Systems Sci.*, 36:490–509, 1988.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *Theoretical Informatics and Applications (RAIRO)*, 21:419–435, 1987.

[Sel92] A. Selman. A taxonomy of complexity classes of functions. *J. Comput. Sytsem Sci.*, 1992. In press.

[War92] H. Wareham. Masters thesis. Department of Computer Science, Memorial University of Newfoundland, 1992.