A Hierarchy Based on Output Multiplicity

Ashish V. Naik[*] John D. Rogers[†] James S. Royer[‡]
Chronologic Simulation DePaul University Syracuse University

Alan L. Selman[§]
University at Buffalo

August 1, 1997

[*]Viewlogic Systems Inc., 20230 Stevens Creek Blvd., Cupertino, CA 9501. Research performed at the Department of Computer Science, University of Chicago, with support fron NSF grant CCR-9253582, and at the Department of Computer Science, University at Buffalo, with support from NSF grant CCR-9400229.

[†]School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, IL 60604. The second author gratefully acknowledges a visiting scholar appointment to the University of Chicago that facilitated this work.

[‡]Department of Elec. Engrg. and Computer Science, Syracuse University, Syracuse, NY 13244. Research supported in part by NSF grant CCR-9522987.

[§]Department of Computer Science, University at Buffalo, Buffalo, NY 14260. Research supported in part by NSF grant CCR-9400229. This author presented some of the results herein in preliminary form at the Eleventh IEEE Conference on Computational Complexity.

Abstract

The class NPkV consists of those partial, multivalued functions that can be computed by a nondeterministic, polynomial time-bounded transducer that has at most k distinct values on any input. We define the *output-multiplicity hierarchy* to consist of the collection of classes NPkV, for all positive integers $k \geq 1$. In this paper we investigate the strictness of the output-multiplicity hierarchy and establish three main results pertaining to this:

1. If for any $k > 1$, the class NPkV collapses into the class NP$(k-1)$V, then the polynomial hierarchy collapses to Σ_2^{P}.
2. If the converse of the above result is true, then any proof of this converse cannot relativize. We exhibit an oracle relative to which the polynomial hierarchy collapses to P^{NP}, but the output-multiplicity hierarchy is strict.
3. Relative to a random oracle, the output-multiplicity hierarchy is strict. This result is in contrast to the still open problem of the strictness of the polynomial hierarchy relative to a random oracle.

In introducing the technique for the third result we prove a related result of interest: relative to a random oracle UP $\neq$ NP.

1 Introduction

One of the central questions about any complexity-theoretic measure is that of fine hierarchies, that is, how small a change in computing resources need one make to bring about a change in computing power. Here we investigate a hierarchy based on the number of distinct output values of members of the class NPMV, the class of partial multivalued functions that are computed by polynomial time-bounded nondeterministic transducers.

Nondeterministic transducers compute *partial multivalued functions*, partial because nondeterministic computations do not necessarily accept every input, and multivalued because nondeterministic computations may output different values on different accepting paths. The study of polynomial-time computable classes of partial multivalued functions has become an increasingly active area of research [BLS84, GS88, Sel92, Sel94, FHOS97, HNOS96, Nai94, Ogi96, FGH$^+$96, FFNR96, JT95, Sel96]. The motivations for this study involve questions about NP search problems, the difficulty of inverting polynomial-time computable functions, and more generally, the power of nondeterminism. For detailed discussion of these motivations we refer the reader to the expository papers of Jenner and Toran [JT95] and Selman [Sel96].

A key notion in this area is that of a *refinement* of a function. We introduce this idea through an important example. Let PF denote the set of all partial functions that are computed by deterministic polynomial time-bounded transducers. A fundamental question is whether for each $f \in$ NPMV, there is a $g \in$ PF, so that $g(x)$ is always some value of $f(x)$. Indeed, this problem is equivalent to the question of whether P = NP [Sel92]. The relation between f and g is an instance of what we mean by one function, g, refining another, f. Formally, given partial multivalued functions f and g, we say that g is 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 also a value of $f(x)$. Let $\mathcal{F}$ and $\mathcal{G}$ be classes of partial multivalued functions. We define $f \in_c \mathcal{G}$ to mean that $\mathcal{G}$ contains a refinement of f, and we write $\mathcal{F} \subseteq_c \mathcal{G}$ if, for each $f \in \mathcal{F}$, $f \in_c \mathcal{G}$. This notation is consistent with the intuition that $\mathcal{F} \subseteq_c \mathcal{G}$ should entail that the complexity of $\mathcal{F}$ is no greater than the complexity of $\mathcal{G}$. Thus, "NPMV $\subseteq_c$ PF" means that every partial multivalued function in NPMV can be computed by some deterministic polynomial-time transducer. Using this notation, the assertion we made above states that NPMV $\subseteq_c$ PF if and only if P = NP.

Hemaspaandra *et al.* [HNOS96] addressed the question (raised by Selman [Sel94]) of whether every function in NPMV has a refinement in NPSV, where NPSV is the set of all partial single-valued functions $f \in$ NPMV. They proved that this is so only if the polynomial hierarchy collapses to Σ_2^{P}. Their proof actually shows more: that some 2-valued partial function in NPMV has no single-valued refinement unless the polynomial hierarchy collapses to Σ_2^{P}. This result suggests that the number of output values of an NP-transducer is a computing resource.

We define the *output-multiplicity hierarchy* to be the collection of all classes NPkV, $k \geq 1$, where these classes are defined as follows. For each $k \geq 1$, a partial multivalued function $f \in$ NPkV if and only if some refinement of f can be computed by a nondeterministic, polynomial time-bounded transducer that has at most k distinct values on any input. Thus, in particular, NP1V = NPSV. Once again, Hemaspaandra *et al.* [HNOS96] proved that if NP2V $\subseteq_c$ NPSV, then the polynomial hierarchy collapses to its second level.

In this paper we investigate the strictness of the output-multiplicity hierarchy and establish three main results pertaining to this:

1. We show, in Section 2, an extension of the result of Hemaspaandra *et al.*: If for some $k > 1$, NPkV $\subseteq_c$ NP$(k-1)$V, then the polynomial hierarchy collapses to Σ_2^{P}.

2. In Section 3, we show that if the converse of the above result is true, then any proof of this converse cannot relativize. We exhibit an oracle relative to which the polynomial hierarchy collapses to P^{NP} but the output-multiplicity hierarchy is strict.

3. We show, in Section 5, that the output-multiplicity hierarchy is strict relative to a random oracle. This result is in contrast to the still open problem of the strictness of the polynomial hierarchy relative to a random oracle.

In introducing the technique for the third result we prove in Section 4 a related result of interest: that relative to a random oracle UP $\neq$ NP.

1.1 Notation

Below $\Sigma = \{\mathbf{0}, \mathbf{1}\}$. For each natural number k and set X, $\mathcal{P}_k(X)$ denotes the collection of all k-element subsets of X. For each natural number k and $A \subseteq \Sigma^*$, $A^{=k} = A \cap \Sigma^k$. For any two languages A and B, $A \oplus B = 0A \cup 1B = \{0x \mid x \in A\} \cup \{1x \mid x \in B\}$.

Let $\langle \cdot, \cdot \rangle$ denote a standard polynomial-time computable pairing function with associated polynomial-time computable projections π_1 and π_2.

Recall that $\mathrm{PH} = \bigcup_{i \geq 0} \Sigma_i^{\mathrm{P}} = \bigcup_{i \geq 0} \Pi_i^{\mathrm{P}}$. For any partial multivalued function f, we write $f(x) \mapsto y$ if y is an output value of f on input x, and define

$$\textit{set-}f(x) \quad = \quad \{\, y \mid f(x) \mapsto y \,\}.$$

For partial multivalued functions f and g, observe that g is a refinement of f if and only if $\mathit{dom}(g) = \mathit{dom}(f)$ and for all $x \in \mathit{dom}(g)$, $\textit{set-}g(x) \subseteq \textit{set-}f(x)$. For any class $\mathcal{F}$ of partial multivalued functions, we let $\mathcal{F}_t$ denote the set of all total functions (i.e., the domain of f is Σ^*) that belong to $\mathcal{F}$.

Fenner *et al.* [FHOS97] studied polynomial-time reductions to NPSV (and NPMV) and introduced the classes $\mathrm{PF}^{\mathrm{NPSV}}$ and $\mathrm{PF}^{\mathrm{NPSV}}(b(n))$ which we define just below. First, let us make the convention that when a query y to a function oracle $g \in \mathrm{NPSV}$ is made, then either (a) the value of $g(w)$ is returned, if $g(w)$ is defined, or else (b) a unique flag $\perp$ is returned, indicating that g is undefined on w. Now, we say that f is in $\mathrm{PF}^{\mathrm{NPSV}}$ if f is computed by a deterministic, polynomial time-bounded oracle Turing machine transducer that accesses an oracle g belonging to NPSV; we say that f is in $\mathrm{PF}^{\mathrm{NPSV}}(b(n))$ if $f \in \mathrm{PF}^{\mathrm{NPSV}}$ and, for some transducer and g that witness this, the number of queries made by the transducer on any input x is no more than $b(|x|)$.

2 If the Output-Multiplicity Hierarchy Collapses, So Does the Polynomial Hierarchy

Once it was known that NP2V $\subseteq_c$ NPSV implies PH $= \Sigma_2^{\mathrm{P}}$ [HNOS96], one natural question to raise was whether every partial multivalued function in NPMV has a refinement in some reduction class to NPSV. Would such an hypothesis still collapse the polynomial hierarchy? The only significant work on this question is due to Ogihara who proved the following result:

Theorem 1 (Ogihara [Ogi96]) *Let $c < 1$ be a constant. If every multivalued function in* NPMV *has a refinement in* $\mathrm{PF}^{\mathrm{NPSV}}(c \log\, n)$*, then* PH $= \Sigma_2^{\mathrm{P}}$.

The proof of the following theorem will not involve reductions to NPSV but will rely on ideas and techniques of Ogihara's proof.

Theorem 2 *Let $k > 1$. If* NPkV $\subseteq_c$ NP$(k-1)$V*, then* PH $= \Sigma_2^{\mathrm{P}}$.

The rest of this section is devoted to the proof of this theorem. To begin, we want a partial multivalued function f that obviously belongs to the class NPkV but that intuitively has no refinement g in NP$(k-1)$V. This leads us to the property of *selectivity*. We say that a set A is *k-selective* (for $k > 0$) if there is a partial multivalued function f from $\mathcal{P}_k(\Sigma^*)$ to $\mathcal{P}_{k-1}(\Sigma^*)$ such that, for each k-element set Y,

1. every member of *set-*$f(Y)$ is a subset of Y, and
2. if at least $k-1$ of the strings in Y belong to A, then *set-*$f(Y)$ is nonempty and every member of it is a subset of A (i.e., $Z \in$ *set-*$f(Y) \Rightarrow Z \subset A$).

We call f as above a *k-selector* of A. By an abuse of notation, we will treat 2-selectors as if they were partial, multivalued functions from $\Sigma^* \times \Sigma^*$ to Σ^*.

We introduce the following running example to help illustrate our notions.

Example 3 *A is 2-selective if there is a partial multivalued function f defined on ordered pairs such that*

$$\textit{set-}f(x, y) \;\subseteq\; \{x, y\}$$

and such that if $x \in A$ or $y \in A$, then

$$\emptyset \;\neq\; \textit{set-}f(x, y) \;\subseteq\; A.$$

For those familiar with prior work on selectivity, we note that a set A is *p-selective* [Sel79] if and only if A has a 2-selector that belongs to PF_t and that A is NPMV-*selective* [HNOS96] if and only if A has a 2-selector that belongs to NPMV.

Claim: For each $k > 0$, every $A \in \mathrm{NP}$ has a k-selector that belongs to $\mathrm{NP}k\mathrm{V}$.

Proof: Fix k. Define a nondeterministic transducer M that, on input $Y \in \mathcal{P}_k(\Sigma^*)$, does the following. First, M nondeterministically guesses a $k-1$ element subset Z of Y. Next, M nondeterministically tries to discover whether $Z \subseteq A$ and, if this test is successful, then M outputs the set Z. Since Y has k distinct subsets of size $k-1$, we see that M computes a element of $\mathrm{NP}k\mathrm{V}$. Hence, the claim follows.

Henceforth in the proof, we take as a hypothesis that $\mathrm{NP}k\mathrm{V} \subseteq_c \mathrm{NP}(k-1)\mathrm{V}$. The reader can easily see that if f is a k-selector for A and g is a refinement of f, then g is a k-selector for A. Hence from the hypothesis and the above claim it follows that *every A in* NP *has a k-selector that belongs to* $\mathrm{NP}(k-1)\mathrm{V}$. We will show that this implies that $\Pi_2^{\mathrm{P}} = \Sigma_2^{\mathrm{P}}$.

Example 3, continued *Let us fix A to be* SAT *and let f be a 2-selector for* SAT *that belongs to* NPMV. *A single-valued refinement of f is a single-valued partial function g such that if either $x \in$* SAT *or $y \in$* SAT*, then $g(x, y)$ is defined and a member of* SAT.

Intuitively, one does not expect a single-valued function, such as the g above, to be able to determine which of two formulas is satisfiable. This intuition is borne out by the result of Selman [Sel79] that SAT is p-selective if and only if SAT $\in$ P and by the result of Hemaspaandra *et al.* [HNOS96] that SAT is NPSV-selective only if $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/poly$.

Continuing with the proof, let $L \in \Pi_2^{\mathrm{P}}$. Then, there exist a polynomial p and a set $A \in \mathrm{NP}$ such that, for all x,

$$x \in L \iff \forall y \in \Sigma^{p(|x|)} \, [\, \langle x, y \rangle \in A \,]. \tag{1}$$

We may assume of our pairing function that there is a polynomial q such that for all strings x of length n and all strings y of length $p(n)$, $|\langle x, y \rangle| = q(n)$. Hence, the right-hand side of (1) is equivalent to: $\forall y \in \Sigma^{p(|x|)} \, [\, \langle x, y \rangle \in$

$A^{=q(|x|)}$]. As argued above, it follows from our hypothesis that A has a k-selector g that belongs to NP$(k-1)$V.

Given a string $x \in \Sigma^{q(n)}$ and a $(k-1)$-element set $Z \subseteq A^{=q(n)}$, we say that *x loses to Z* if every output value of $g(Z \cup \{x\})$ contains x.

Claim: If x loses to Z, then $x \in A$.

Proof: Since $Z \subseteq A^{=q(n)}$, *set-*$g(Z \cup \{x\}) \neq \emptyset$. Furthermore, for every output value Y, $Y \subseteq A$. Thus, for each such Y, $x \in Y \subseteq A$. Hence, the claim follows.

Example 3, continued *For $z \in \mathrm{SAT}^{=n}$, a string x loses to z if $g(x,z) = x$. So, by definition of a selector, we must have $x \in \mathrm{SAT}$.*

The following lemma is due to Ogihara [Ogi96].

Lemma 4 *For each $n \geq 1$, there is a set $S_n = \{Z_1, \ldots, Z_m\}$, $m \leq q(n)$, such that for every i, $1 \leq i \leq m$, $Z_i \subseteq A^{=q(n)}$, $\|Z_i\| = k-1$, and for all $x \in \Sigma^{q(n)}$, $x \in A^{=q(n)}$ if and only if for some i with $1 \leq i \leq m$, x loses to Z_i.*

Example 3, concluded *Lemma 4 asserts the existence of a set of strings $S_n = \{z_1, \ldots, z_m\}$, $m \leq q(n)$, such that $S_n \subseteq \mathrm{SAT}^{=q(n)}$, and for each $x \in \mathrm{SAT}^{=q(n)}$, there is a z_i such that $g(x, z_i) = x$.*

We will not give the proof of Lemma 4. The argument is similar to proofs of Ko [Ko83] and of later researchers [LS93, HNOS96] that dealt essentially with the scenario of Example 3. The combinatorics of Ogihara's argument is necessarily more involved. The key idea of the proof is to note that some set Z is a winner to more than the average number of strings x (meaning that x loses to Z). So to construct S_n, start with $S_n = \emptyset$, place such a Z into S_n, delete from consideration all strings that lose to Z, and continue this process until all remaining strings have been deleted from consideration.

Define a string u to be *correct for length $q(n)$* if u encodes a pair $\langle S, \mathit{WIT}\rangle$ such that $S = \{Z_1, \ldots, Z_m\}$ and $\mathit{WIT} = \{W_1, \ldots, W_m\}$, $m \leq q(n)$, that satisfy the following three conditions.

(i) For all i, $1 \leq i \leq m$, $\|Z_i\| = k-1$.

(ii) For all i, $1 \leq i \leq m$, $Z_i \subseteq A^{=q(n)}$ and W_i is a set of witnesses proving that $Z_i \subseteq A^{=q(n)}$.

(iii) For all $x \in A^{=q(n)}$ there exists i, $1 \leq i \leq m$, such that x loses to Z_i.

If u is correct for length $q(n)$, we write $Loses(x, u)$ to mean x loses to some Z_i.

Claim: For all x,

$$x \in L \iff \exists u \left[\begin{array}{c} (u \text{ is correct for length } q(|x|)) \\ \text{and } \ \forall y \, Loses(\langle x, y \rangle, u) \end{array} \right]. \tag{2}$$

Proof: The implication from left to right follows from Lemma 4. The implication from right to left is straightforward. Hence the claim follows.

Note that by the definitions of "correct for length $q(|x|)$" and "Loses$(\cdot,\cdot)$," it follows that we can replace the "$\exists u$" and "$\forall y$" in the right-hand side of (2) with "$\exists u \in \Sigma^{p_0(|x|)}$" and "$\forall y \in \Sigma^{p_1(|x|)}$" for some appropriate polynomials p_0 and p_1.

To complete the argument that $L \in \Sigma_2^{\mathrm{P}}$, we merely have to prove that the predicates

1. "u is correct for length $q(|x|)$," and
2. $Loses(\langle x, y \rangle, u)$

are in coNP.

To prove that "u is correct for length $q(|x|)$" belongs to coNP, we give the following NP-algorithm for the complement "u is not correct for length $q(|x|)$": If u does not encode a pair $\langle S, \mathit{WIT} \rangle$ that satisfies the defining conditions (i) and (ii), then accept. Otherwise, we have $S = \{ Z_1, \ldots, Z_m \}$ and for each i, $Z_i \subseteq A^{=q(n)}$. Thus, and this is the important observation, for each $x \in \Sigma^{q(n)}$ and each Z_i, $g(Z_i \cup \{ x \})$ is defined. Nondeterministically select an $x \in A^{=q(n)}$. For each i, compute an output value Y of $g(Z_i \cup \{ x \})$ and verify that $x \notin Y$. If each of these tests is successful, then accept.

The proof that the second predicate belongs to coNP is similar.

This completes the proof of Theorem 2.

3 Collapsing the Polynomial Hierarchy while Leaving the Output-Multiplicity Hierarchy Strict

In the previous section we showed that if the output-multiplicity hierarchy collapses at any level, then the polynomial hierarchy collapses to Σ_2^{P}. In this section we demonstrate an oracle relative to which the converse of this result is false. Specifically, we exhibit an oracle relative to which the polynomial hierarchy collapses to Δ_2^{P} while the output-multiplicity hierarchy is proper. The oracle we use is a *generic* oracle derived from conditions with certain restrictions placed on them. Generic oracles based on restricted conditions have been investigated and applied by a number of researchers, notably by Fenner *et al.* [FFKL93], Fortnow [FR94], and Rogers [Rog97].

It is easy to find an oracle relative to which both hierarchies collapse. Because the proof of the previous section relativizes, any oracle making the polynomial hierarchy proper [Has86, Yao85] will also make the output-multiplicity hierarchy proper. Thus, oracles exist for all possible scenarios concerning the relationships between a collapse of the polynomial hierarchy and a collapse of the output-multiplicity hierarchy.

3.1 Generic oracles

A *condition* is a partial function from Σ^* to $\{0,1\}$. A condition σ *extends* another condition τ if for all $x \in \mathrm{domain}(\tau)$, $\sigma(x) = \tau(x)$. Two conditions σ and τ are *compatible* if for all $x \in \mathrm{domain}(\sigma) \cap \mathrm{domain}(\tau)$, $\sigma(x) = \tau(x)$. They *conflict* otherwise.

In this paper, we only consider conditions having finite domains. We assume that if a condition is defined on a string of some length n, then it is defined on all strings of length no greater than n.

A condition σ is *gappy* if, whenever $\sigma(x) = 1$, the length of x is *acceptable*. An *acceptable length* is an integer in the range of the *tower* function, which has the recursive definition $tower(0) = 2$ and $tower(n+1) = 2^{tower(n)}$. That is, $tower(n)$ is an exponential tower of 2's with height $n+1$. A related function is $\log^*$, which has the recursive definition $\log^*(0) = \log^*(1) = \log^*(2) = 0$ and

$\log^*(n) = 1 + \log^*(\lceil \log(n) \rceil)$ $(n > 2)$. For values in the range of *tower*, $\log^*$ computes $tower^{-1}$.

Let $n = \langle n_0, n_1 \rangle$. An *output-multiplicity condition* (a.k.a. *om-condition*) σ is a gappy condition with a finite domain such that, at every acceptable length $tower(\langle n_0, n_1 \rangle)$, there are at most n_1 strings x for which $\sigma(x) = 1$.

For any condition τ, we define the total function $\overline{\tau} : \Sigma^* \times \{0,1\} \to \{0,1\}$ as follows:

$$\overline{\tau}(x,i) = \begin{cases} 1 & \text{if } x \in dom(\tau) \text{ and } t(x) = i, \\ 0 & \text{otherwise.} \end{cases}$$

A set of conditions is *definable* if the set $\{\overline{\tau} \mid \tau \in S\}$ is a Π^1_1 class.

A set S of om-conditions is *dense* if, for every om-condition τ, there is an om-condition σ in S such that σ extends τ. A language (oracle) A is *om-generic* if, for every definable dense set S of om-conditions, A extends some σ in S.

As in earlier papers [FFK96, FFKL93, FR94], it is easy to show that every om-condition τ is extended by some om-generic language A. In particular, om-generic languages exist.

For every $n \geq 1$ and $k \geq 1$, the set $S_{\langle n,k \rangle}$ of all om-conditions that are defined at length $tower(\langle n,k \rangle)$ is dense and definable. Thus, for every $n \geq 1$ and $k \geq 1$, every om-generic oracle is defined at length $tower(\langle n,k \rangle)$. In particular, if G is an om-generic oracle, then, for every $k \geq 1$, there are infinitely many acceptable lengths at which G contains no more than k strings. Moreover, G is a sparse set and has census function $O(\log^*(n))$.

3.2 Oracle Construction

Theorem 5 *There is an oracle C relative to which the polynomial hierarchy collapses to* P^{NP} *but the output-multiplicity hierarchy is proper; that is, for all $k \geq 1$, there is a function $f \in_c \mathrm{NP}(k{+}1)\mathrm{V}^C$ that has no refinement in* $\mathrm{NP}k\mathrm{V}^C$.

Proof. Let H be an oracle for which $\mathrm{P}^H = \mathrm{PSPACE}^H$. Relative to H, the polynomial hierarchy collapses to P and, for this reason, the output-multiplicity hierarchy collapses to PF. (Recall that NPMV $\subseteq_c$ PF if and only if P = NP.)

Let G be an om-generic oracle. Let $C = H \oplus G$. Recall that $\Delta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$. Long and Selman [LS86] proved that $\Sigma_2^{\mathrm{P}} = \Delta_2^{\mathrm{P}}$ if and only if $\Sigma_2^{\mathrm{P},S} = \Delta_2^{\mathrm{P},S}$ for all sparse sets S. Since their proof relativizes, recalling that G is sparse, it follows that $\Sigma_2^{\mathrm{P},C} = \Delta_2^{\mathrm{P},C}$. Thus, relative to C, the polynomial hierarchy collapses to P^{NP}.

Now our goal is to show that the output-multiplicity hierarchy is proper relative to C. For each oracle X, $k \geq 1$, and $x \in \Sigma^*$, we define the partial multivalued function f_k^X by

$$f_k^X(x) \mapsto y, \text{ if } |y| = |x|, \text{ for some } n \geq 1\ , |x| = \mathit{tower}(\langle n, k\rangle), \text{ and } y \in 1X.$$

Fix a $k > 1$. To see that f_k^C belongs to the class $\mathrm{NP}k\mathrm{V}^C$, let M be a nondeterministic oracle transducer that, on input x, nondeterministically guesses a string y, verifies that $|y| = |x|$ and that $|x| = \mathit{tower}(\langle n, k\rangle)$ for some n, and then outputs y if $y \in G$. Since G contains no more than k strings at any acceptable length $\mathit{tower}(\langle n, k\rangle)$, it follows that $f_k^C(x)$ has no more than k output values. Next, we will show, for each $k > 1$, that $f_k^C \notin_c \mathrm{NP}(k-1)\mathrm{V}^C$.

Let $\{M_i\}_{i\geq 1}$ be a standard indexing of noneterministic polynomial-time oracle Turing transducers, where the running time of each M_i on an input x is bounded by $(|x|+2)^i$, and where the run times are independent of their oracle. For each oracle X, we define the assertion R_i^X as follows:

> Either there is a string y such that the computation of $M_i^{H\oplus X}(y)$ outputs more than $k-1$ values or there is an $n \in N$ such that the output of the computation of $M_i^{H\oplus X}(\mathbf{0}^n)$ does not equal $\mathit{set}\text{-}f_k^{H\oplus X}(\mathbf{0}^n)$.

We argue that each R_i is true relative to every om-generic oracle, from which it follows immediately that $f_k^C \notin_c \mathrm{NP}(k-1)\mathrm{V}^C$.

Now we will view R_i as a requirement: We say that an om-condition σ *satisfies* requirement R_i if, for every oracle X extending σ, there is a string y such that either the computation of $M_i^{H\oplus X}(y)$ outputs more than $k-1$ values or its output is not equal to $\mathit{set}\text{-}f_k^{H\oplus X}(y)$. We will show that the set of om-conditions satisfying each R_i is definable and dense. Thus, each R_i will be true relative to every om-generic oracle.

The set of om-conditions satisfying R_i is certainly definable (including the fact that P = PSPACE relative to H). To show that it is dense, we demonstrate how, given any om-condition σ, we can extend it to an om-condition τ that satisfies R_i.

If there is an om-condition τ extending σ and a string y such that $M_i^{H\oplus\tau}(y)$ outputs more than $k-1$ values, then $M_i^{H\oplus X}(y)$ outputs more than $k-1$ values for every X that extends τ. In this case τ satisfies the first disjunct of requirement R_i, so we are done. If there is no such om-condition τ, then for every om-generic oracle G that extends σ and every input string y, $M_i^{H\oplus G}(y)$ outputs $k-1$ or fewer values. We say in this case that σ *forces* M_i to be an NP$(k-1)$V machine. This is the case that we need to consider.

Let $n = tower(\langle n_0, k\rangle)$ be an acceptable length on which σ is not defined such that $2^n > (k+1)(n+2)^{2i}$. Let τ denote an om-condition that extends σ. Setting $\tau(x) = 0$ for all x of length n is called *leaving the length empty.* If there is a way to do this and have $M_i^{H\oplus\overline{\tau}}(\mathbf{0}^n)$ accept, then we are done because $set\text{-}f_k^{H\oplus\overline{\tau}}(\mathbf{0}^n) = \emptyset \neq set\text{-}M_i^{H\oplus\overline{\tau}}(\mathbf{0}^n)$.

If we cannot do this, leaving the acceptable length empty forces $M_i^{H\oplus\overline{\tau}}$ to reject $\mathbf{0}^n$. This can happen only if along every accepting path of M_i (on input $\mathbf{0}^n$, keeping H fixed, and varying over om-conditions τ that extend σ), at least one string of length n in the oracle is positively queried. *Positively querying* a string x means that x is a query on the path that is found to belong to the oracle. Similarly, *negatively querying* a string means that it is queried and does not belong to the oracle. By setting $\tau(x) = 1$ for exactly one string x of length n, we need not consider computation paths that positively query two or more strings.

If there are computation paths that positively query exactly one string x and that accept and output a string $y \neq x$, then we can set $\tau(x) = 1$ and $\tau(z) = 0$ for all strings z of length n other than x. It follows that $set\text{-}f_k^{H\oplus\overline{\tau}}(\mathbf{0}^n) = \{x\}$ but $y \in set\text{-}M_i^{H\oplus\overline{\tau}}(\mathbf{0}^n)$.

We are left with the case that if a computation path positively queries exactly one string x, then it outputs x. Now we will show that there exists a string z of length n that is not positively queried by any accepting path. Setting $\tau(z) = 1$ and setting $\tau(y) = 0$ for all other strings y of length n yields $set\text{-}f_k^{H\oplus\overline{\tau}}(\mathbf{0}^n) = \{z\} \neq \emptyset = set\text{-}M_i^{H\oplus\overline{\tau}}(\mathbf{0}^n)$, which will complete the proof.

To prove that z exists, we will describe a process that, at the beginning of each step $j+1$ ($j \geq 0$), is given a set S_j of strings and a set A_j of accepting paths of $M_i(\mathbf{0}^n)$ that positively query exactly one string. When step $j+1$ is finished, it will yield a set $S_{j+1} \subset S_j$ and a set $A_{j+1} \subset A_j$. The process iterates through $(n+2)^i$ steps. At its conclusion, we show that $S_{(n+2)^i}$ is not empty but that $A_{(n+2)^i}$ is. We can then choose z from $S_{(n+2)^i}$.

Step 0: Let S_0 be the set of all length n strings and let A_0 be the set of accepting paths of $M_i(\mathbf{0}^n)$ that positively query exactly one string.

Step j+1: Select from A_j a set P_j of compatible accepting paths that maximizes the number of different values output on these paths. Because M_i is forced by σ to be an NP$(k-1)$V machine, these paths positively query and collectively output a set of strings $X_j = \{ x_1, \ldots, x_{k'} \}$ with $k' \leq k-1$. Divide P_j into k'-many subsets $P_{j,x_1}, \ldots, P_{j,x_{k'}}$ where each P_{j,x_i} contains the paths that positively query x_i.

Let $B_j = A_j - P_j$. Every accepting path in B_j must conflict with some path in P_j. Divide B_j into the sets $B_{j,neg}$, the set of paths that conflict because they negatively query one of the x_i, and $B_{j,pos}$, the set of paths that conflict because they positively query some string that is negatively queried by a path in P_j.

Let p be a path in $B_{j,pos}$ that positively queries some y. There must be some x_i such that p conflicts with every path in P_{j,x_i}. If this were not true, the set P_j would not have been selected so as to contain paths outputting the maximum number of strings. All of the paths in P_{j,x_i} conflict with p because each of them negatively queries y. Because the length of each accepting path is no greater than $(n+2)^i$ and because there are k' ($\leq k-1$) many of the P_{j,x_i}'s, there can be at most $(k-1)((n+2)^i - 1)$ such y. Let Y_j denote the set of these y.

Let S_{j+1} be $S_j - (X_j \bigcup Y_j)$. By the argument above, the cardinality of $X_j \bigcup Y_j$ is at most $(k-1)(n+2)^i$. Also, the set S_{j+1} is such that, if we set $\tau(z) = 1$ for some $z \in S_{j+1}$ and $\tau(z') = 0$ for all other length n strings z', the only computation paths that could be accepting paths are those in $B_{j,neg}$. This is so because S_{j+1} does not contain any string in X_j, so the computation paths in P_j cannot be accepting. Also, S_{j+1} does not contain any string in Y_j, so the computation paths in $B_{j,pos}$ cannot be accepting. Furthermore, every path in $B_{j,neg}$ negatively queries some string in $S_j - S_{j+1}$. Thus, A_{j+1} and B_{j+1} are as required.

End of step $j+1$.

Let $m = (n+2)^i$. After step $m-1$, we have the sets S_m and $B_{m,neg}$. The process guarantees that S_m is nonempty. To see this, recall that the cardinality of S_0 is 2^n, where n was chosen so that $2^n > m(k-1)(n+2)^{2i}$. At each step j, S_{j+1} is formed from S_j by removing at most $(k-1)(n+2)^i$ strings. So the cardinality of S_m is at least $2^n - m(k-1)(n+2)^{2i}$, which is greater than 0.

The process also guarantees that $B_{m,neg}$ is empty. To see this, recall that at the end of each step j, $B_{j,neg}$ only contains computation paths that negatively query some string $x_i \in X_j$. In the previous step $j-1$, these paths were in $B_{j-1,neg}$, and so negatively queried some string $x_k \in X_{j-1}$ such that $x_i \neq x_k$. Carrying this back to $B_{0,neg}$, it must be true that every path in $B_{j,neg}$ negatively queries j different strings. For a path p to be in $B_{m,neg}$, it would have to query negatively m different strings. But because a computation path can negatively query at most $m-1$ strings, p cannot exist.

This means that there is some string z not queried by any computation path. Setting $\tau(z) = 1$ and $\tau(z') = 0$ for all other z' of length n guarantees that $\textit{set-}f_k^{H\oplus\overline{\tau}}(\mathbf{0}^n) = \{z\} \neq \emptyset = \textit{set-}M_i^{H\oplus\overline{\tau}}(\mathbf{0}^n)$. $\square$

The class $\mathrm{NP}k\mathrm{V}_t$ is the set of all *total* k-valued functions that belong to $\mathrm{NP}k\mathrm{V}$. We can use the om-generic oracles to obtain the following result.

Corollary 6 *There is an oracle relative to which the output-multiplicity hierarchy is proper, but, for all* $k \geq 1$, $\mathrm{NP}k\mathrm{V}_t \subseteq_c \mathrm{PF}$.

Proof Sketch. An om-generic oracle is sparse and possesses the subset property defined by Fortnow and Rogers [FR94]. As Fortnow and Rogers ([FR94]) showed, proposition Q' holds. Fenner, et al. ([FFNR96]) showed, in a proof that relativizes, that Q' implies that, for all $k \geq 1$, $\mathrm{NP}k\mathrm{V}_t \subseteq_c \mathrm{PF}$. $\square$

4 Random Oracles and UP≠NP

In the next section we show that the output-multiplicity hierarchy is strict relative to a random oracle. This section introduces some of the tools and techniques for the output-multiplicity hierarchy theorem by showing a simpler, related result: that UP $\neq$ NP relative to a random oracle. The technique for both the UP $\neq$ NP and the output-multiplicy hierarchy is derived from Bennett and Gill's proof that NP $\neq$ coNP relative to a random oracle [BG81]. For a more detailed introduction to random oracle arguments and results, we refer the reader to Bennett and Gill's original paper [BG81] or either of Kurtz, Mahaney, and Royer's papers [KMR95, KMR92]. In particular, we will refer the reader to these papers on the ticklish matter of justifying the standard interpretation of random oracle separations. This standard interpretation is this: If two relativized classes separate relative to a random oracle *and* if neither of these classes is probabilistic, then this is evidence that the existence of strong one-way functions or pseudo-random generators may imply that the unrelativized versions of these classes separate.

4.1 Preliminaries

We identify the elements of N and Σ^* in the standard way: $n \in N \equiv$ the $(n+1)^{st}$ string in the lexicographical ordering on Σ^*. Recall that Σ^N is the collection of all total functions from N to $\{\mathbf{0}, \mathbf{1}\}$, or, equivalently, the collection of all infinite sequences of $\mathbf{0}$'s and $\mathbf{1}$'s. There is also a one-one correspondence between Σ^N and the collection of all languages over Σ given by: $R \in \Sigma^N \equiv \{ w : R(w) = \mathbf{1} \}$. We shall pun freely among these views of Σ^N.

To do probability over Σ^N, we adopt yet another view of Σ^N as the collection of all possible infinite sequences of independent tosses of a fair coin. Let $(\Sigma^N, \mathcal{E}, \mu)$ be the standard probability space on Σ^N, where $\mathcal{E}$ is the collection of *events* or *measurable sets* and μ is the *probability measure* or simply *measure* on this space that assigns each $\mathcal{A} \in \mathcal{E}$, a real number $\mu(\mathcal{A}) \in [0,1]$ (see, [Dud89, Oxt80, Rud66]). All of the $\mathcal{A} \subseteq \Sigma^N$ considered below are first-order definable (i.e., Borel) and standard results show that all such $\mathcal{A}$ are in $\mathcal{E}$.

A *tail set* is a $\mathcal{P} \subseteq \Sigma^N$ that is closed under finite variants, i.e., if X and $Y \in \Sigma^N$ are such that $X \triangle Y$ is finite, then $X \in \mathcal{P} \iff Y \in \mathcal{P}$. *Kolmogorov's* *zero-one law* [Oxt80, Theorem 21.3] states that a measurable tail set must have measure 0 or 1. If P is a predicate over Σ^N with $\mu(\{\, R : P(R) \,\}) = 1$, then we say *$P$ holds relative to a random oracle.* Structural properties such as $\{\, R : \mathrm{P}^R \neq \mathrm{NP}^R \,\}$ are first-order definable tail sets, and so have measure 0 or 1 by Kolmogorov's zero-one law. Informally, this means that there is a well-defined "measure 1" theory.

Our next goal is to state Lemma 7, a result by Stuart Kurtz [Kur83] that improves on a technical lemma of Bennett & Gill [BG81, Lemma 1]. Lemma 7 is a key tool in obtaining sufficient conditions for certain sets to be measure 1. Lemma 8 below gives a sample such application. We provide some preliminary definitions and conventions before stating this lemma. Suppose α and β are partial functions and $n \in N$. Define:

$$\alpha \lhd \beta \;=\; \lambda x \centerdot \begin{cases} \alpha(x), & \text{if } \alpha(x)\!\downarrow; \\ \beta(x), & \text{otherwise.} \end{cases}$$

$$\alpha \lhd_n \beta \;=\; \lambda x \centerdot \begin{cases} \alpha(x), & \text{if } x < n; \\ \beta(x), & \text{otherwise.} \end{cases}$$

In Lemma 7, read $\mathbf{A}(i, R, B)$ as "machine i with oracle R accepts language B," where the indexing of machines is over some restricted class of machines and "accepts" might mean something like "UP-accepts" or "BPP-accepts." For example, we could have $\mathbf{A}(i, R, B) \equiv$ [the i^{th} nondeterministic, relativized, polynomial time Turing machine with oracle R UP-accepts the set B]. Let D range over oracle dependent languages, where an *oracle dependent language* is simply a relativized language in the sense that it has a characteristic function of type $\Sigma^N \times \Sigma^* \to \{\, 0, 1 \,\}$. We say that D is *uniformly recursive* if there is an oracle Turing machine M such that, for all R, M^R decides D^R.

Lemma 7 *Suppose that 1 through 5 hold.*

1. *B is a uniformly recursive oracle dependent language and $\mathbf{A}$ is an arithmetic relation on $N \times \Sigma^N \times \Sigma^N$.*

2. *$\mathbf{A}$ is finitely patchable with respect to oracles. That is, there is a (not necessarily computable) function f, such that for each i, σ, D, and R, $\mathbf{A}(i, \sigma \lhd R, D^{\sigma \lhd R}) \Longrightarrow \mathbf{A}(f(i, \sigma), R, D^{\sigma \lhd R})$.*

3. $\mathbf{A}$ *is finitely patchable with respect to initial segments of uniformly R-recursive languages. That is, for each uniformly recursive oracle dependent language* C*, there is a (not necessarily computable) function* g_C *such that such that for each* i*,* n*,* D*, and* R*,* $\mathbf{A}(i, R, D^R) \implies \mathbf{A}(g_C(i,n), R, C^R \lhd_n D^R)$*.*

4. *Each bit of* R *affects only finitely many bits of* B^R*. That is, for each* σ*, there is an* n_σ *such that, for all* R*,* $\max(B^{\sigma \lhd R} \triangle B^R) < n_\sigma$*.*

5. *There is an* $a > 0$ *such that for each* i*,* $\mu^*(\{ R : \neg \mathbf{A}(i, R, B^R) \}) \geq a$*.*

Then, $\mu(\{ R : (\forall i)[\, \neg \mathbf{A}(i, R, B^R)\,] \}) = 1$*.*

The proof of this lemma is a simple density argument similar to Bennett and Gill's [BG81] proof of their Lemma 1.

4.2 UP $\neq$ NP Relative to a Random Oracle

We say that M, a polynomial time, nondeterministic Turing machine, UP-*accepts* a language A (written: $L_{\mathrm{UP}}(M) = A$) if $A = L(M)$ and, for each $a \in A$, there is exactly one accepting computation of M on input a. For each $R \subseteq N$ and $x \in N$ define

$$\begin{aligned} \xi^R(x) &= R(x\mathbf{1})R(x\mathbf{10}) \ldots R(x\mathbf{10}^{|x|-1}), \\ L^R &= \mathbf{0}^* \cap \mathrm{range}(\xi^R), \text{ and} \\ \widehat{L}^R &= \mathbf{0}^* - L^R. \end{aligned}$$

The function ξ is from Bennett and Gill's proof that NP $\neq$ coNP relative to a random oracle. If it *were* the case that NP = coNP relative to a random oracle, then by an application of Lemma 7 it follows that there would be a polynomial time, nondeterministic M for which, for most R, $L(M^R) = \widehat{L}^R$. Bennett and Gill showed that this is *not* the case, and, hence, that NP $\neq$ coNP relative to a random oracle. Here we extend the techniques of the Bennett and Gill argument to show that UP $\neq$ NP relative to a random oracle. Our strategy parallels Bennett and Gill's: we apply Lemma 7 to show that if it *were* the case that UP = NP relative to a random oracle, then there would be an M^R

that UP-accepts L^R for most R; then we show that, for an arbitrarily chosen M, M fails to UP^R accept L^R for most R.

Let M range over nondeterministic oracle Turing machines that have polynomial-bounded run times that are *independent* of their oracle. That is, for each M there is a polynomial $p(\cdot)$ such that for all x and R, M on input x and oracle R runs within $p(|x|)$ time. Let $\widetilde{M}$ range over polynomial-time, nondeterministic oracle Turing machines, so that the $\widetilde{M}$'s may have run times that depend on their oracles.

Lemma 8 *Suppose that there is an $a > 0$ such that, for all M, $\mu(\{\, R : M^R$ fails to* UP*-accept $L^R \,\}) \geq a$. Then, $\{\, R : L^R \notin \mathrm{UP}^R \,\}$ has measure 1.*

Proof. Clearly, $\{\, R : L^R \in \mathrm{UP}^R \,\} = \{\, R : (\exists \widetilde{M})[\, L^R = L_{\mathrm{UP}}(\widetilde{M}) \,]\,\}$. Let p range over polynomials and let $\widetilde{M}_p$ denote a version of $\widetilde{M}$ whose run time is clocked by p. An easy argument shows that

$$\{\, R : (\exists \widetilde{M})[\, L^R = L_{\mathrm{UP}}(\widetilde{M}^R) \,]\,\} \quad = \quad \{\, R : (\exists p)(\exists \widetilde{M})[\, L^R = L_{\mathrm{UP}}(\widetilde{M}_p^R) \,]\,\}.$$

Hence, $\{\, R : L^R \in \mathrm{UP}^R \,\} = \{\, R : (\exists M)[\, L^R = L_{\mathrm{UP}}(M^R) \,]\,\}$.

Let $\{M_i\}_{i \geq 1}$ be a standard indexing of nondeterministic oracle Turing machines that have polynomial-bounded run times that are independent of their oracle. It is straightforward to check that, when $\mathbf{A}(i, R, D) \equiv [\, D = L_{\mathrm{UP}}(M_i^R) \,]$ and $B = L$, hypotheses 1 through 4 of Lemma 7 are satisfied. The hypothesis of the present lemma implies hypothesis 5 of Lemma 7. Thus, by Lemma 7, $\mu(\{\, R : (\forall M)[\, M^R$ fails to UP-accept $L^R \,]\,\}) = 1$. Therefore, $\{\, R : L^R \notin \mathrm{UP}^R \,\}$ also has measure 1. $\square$ **Lemma 8**

We now need to understand the difficulties encountered by an M that "tries" to UP-accept L^R for most R. The nature of these difficulties is that success on a significant part of Σ^N entails failure on other significant parts of Σ^N. To understand this balance between successes and failures, we need to understand the structure that ξ^R imposes on Σ^R and computations over $\Sigma^N \times N$ (Definition 9 and Lemma 10) and to understand the measure-theoretic relation between regions of Σ^N where ξ^R has different behaviors (Lemma 11).

Variants and Interrogation

Definition 9 *Suppose R and $S \subseteq N$, $x, x_0, \dots, x_\ell, y \in \Sigma^n$, and that M is a nondeterministic relativized machine.*

*(a) R and S are $x_0, \dots, x_\ell$-*variants *(written $R \sim_{x_0,\dots,x_\ell} S$) if $R \triangle S \subseteq \{\, x_i\mathbf{10}^k : i \leq \ell \;\;\&\;\; k < |x_i| \,\}$, i.e., R and S are identical except perhaps on the strings that determine the value of ξ on arguments $x_0, \dots, x_\ell$.*

(b) Define

$$R/x_0, \dots, x_\ell \;\;=\;\; \Big(R - \Big\{\, x_i\mathbf{10}^k : i \leq \ell \;\;\&\;\; k < n \,\Big\}\Big).$$

In other words, $R/x_0, \dots, x_\ell$ is the $x_0, \dots, x_\ell$-variant of R that makes ξ map each of $x_0, \dots, x_\ell$ to $\mathbf{0}^n$.

(c) A particular computation of M on (R, y) examines *x if in the course of the computation the oracle R is queried about some string of the form $x\mathbf{10}^k$ for $k < |x|$; intuitively, the computation learns some information about the value of $\xi^R(x)$.*

(d) M on *(R, y)* interrogates *x if* every *computation of M on (R, y) examines x.*

(e) M on *(R, y)* depends on *x if there is an $S \sim_x R$ such that $M^R(y) \neq M^S(y)$.*

The notion of "examines" is a direct lift from Bennett and Gill. We note the following without proof.

Lemma 10 *Suppose that M is a nondeterministic relativized machine that on (R, y) runs in time t and accepts.*

(a) If M on (R, y) depends on x, then M on (R, y) interrogates x.

(b) The number of x's that M interrogates on (R, y) is $\leq t$.

Measure Scaling Maps

Terminology: Suppose T is a map from one probability space $(X_0, \mathcal{E}_0, \mu_0)$ to another $(X_1, \mathcal{E}_1, \mu_1)$. Suppose $a > 0$. T is an *a-measure scaling map* if $T\colon X_0 \to X_1$ is onto and, for all $\mathcal{A} \in \mathcal{M}_1$, we have $T^{-1}(\mathcal{A}) \in \mathcal{M}_0$ and $a \cdot \mu_0(T^{-1}(\mathcal{A})) = \mu_1(\mathcal{A})$. T is a *measure preserving map* if T is a 1-measure scaling map.

We are interested in a family of measure scaling maps involving particular regions of Σ^N defined by the behavior of ξ^R. For each k and $n \in N$, define

$$\begin{aligned} \mathcal{W}(n,k) &= \left\{ R : \left\| \left\{ x : \xi^R(x) = \mathbf{0}^n \right\} \right\| = k \right\} \\ &= \left\{ R : \text{there are exactly } k \text{ witnesses to } \mathbf{0}^n \in L^R \right\}, \text{ and} \\ w_{n,k} &= \mu(\mathcal{W}(n,k)). \end{aligned}$$

For each $k \leq 2^n$, it follows from some basic probability that

$$w_{n,k} = \binom{2^n}{k} \left(\frac{1}{2^n}\right)^k \left(1 - \frac{1}{2^n}\right)^{2^n - k} \tag{3}$$

and by some basic analysis that

$$\lim_{n\to\infty} w_{n,k} = \frac{1}{k! \cdot e}. \tag{4}$$

By (3) we have that, for each n and $k \leq 2^n$,

$$w_{n,k} = a_{n,k} \cdot w_{n,0}, \quad \text{where } a_{n,k} = \binom{2^n}{k} \left(\tfrac{1}{2^n - 1}\right)^k = \tfrac{1}{k!} + O(\tfrac{1}{2^n}).$$

Let $P_{n,k} = \{\, (x_0, x_1, \ldots, x_{k-1}) \in (\Sigma^n)^k : \text{the } x_i\text{'s are all pairwise distinct} \,\}$. Clearly, $\|P_{n,k}\| = 2^n \cdot (2^n - 1) \cdot \ldots \cdot (2^n - k + 1)$. We view $P_{n,k}$ as a measure space under the uniform, normalized counting measure, i.e., each $(\vec{x}) \in P_{n,k}$ has weight $\|P_{n,k}\|^{-1}$. Thus, $\mathcal{W}(n,0) \times P_{n,k}$ is a measure space under the product of the induced Lebesgue measure on $\mathcal{W}(n,0)$ and the normalized counting measure on $P_{n,k}$. Let $\mu_{n,k}$ be this product measure. By convention, let $(R, x_0, \ldots, x_{k-1})$ range over the elements of $\mathcal{W}(n,0) \times P_{n,k}$ (where n is understood).

For each n and for each $k \leq 2^n$, let $T_{n,k}$ be the map from $\mathcal{W}(n,0) \times P_{n,k}$ to $\mathcal{W}(n,k)$ defined by the equation

$$T_{n,k}(R, x_0, \ldots, x_{k-1}) = R/x_0, \ldots, x_{k-1}.$$

Lemma 11 *For each n and each $k \leq 2^n$, $T_{n,k}$ is an $a_{n,k}$-measure scaling map.*

Thus, if the two measure spaces $\mathcal{W}(n,0) \times P_{n,k}$ and $\mathcal{W}(n,k)$ where normalized, then $T_{n,k}$ would be measure preserving.

Proof. Fix n and k. Fix an arbitrary $R_0 \in \mathcal{W}(n,k)$ and let $\{ x'_0, \ldots, x'_{k-1} \} = (\xi^{R_0})^{-1}(\mathbf{0}^n)$. Then

$$T_{n,k}^{-1}(R_0) \;=\; \left\{ (R, x_0, \ldots, x_{k-1}) \;:\; \begin{array}{l} R \sim_{x_0,\ldots,x_{k-1}} R_0 \;\text{ and} \\ \mathbf{0}^n \notin \xi^R(\{ x_0, \ldots, x_{k-1} \}) \;\text{ and} \\ \{ x_0, \ldots, x_{k-1} \} = \{ x'_0, \ldots, x'_{k-1} \} \end{array} \right\}.$$

This set is easily seen to have cardinality $k!(2^n - 1)^k$. Hence, $T_{n,k}$ is onto and $k!(2^n - 1)^k$ to 1.

For the moment let us pretend that there are only m many elements in Σ^N. (In fact, take $m = 2^{n2^n}$.) Thus, each $R \in \Sigma^N$ has weight m^{-1}. Then, each point $R_0 \in \mathcal{W}(n,k)$ of mass m^{-1} is mapped to by $k!(2^n - 1)^k$ many points in $\mathcal{W}(n,0) \times P_{n,k}$, each of mass $(2^n \cdot (2^n - 1) \cdot \ldots \cdot (2^n - k + 1))^{-1}$. Hence, each point of m^{-1} mass is mapped to by $a_{n,k}^{-1} \cdot m^{-1}$ much mass. Therefore, $T_{n,k}$ is $a_{n,k}$-measure scaling.

We can justify the reasoning in the above paragraph as follows. Factor Σ^N into $\Sigma^{n2^n} \times \Sigma^N$, where each $R \in \Sigma^N$ corresponds to (r, R') and where $r \in \Sigma^{n2^n}$ is the subsequence of $n2^n$ bits of R that determine ξ^R on Σ^n and $R' \in \Sigma^N$ is the sequence that results from omitting the initial subsequence r from R. Since all the bits concerned with ξ^R on Σ^n and with $T_{n,k}$ are among the r's and since

$$\mu(\mathcal{A}) \;=\; \int_{\Sigma^N} \int_{\Sigma^{n2^n}} 1_{\mathcal{A}}(r, R')\, dr\, dR' \;=\; \int_{\Sigma^N} \sum_{r \in \Sigma^{n2^n}} \frac{1_{\mathcal{A}}(r, R')}{2^{n2^n}}\, dR',$$

we can reduce measure computations to simple counting as in the above paragraph. $\square$

The Main Argument

We now have all the tools at hand to prove:

Theorem 12 *Relative to a random oracle R,* $\mathrm{UP}^R \neq \mathrm{NP}^R$.

Proof. Let M range over nondeterministic oracle Turing machines that have polynomial-bounded run times that are independent of their oracle. By Lemma 8, to establish the theorem it suffices to prove that there is an $a > 0$ such that for all M, we have $\mu(\{\, R : M^R \text{ fails to UP-accept } L^R \,\}) \geq a$. So fix an M and let $p(\cdot)$ be a polynomial that bounds its run-time on all oracles. For each n and k, define

$$\begin{aligned}
\mathcal{A}(n) &= \left\{\, R : M^R(\mathbf{0}^n) = L^R(\mathbf{0}^n) \,\right\}, \\
\mathcal{A}(n,k) &= \mathcal{A}(n) \cap \mathcal{W}(n,k), \\
\mathcal{D}(n,k) &= \mathcal{W}(n,k) - \mathcal{A}(n,k), \text{ and} \\
\mathcal{M}(n,k) &= \left\{ R \in \mathcal{W}(n,k) \;:\; \begin{array}{l} M^R \text{ on input } \mathbf{0}^n \text{ has at least} \\ k \text{ accepting computations} \end{array} \right\}.
\end{aligned}$$

("$\mathcal{A}$" for agree, "$\mathcal{D}$" for disagree, and "$\mathcal{M}$" for multiple.)

The heart of the argument is the following curious looking lemma.

Lemma 13 *For all n,*

$$\mu(\mathcal{D}(n,0)) + \mu(\mathcal{D}(n,1)) + \mu(\mathcal{M}(n,2)) \;\geq\; \frac{w_{n,0}}{2} - \frac{p(n)}{2^n - 1} - O(\tfrac{1}{2^n}).$$

Before proving the lemma, we show how to use it to establish the theorem. Since $\left(w_{n,0}/2 - p(n)/(2^n-1)\right) \to 1/(2\cdot e)$ as $n \to \infty$ it follows from the lemma that

$$\liminf_{n\to\infty} \mu(\mathcal{D}(n,0)) + \mu(\mathcal{D}(n,1)) + \mu(\mathcal{M}(n,2)) \;\geq\; \frac{1}{2\cdot e}.$$

Since $\mathcal{D}(n,0)$, $\mathcal{D}(n,1)$, and $\mathcal{M}(n,2)$ are pairwise disjoint and M fails to UP-accept L^R on each of these set, we have that

$$\mu(\{\, R : M^R \text{ fails to UP-accept } L^R \,\}) \;\geq\; \frac{1}{2\cdot e}.$$

Since the choice of M was arbitrary, by Lemma 8 this inequality implies the theorem. It remains to show Lemma 13.

The Proof of Lemma 13

Fix n. We obtain our lower bound on $\mu(\mathcal{M}(n,2)) + \mu(\mathcal{D}(n,0)) + \mu(\mathcal{D}(n,1))$ by finding bounds on the measures of a number of other sets. The idea is to gather enough information about the behavior of M on $\mathcal{W}(n,0)$ and $\mathcal{W}(n,1)$

to be able to deduce something of the behavior of M on $\mathcal{W}(n,2)$. All of these "behaviors" manifest themselves as the measure of various sets.

In what follows we often use the following simple version of the principle of inclusion-exclusion: if $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{U}$ are measurable sets with $\mathcal{S}_0 \cup \mathcal{S}_1 \subseteq \mathcal{U}$, then $\mu(\mathcal{S}_0 \cap \mathcal{S}_1) \geq \mu(\mathcal{S}_0) + \mu(\mathcal{S}_1) - \mu(\mathcal{U})$.

Step 1: Estimating $\mu_{n,1}(\mathcal{X})$. Define

$$\mathcal{X} \;=\; \left\{ (R,x) :\ \mathbf{0}^n \notin L^R, \quad M^R(\mathbf{0}^n) \text{ rejects, and } M^{R/x}(\mathbf{0}^n) \text{ accepts} \right\}.$$

We claim

$$\mu_{n,1}(\mathcal{X}) \;\geq\; w_{n,0} - \mu(\mathcal{D}(n,0)) - \mu(\mathcal{D}(n,1)) + O(\tfrac{1}{2^n}). \tag{5}$$

To show this claim we first observe that $\mathcal{X} = (\mathcal{A}(n,0)\times\Sigma^n) \;\bigcap\; T_{n,1}^{-1}(\mathcal{A}(n,1))$. Since $\mathcal{A}(n,0)\times\Sigma^n$ and $T_{n,1}^{-1}(\mathcal{A}(n,1))$ are both subsets of $\mathcal{W}(n,0)\times\Sigma^n$, we have

$$\mu_{n,1}(\mathcal{X}) \;\geq\; \mu_{n,1}(\mathcal{A}(n,0)\times\Sigma^n) + \mu_{n,1}(T_{n,1}^{-1}(\mathcal{A}(n,1)) - \mu_{n,1}(\mathcal{W}(n,0)\times\Sigma^n).$$

We also observe the following:

$$\begin{aligned}
\mu_{n,1}(\mathcal{A}(n,0)\times\Sigma^n) &= \mu(\mathcal{A}(n,0)) \;=\; w_{n,0} - \mu(\mathcal{D}(n,0)).\\
\mu_{n,1}(T_{n,1}^{-1}(\mathcal{A}(n,1))) &= a_{n,1}^{-1}\cdot\mu(\mathcal{A}(n,1))\\
&= a_{n,1}^{-1}\cdot(w_{n,1} - \mu(\mathcal{D}(n,1)))\\
&= w_{n,0} - a_{n,1}^{-1}\cdot\mu(\mathcal{D}(n,1))\\
&= w_{n,0} - \mu(\mathcal{D}(n,1)) + O(\tfrac{1}{2^n}).\\
\mu_{n,1}(\mathcal{W}(n,0)\times\Sigma^n) &= \mu(\mathcal{W}(n,0)) \;=\; w_{n,0}.
\end{aligned}$$

Hence,

$$\begin{aligned}
\mu(\mathcal{X}) &\geq w_{n,0} - \mu(\mathcal{D}(n,0)) \;+\; w_{n,0} - \mu(\mathcal{D}(n,1)) + O(\tfrac{1}{2^n}) \;-\; w_{n,0}\\
&\geq w_{n,0} - \mu(\mathcal{D}(n,0)) - \mu(\mathcal{D}(n,1)) + O(\tfrac{1}{2^n})
\end{aligned}$$

as claimed.

Step 2: Estimating $\mu_{n,2}(\mathcal{Y})$. Define

$$\begin{aligned}\mathcal{Y} &= \Big\{ (R,x,y) : (R,x) \in \mathcal{X} \ \& \ (R,y) \in \mathcal{X} \Big\} \\ &= \left\{ (R,x,y) \ : \ \begin{array}{l} \mathbf{0}^n \notin L^R,\ M^R(\mathbf{0}^n) \text{ rejects, \& both} \\ M^{R/x}(\mathbf{0}^n) \text{ and } M^{R/y}(\mathbf{0}^n) \text{ accept} \end{array} \right\}.\end{aligned}$$

Since $\mathcal{Y} \subseteq \mathcal{W}(n,0) \times P_{n,k}$, we have

$$\begin{aligned}\mu_{n,2}(\mathcal{Y}) &\geq \mu_{n,2}(\{\,(R,x,y) : (R,x) \in \mathcal{X}\,\}) \\ &\qquad\qquad +\mu_{n,2}(\{\,(R,x,y) : (R,y) \in \mathcal{X}\,\}) \\ &\qquad\qquad -\mu_{n,2}(\mathcal{W}(n,0) \times P_{n,k}) \\ &= \mu_{n,1}(\mathcal{X}) + \mu_{n,1}(\mathcal{X}) - w_{n,0} \\ &\geq w_{n,0} - 2\mu(\mathcal{D}(n,0)) - 2\mu(\mathcal{D}(n,1)) - O(\tfrac{1}{2^n}) \qquad \text{(by Eqn. 5).}\end{aligned}$$

Step 3: Estimating $\mu_{n,2}(\mathcal{Y}')$. Define

$$\begin{aligned}\mathcal{I}_0 &= \Big\{ (R,x,y) \in \mathcal{Y} \ : M \text{ on } (R/x, \mathbf{0}^n) \text{ interrogates } y \Big\}, \\ \mathcal{I}_1 &= \Big\{ (R,x,y) \in \mathcal{Y} \ : M \text{ on } (R/y, \mathbf{0}^n) \text{ interrogates } x \Big\}, \text{ and} \\ \mathcal{Y}' &= \mathcal{Y} - \mathcal{I}_0 - \mathcal{I}_1 \\ &= \left\{ (R,x,y) \ : \ \begin{array}{l} \text{(i) } \mathbf{0}^n \notin L^R,\ M^R(\mathbf{0}^n) \text{ rejects, \& both } M^{R/x}(\mathbf{0}^n) \\ \qquad \text{and } M^{R/y}(\mathbf{0}^n) \text{ accept,} \\ \text{(ii) } M \text{ on } (R/x, \mathbf{0}^n) \text{ does not interrogate } y \text{, and} \\ \text{(iii) } M \text{ on } (R/y, \mathbf{0}^n) \text{ does not interrogate } x \end{array} \right\}.\end{aligned}$$

To obtain a lower bound on $\mu(\mathcal{Y}')$, we find upper bounds on the $\mu_{n,2}(\mathcal{I}_i)$'s. We consider $\mathcal{I}_0$ first. For each $R \in \mathcal{W}(n,0)$ and $x \in \Sigma^n$, if $M^{R/x}(\mathbf{0}^n)$ accepts, then by Lemma 10(b), $\|\{\, y \in \Sigma^n \ : \ M \text{ on } (R/x, \mathbf{0}^n) \text{ interrogates } y \,\}\| \leq p(n)$. Hence, for each $R \in \mathcal{W}(n,0)$,

$$\left\| \left\{ (x,y) \in P_{n,k} \ : \ \begin{array}{l} M^{R/x}(\mathbf{0}^n) \text{ accepts \& } M \text{ on} \\ (R/x, \mathbf{0}^n) \text{ interrogates } y \end{array} \right\} \right\| \ \leq \ p(n) \cdot 2^n.$$

Thus, we obtain

$$\mu_{n,2}(\mathcal{I}_0) \ \leq \ \frac{p(n) \cdot 2^n}{2^n \cdot (2^n - 1)} \ \leq \ \frac{p(n)}{2^n - 1}.$$

Similarly, $\mu_{n,2}(\mathcal{I}_1) \leq p(n)/(2^n - 1)$. Therefore, by the lower bound on $\mu(\mathcal{Y})$ and the upper bounds on $\mu_{n,2}(\mathcal{I}_0)$ and $\mu_{n,2}(\mathcal{I}_1)$ we have

$$\mu_{n,2}(\mathcal{Y}') \;\geq\; w_{n,0} - 2\mu(\mathcal{D}(n,0)) - 2\mu(\mathcal{D}(n,1)) - \frac{2 \cdot p(n)}{2^n - 1} - O(\tfrac{1}{2^n}).$$

Last Step: Estimating $\mu(\mathcal{M}(n,2))$. It is easy to check that $\mathcal{Y}' = T_{n,2}^{-1} \circ T_{n,2}(\mathcal{Y}')$. Hence,

$$\begin{aligned}
&\mu(T_{n,2}(\mathcal{Y}')) \\
&\quad= a_{n,2} \cdot \mu_{n,2}(T_{n,2}^{-1}(T_{n,2}(\mathcal{Y}'))) \\
&\quad= a_{n,2} \cdot \mu_{n,2}(\mathcal{Y}') \\
&\quad\geq a_{n,2} \cdot \left(w_{n,0} - 2\mu(\mathcal{D}(n,0)) - 2\mu(\mathcal{D}(n,1)) - \frac{2 \cdot p(n)}{2^n - 1} - O(\tfrac{1}{2^n}) \right) \\
&\quad\geq (\tfrac{1}{2} + O(\tfrac{1}{2^n})) \cdot \left(w_{n,0} - 2\mu(\mathcal{D}(n,0)) - 2\mu(\mathcal{D}(n,1)) - \frac{2 \cdot p(n)}{2^n - 1} - O(\tfrac{1}{2^n}) \right) \\
&\quad\geq \tfrac{1}{2} w_{n,0} - \mu(\mathcal{D}(n,0)) - \mu(\mathcal{D}(n,1)) - \frac{p(n)}{2^n - 1} - O(\tfrac{1}{2^n}).
\end{aligned}$$

Also, it follows from the definitions of $\mathcal{M}(n,2)$, $\mathcal{Y}'$, and $T_{n,2}$ that $T_{n,2}(\mathcal{Y}') \subseteq \mathcal{M}(n,2)$. Therefore, the lemma follows. $\square$ **Lemma 13**

The proof of Theorem 12 is thus complete.

5 Random Oracles and the Output-Multiplicity Hierarchy

We now show that the output-multiplicity hierarchy is infinite relative to a random oracle.

Theorem 14 *Relative to a random oracle, for all $k > 1$,* $\mathrm{NP}k\mathrm{V} \not\subseteq_c \mathrm{NP}(k-1)\mathrm{V}$.

Since the intersection of countably many sets of measure 1 is itself a set of measure 1, it suffices to prove:

Theorem 15 *For all $k > 1$, relative to a random oracle,* $\mathrm{NP}k\mathrm{V} \not\subseteq_c \mathrm{NP}(k-1)\mathrm{V}$.

The rest of the section is devoted to proving Theorem 15. The proof proceeds as follows. Fix k. In place of the ξ^R function of the last section, we introduce a partial function $f^R\colon \mathbf{0}^* \to \Sigma^*$ as follows. Let s_i be the $(i+1)^{st}$ binary string of length $\lceil \log k \rceil$, and for each $i < k$ and $y \in \Sigma^*$, let $tag_i(y)$ be the string $ys_i\mathbf{1}$. Then, for each R, i, and n, we define

$$f^R(\mathbf{0}^n) \mapsto i, \quad \text{if } i < k \text{ and, for some } y \in \Sigma^n,\ \ tag_i(y)\mathbf{0}^j \notin R \text{ for each } j < n.$$

Clearly, for each oracle R, f^R can be computed by a k-valued NP^R transducer. We will show that the collection of all R for which there is a $(k-1)$-valued NP^R transducer that computes a refinement of f^R is a set of measure 0. We first note the following lemma. Let M range over relativized, nondeterministic TM transducers that have polynomial bounded run times which are *independent* of their oracle.

Lemma 16 *Suppose that there is an $a > 0$ such that, for all M, $\mu(\{\, R : M^R$ fails to be $(k-1)$-valued transducer that computes a refinement of $f^R \,\}) \geq a$. Then, $\{\, R : \mathrm{NP}k\mathrm{V}^R \not\subseteq_c \mathrm{NP}(k-1)\mathrm{V}^R \,\}$ has measure 1.*

Proof Sketch: First adjust Lemma 7 so that the oracle-dependent languages B, C, and D are respectively replaced by oracle-dependent, multivalued functions g_B, g_C, and g_D and change "$\max(B^{\sigma \triangleleft R} \triangle B^R) < n_\sigma$" in item 4 to "$\max(\{\, x : set\text{-}g_B^{\sigma\triangleleft R}(x) \neq set\text{-}g_B^R(x) \,\}) < n_\sigma$." Minor changes in the proof of Lemma 7 suffice to obtain the revised version of the lemma. Now note the hypotheses of the revised Lemma 7 are satisfied when $g_B^R = f^R$ and $\mathbf{A}(i, R, g^R) \equiv$ nondeterministic transducer i is $(k-1)$-valued and computes a refinement of g^R. Thus, a simple argument analogous to the one given for Lemma 8 suffices to obtain the present lemma.

So, by this lemma, it suffices to understand the difficulties encountered by an M that "tries" to be a $(k-1)$-valued transducer computing a refinement of f^R. The nature of these difficulties is that success in accurately computing f^R on a significant part of Σ^N entails that M^R is k-valued on another significant part of Σ^N. To understand this balance between successes and failures, we need to understand the structure that f^R imposes on Σ^N and computations over $\Sigma^N \times N$. Towards this end, we introduce the following definitions.

For each n and i and oracle R, define

$$\begin{aligned} witnesses(n,i,R) &= \{\, y \in \Sigma^n \;:\; \text{for each } j < n,\;\; tag_i(y)\mathbf{0}^j \notin R \,\} \\ &= \{\, y \in \Sigma^n \;:\; y \text{ witnesses } f^R(\mathbf{0}^n) \mapsto i \,\}. \end{aligned}$$

For each a, i, and n, define

$$\mathcal{W}(n,a,i) \;=\; \{\, R \;:\; \|witnesses(n,i,R)\| = a \,\}.$$

The measure of $\mathcal{W}(n,a,i)$ is equal to the probability that a set R has exactly a-many witnesses to $f^R(\mathbf{0}^n) \mapsto i$. The probability that a given string $y \in \Sigma^n$ is such a witness is 2^{-n}. So, by counting the number of ways in which there can be exactly a-many witnesses in Σ^n, we obtain

$$\mu(\mathcal{W}(n,a,i)) \;=\; \binom{2^n}{a} \cdot \left(\frac{1}{2^n}\right)^a \cdot \left(1 - \frac{1}{2^n}\right)^{2^n - a}. \tag{6}$$

Observe that the right-hand side of the above equation has no dependence on i. For each $a \leq k$, we let $w_{n,a}$ denote the right-hand side of equation 6. We note that for each n, $w_{n,0} = w_{n,1}$ and, for each $i < k$ and each $a \leq k$,

$$\lim_{n\to\infty} w_{n,a} \;=\; \frac{1}{a! \cdot e}. \tag{7}$$

Now, from the $\mathcal{W}(n,a,i)$'s we build the following sets:

$$\mathcal{W}_\uparrow(n) \;=\; \{\, R \;:\; f^R(\mathbf{0}^n)\!\uparrow \,\}$$

$$
\begin{aligned}
&= \cap_{j<k}\mathcal{W}(n,0,j). \\
\mathcal{W}_\star(n) &= \left\{ R \;\middle|\; \begin{array}{l} f^R(\mathbf{0}^n) \text{ is } k\text{-valued and there is a} \\ \text{unique witness for each output value} \end{array} \right\} \\
&= \cap_{j<k}\mathcal{W}(n,1,j). \\
\mathcal{W}_i(n) &= \left\{ R \;\middle|\; \begin{array}{l} f^R(\mathbf{0}^n) \text{ has just } i \text{ as its value and} \\ \text{there is a unique witness to this} \end{array} \right\} \\
&= \cap_{j<i}\mathcal{W}(n,0,j) \bigcap \mathcal{W}(n,1,i) \bigcap \cap_{j>i}\mathcal{W}(n,0,j),
\end{aligned}
$$

where $i < k$. Since for all a_1, a_2, i, and j with i and j distinct, the events $\mathcal{W}(n, a_1, i)$ and $\mathcal{W}(n, a_2, j)$ are independent, we conclude that

$$
\begin{aligned}
\mu(\mathcal{W}_\uparrow(n)) &= w_{n,0}^k, \text{ and} && (8) \\
\mu(\mathcal{W}_i(n)) &= w_{n,1} \cdot w_{n,0}^{k-1} = w_{n,0}^k, \quad \text{for each } i < k. && (9)
\end{aligned}
$$

Recall that *set*-$M(x)$ is the set of output values of M on input x. Given a transducer M, a multivalued function f, and a string x, if *set*-$M(x) =$ *set*-$f(x)$, then we write $M(x) \simeq_c f(x)$.

As we noted before, to prove the theorem is suffices to show that there is an $a > 0$ such that, for all M, $\mu(\{ R : M^R$ fails to be a $(k-1)$-valued transducer that computes a refinement of $f^R \}) \geq a$. So, fix an M and let $p(\cdot)$ be a polynomial function that bounds M's run-time on all oracles. We define the following sets for each $n \in N$.

$$
\begin{aligned}
\mathcal{C}(n) &= \{ R \;:\; M^R(\mathbf{0}^n) \simeq_c f^R(\mathbf{0}^n) \}. \\
\mathcal{C}_\uparrow(n) &= \mathcal{C}(n) \cap \mathcal{W}_\uparrow(n). \\
\mathcal{C}_i(n) &= \mathcal{C}(n) \cap \mathcal{W}_i(n). \\
\mathcal{M}(n) &= \{ R \in \mathcal{W}_\star(n) \;:\; \|\textit{set-}M^R(\mathbf{0}^n)\| = k \}.
\end{aligned}
$$

Intuitively, $\mathcal{C}(n)$ is the set of oracles where M correctly computes f; $\mathcal{C}_\uparrow(n)$ is the subset of $\mathcal{C}(n)$ in which $f^R(\mathbf{0}^n)$ is undefined; $\mathcal{C}_i(n)$ is the subset of $\mathcal{C}(n)$ in which $f^R(\mathbf{0}^n)$ has the unique value i and has a unique witness for this; and $\mathcal{M}(n)$ is the subset of $\mathcal{W}_\star(n)$ where $M^R(\mathbf{0}^n)$ happens to be k-valued too. The following key lemma gives a lower bound on $\mu(\mathcal{M}(n))$ in terms of $\mu(\mathcal{C}_\uparrow(n))$ and the $\mu(\mathcal{C}_i(n))$'s.

Lemma 17 *For each n, we have that*

$$\mu(\mathcal{M}(n)) \;\geq\; k \cdot \mu(\mathcal{C}_\uparrow(n)) + \left(\sum_{i<k} \mu(\mathcal{C}_i(n))\right) - (k+1) \cdot w_{n,0} - k^2 \cdot p(n) \cdot 2^{-n}.$$

Before proving the lemma, we show how to use it to establish Theorem 15.

For each n and $i < k$, define $\mathcal{I}_\uparrow(n) = \mathcal{W}_\uparrow(n) - \mathcal{C}_\uparrow(n)$ and $\mathcal{I}_i(n) = \mathcal{W}_i(n) - \mathcal{C}_i(n)$. ("$\mathcal{I}$" for incorrect.) By Lemma 17, we have

$$\begin{aligned}
\mu(\mathcal{M}(n)) \;&\geq\; k \cdot \mu(\mathcal{C}_\uparrow(n)) + \left(\sum_{i<k} \mu(\mathcal{C}_i(n))\right) - (k+1) \cdot w^k_{n,0} - k^2 \cdot p(n) \cdot 2^{-n} \\
&=\; k \cdot (\mu(\mathcal{W}_\uparrow(n)) - \mu(\mathcal{I}_\uparrow(n))) + \left(\sum_{i<k} \mu(\mathcal{W}_i(n)) - \mu(\mathcal{I}_i(n))\right) \\
&\qquad - (k+1) \cdot w^k_{n,0} - k^2 \cdot p(n) \cdot 2^{-n}.
\end{aligned}$$

By some algebra we obtain

$$\begin{aligned}
&\mu(\mathcal{M}(n)) + \mu(\mathcal{I}_\uparrow(n)) + \sum_{i<k} \mu(\mathcal{I}_i(n)) \\
&\quad\geq\; k \cdot \mu(\mathcal{W}_\uparrow(n)) + \left(\sum_{i<k} \mu(\mathcal{W}_i(n))\right) - (k+1) \cdot w^k_{n,0} - k^2 \cdot p(n) \cdot 2^{-n} \\
&\quad=\; (k-1) \cdot w^k_{n,0} - k^2 \cdot p(n) \cdot 2^{-n} \qquad \text{(by Equations 8 and 9).}
\end{aligned}$$

Thus, by Equation 7,

$$\liminf_{n\to\infty} \left(\mu(\mathcal{M}(n)) + \mu(\mathcal{I}_\uparrow(n)) + \sum_{i<k} \mu(\mathcal{I}_i(n))\right) \;\geq\; (k-1) \cdot e^{-k}.$$

Since $\mathcal{I}_\uparrow(n)$, $\mathcal{I}_0(n), \ldots, \mathcal{I}_{k-1}(n)$, and $\mathcal{M}(n)$ are pairwise disjoint and since M behaves "incorrectly" on each of them, we have by the above inequality that $\mu(\{\, R : M^R$ fails to be $(k-1)$-valued transducer that computes a refinement of $f^R \,\}) \geq (k-1) \cdot e^{-k}$. Since the choice of M was arbitrary, by Lemma 16, we have Theorem 15.

It remains to show Lemma 17.

The Proof of Lemma 17

Fix n. We establish our lower bound on $\mu(\mathcal{M}(n))$ by obtaining some information about how M behaves on certain other sets. For the remainder of this proof, we assume that each free occurrence of i is implicitly quantified as "for each $i < k$".

For each oracle R and $z \in \Sigma^n$, define

$$R/z \;=\; R - \{\, z\mathbf{0}^r \;:\; r < n \,\}.$$

Observe that $f^{R/tag_i(x)}(\mathbf{0}^n) \mapsto i$. Also, for each R, and $x_0, x_1, \ldots x_{k-1} \in \Sigma^n$, define

$$R/[x_0, x_1 \ldots, x_{k-1}] \;=\; R - \{\, tag_i(x_i)\mathbf{0}^r \;:\; i < k \text{ and } r < n \,\}.$$

Observe that $set\text{-}f^{R/[x_0,\ldots,x_{k-1}]}(\mathbf{0}^n) = \{\, 0, \ldots, k-1 \,\}$.

Step 1: Estimating the $\mu(\mathcal{A}(i))$'s. We view Σ^n as a measure space under the uniform, normalized counting measure μ_0. Thus, for all $x \in \Sigma^n$, $\mu_0(x) = 2^{-n}$. $\mathcal{W}_\uparrow(n) \times \Sigma^n$ is thus a measure space under the product measure $\mu_1 = \mu \times \mu_0$. (Where μ in $\mu \times \mu_0$ is understood as the measure μ restricted to the subspace $\mathcal{W}_\uparrow(n)$.) For each i, define

$$\begin{aligned}
\mathcal{A}(i) &= \{\, (R,x) \;:\; R \in \mathcal{C}_\uparrow(n) \text{ and } R/tag_i(x) \in \mathcal{C}_i(n) \,\} \\
&= \left\{ (R,x) \;:\; \begin{array}{l} \text{(i) } M^R(\mathbf{0}^n) \simeq_c f^R(\mathbf{0}^n)\uparrow \quad \text{and} \\ \text{(ii) } M^{R/tag_i(x)}(\mathbf{0}^n) \simeq_c f^{R/tag_i(x)}(\mathbf{0}^n) \mapsto i \end{array} \right\}.
\end{aligned}$$

Our goal in this step is to establish

$$\mu_1(\mathcal{A}(i)) \;\geq\; \mu(\mathcal{C}_\uparrow(n)) + \mu(\mathcal{C}_i(n)) - w_{n,0}. \tag{10}$$

To help prove this, we introduce the map $T_i\colon \mathcal{W}_\uparrow(n) \times \Sigma^n \to \mathcal{W}_i(n)$ defined by the equation $T_i(R,x) = R/tag_i(x)$.

Claim: $T_{n,k}$ is $(2^n/(2^n-1))^k$-measure scaling. *Proof:* T_i is onto and $2^n - 1$ to 1. For the moment let us pretend, as in the proof of Lemma 11, that there only m many elements in Σ^N. Then each $R \in \Sigma^N$ has mass m^{-1}. Then each point $R_0 \in \mathcal{W}_i(n)$ of mass m^{-1} is mapped to by $(2^n - 1)$ many points in $\mathcal{W}_\uparrow(n) \times \Sigma^n$ each of mass $(m \cdot 2^n)^{-1}$. Hence, each point of m^{-1} mass is mapped to by $((2^n-1)/2^n) \cdot m^{-1}$ much mass. Therefore, $T_{n,k}$ is $(2^n/(2^n-1))^k$-measure scaling. (We can justify this reasoning exactly as we did in the proof of Lemma 11.) Thus the claim follows.

Now, by definition of $\mathcal{A}(i)$, we have

$$\mathcal{A}(i) \;=\; (\, \mathcal{C}_\uparrow(n) \times \Sigma^n \,) \cap \Big(T_i^{-1}(\mathcal{C}_i(n)) \Big).$$

Since the sets $(\mathcal{C}_\uparrow(n) \times \Sigma^n)$ and $(T_i^{-1}(\mathcal{C}_i(n)))$ are both subsets of $\mathcal{W}_\uparrow(n) \times \Sigma^n$, it follows by the principle of inclusion and exclusion that,

$$\mu_1(\mathcal{A}(i)) \;\; \geq \;\; \mu_1(\mathcal{C}_\uparrow(n) \times \Sigma^n) + \mu_1(T_i^{-1}(\mathcal{C}_i(n)) - \mu_1(\mathcal{W}_i(n) \times \Sigma^n). \quad (11)$$

Since, for each i, T_i is measure preserving, we have

$$\begin{array}{rcccl} \mu_1(\mathcal{C}_\uparrow(n) \times \Sigma^n) & = & \mu(\mathcal{C}_\uparrow(n)), & & \\ \mu_1(T_i^{-1}(\mathcal{C}_i(n))) & = & \frac{2^n}{2^n-1} \cdot \mu(\mathcal{C}_i(n)) & \geq & \mu(\mathcal{C}_i(n)), \text{ and} \\ \mu_1(\mathcal{W}_\uparrow(n) \times \Sigma^n) & = & \mu(\mathcal{W}_\uparrow(n)) & = & w_{n,0}. \end{array}$$

Using the above and Equation 11, we obtain Equation 10 as desired.

Step 2: Estimating $\mu(\mathcal{B})$. *Convention:* We write $\vec{x}$ for $x_0, \ldots, x_{k-1}$. The set $(\Sigma^n)^k$ can be viewed as a measure space under the uniform, normalized counting measure μ_* such that, for each $(\vec{x}) \in (\Sigma^n)^k$, $\mu_*(\vec{x}) = 2^{-k \cdot n}$. $\mathcal{W}_\uparrow(n) \times (\Sigma^n)^k$ is a measure space under $\mu_2 = \mu \times \mu_*$. We define

$$\begin{array}{rcl} \mathcal{B}_i & = & \{\, (R, \vec{x}) \;:\; (R, x_i) \in \mathcal{A}(i) \,\}, \text{ and} \\ \mathcal{B} & = & \{\, (R, \vec{x}) \mid \text{for all i, } (R, x_i) \in \mathcal{A}(i) \,\} \;\; = \;\; \cap_{i<k} \mathcal{B}_i \\ & = & \left\{ (R, \vec{x}) \;:\; \begin{array}{l} \text{(i) } \textit{set-}M^R(\mathbf{0}^n) = \textit{set-}f^R(\mathbf{0}^n) = \emptyset, \quad \text{and} \\ \text{(ii) for all } i, \;\; M^{R/tag_i(x)}(\mathbf{0}^n) \simeq_c f^{R/tag_i(x)}(\mathbf{0}^n) \mapsto i \end{array} \right\}. \end{array}$$

Clearly, for each i, $\mathcal{B}_i \subseteq \mathcal{W}_\uparrow(n) \times (\Sigma^n)^k$ and $\mu_2(\mathcal{B}_i) = \mu_1(\mathcal{A}(i))$. Using the principle of inclusion and exclusion once more, we obtain

$$\mu_2(\mathcal{B}) \;\; \geq \;\; \Big(\sum_{i<k} \mu_1(\mathcal{A}(i)) \Big) - \mu(\mathcal{W}_\uparrow(n)).$$

On substituting from Equation 10, we have the following estimate:

$$\mu_2(\mathcal{B}) \;\; \geq \;\; k \cdot \mu(\mathcal{C}_\uparrow(n)) + \Big(\sum_{i<k} \mu(\mathcal{C}_i(n)) \Big) - k \cdot w_{n,0}^k - w_{n.0}^k. \qquad (12)$$

Step 3: Estimating $\mu(\mathcal{B}')$. As in Definition 9(d), we say that M^R *on* y *interrogates* z if, in *every* computation of M with oracle R and input y, the machine queries R about some string of the form $z\mathbf{0}^j$ with $j < k$. So, if M^R on $\mathbf{0}^n$ interrogates $tag_i(x)$, then every computation of M^R on $\mathbf{0}^n$ knows something about whether $f^R(\mathbf{0}^n) \mapsto i$. We are interested in $\mathcal{B}'$, a subset of $\mathcal{B}$ obtained by removing from $\mathcal{B}$ those $(R, \vec{x})$ where an interrogation of some $tag_i(x)$ occurs. That is, $\mathcal{B}' = \mathcal{B} - \cup_{j<k} \mathcal{J}_i$, where each $i < k$,

$$\mathcal{J}_i \quad = \quad \left\{ (R, \vec{x}) \in \mathcal{B} \;\middle|\; \begin{array}{l} M^{R/tag_i(x)} \text{ on } \mathbf{0}^n \text{ accepts and interrogates} \\ \text{some } tag_j(x_j) \text{ with } j < k \text{ and } j \neq i \end{array} \right\}.$$

So, $\mathcal{B}' = \mathcal{B} - \cup_{j<k} \mathcal{J}_i =$

$$\left((R, \vec{x}) \ : \ \begin{array}{l} \text{(i) } M^R(\mathbf{0}^n) \simeq_c f^R(\mathbf{0}^n)\uparrow, \\ \text{(ii) For all } i,\ M^{R/tag_i(x)}(\mathbf{0}^n) \simeq_c f^{R/tag_i(x)}(\mathbf{0}^n) \mapsto i, \text{ and} \\ \text{(iii) For all } i,\ M^{R/tag_i(x)}(\mathbf{0}^n) \text{ does not interrogate any} \\ \qquad tag_j(x_j) \text{ with } j < k \text{ and } j \neq i \end{array} \right).$$

To establish a lower bound on the measure of $\mathcal{B}'$, we first obtain upper bounds on the measure of the $\mathcal{J}_i$'s. For each $x \in \Sigma^n$ and $R \in \mathcal{W}_\uparrow(n)$, if $M^{R/tag_i(x)}(\mathbf{0}^n)$ accepts, then Lemma 10(b) yields $\|\{\, y \in \Sigma^n \mid M^{R/tag_i(x)}(\mathbf{0}^n) \text{ interrogates } y \,\}\| \leq p(n)$. So, for each $R \in \mathcal{W}_\uparrow(n)$ and each i, a simple counting argument shows

$$\left\| \left\{ (\vec{x}) \in (\Sigma^n)^k \;\middle|\; \begin{array}{l} M^{R/tag_i(x)} \text{ on } \mathbf{0}^n \text{ accepts \& interrogates} \\ \text{some } tag_j(x_j) \text{ with } j < k \text{ and } j \neq i \end{array} \right\} \right\|$$

$$\leq \quad (k-1) \cdot p(n) \cdot 2^{(k-1) \cdot n}.$$

Thus, for all i,

$$\mu_2(\mathcal{J}_i) \quad \leq \quad \frac{(k-1) \cdot p(n) \cdot 2^{(k-1) \cdot n}}{2^{k \cdot n}} \quad \leq \quad k \cdot p(n) \cdot 2^{-n}.$$

Since $\mathcal{B}' = \mathcal{B} - \cup_{i<k} \mathcal{J}_i$, by the lower bound of Equation 12 and the above upper bounds on the $\mu(\mathcal{J}_i)$'s, we obtain

$$\begin{aligned} \mu_2(\mathcal{B}') \quad &\geq \quad k \cdot \mu(\mathcal{C}_\uparrow(n)) + \left(\textstyle\sum_{i<k} \mu(\mathcal{C}_i(n)) \right) - (k+1) \cdot w_{n,0} - \textstyle\sum_{i<k} \mu_2(\mathcal{J}_i) \\ &\geq \quad k \cdot \mu(\mathcal{C}_\uparrow(n)) + \left(\textstyle\sum_{i<k} \mu(\mathcal{C}_i(n)) \right) - (k+1) \cdot w_{n,0} - k^2 \cdot p(n) \cdot 2^{-n}. \end{aligned}$$

Last Step: Estimating $\mu(\mathcal{M}(n))$. Let $G \colon \mathcal{W}_\uparrow(n) \times (\Sigma^n)^k \to \mathcal{W}_\star(n)$ be the map defined by $G(R, \vec{x}) = R/[\vec{x}]$. It is easy to see that G is onto and $(2^n - 1)^k$ to 1. By an argument similar to the one for T_i above, we can establish that G is $(2^n/(2^n-1))^k$-measure scaling. Hence,

$\mu(G(\mathcal{B}'))$

$$
\begin{aligned}
&= \left(\frac{2^n}{2^n-1}\right)^k \cdot \mu_2(G^{-1}(G(\mathcal{B}'))) \\
&\geq \left(\frac{2^n}{2^n-1}\right)^k \cdot \mu_2(\mathcal{B}') \qquad\qquad (\text{since } \mathcal{B}' \subseteq G^{-1}(G(\mathcal{B}'))) \\
&\geq \mu_2(\mathcal{B}') \\
&\geq k \cdot \mu(\mathcal{C}_{\uparrow}(n)) + \left(\textstyle\sum_{j<k} \mu(\mathcal{C}_j(n))\right) - (k+1) \cdot w_{n,0} - k^2 \cdot p(n) \cdot 2^{-n}.
\end{aligned}
$$

Since M^R is k-valued on all oracles in $G(\mathcal{B}')$ and $G(\mathcal{B}') \subseteq \mathcal{W}_\star(n)$, it follows that $G(\mathcal{B}') \subseteq \mathcal{M}(n)$, and so Lemma 16 follows.

The proof of Theorem 15 thus is complete.

6 Remarks and Open Questions

Define UP_k to be the class of all languages in NP that are acceptable by an NP-machine that has at most k accepting computations on every input. One can associate each language $L \in \mathrm{UP}_k$ with the partial function in $\mathrm{NP}k\mathrm{V}_g$ that maps each $x \in L$ to the accepting computations of the UP_k-acceptor for L. For $k \geq 1$, does $\mathrm{UP}_{k+1} = \mathrm{UP}_k$ imply that the polynomial hierarchy collapses? Does $\mathrm{UP} = \mathrm{NP}$ imply that the polynomial hierarchy collapses? The results about function classes seem not to imply anything about the corresponding language classes. The problem is that some strange unambiguous Turing machine might accept SAT whose accepting paths have no connection with the problem of computing satisfying assignments.

Similarly, we have not been able to separate the classes UP_k by a random oracle. The reason why the obvious application of the proof of Theorem 15 fails is that the domain of the k-valued function f^R is not necessarily in UP_k^R It seems difficult to construct an oracle-dependent language that, for almost all oracles, has k witnesses but not $k-1$ withnesses. A Turing machine that, on input x, randomly decides on a subspace of the witness space and then searches for witnesses only in this subspace will frustrate any language that is defined using the function f^R.

In light of the result of Section 3, existence of an oracle relative to which the polynomial hierarchy collapses to P^{NP} while the output-multiplicity hierarchy is strict, is it possible that the result of Section 2 can be improved? Does a collapse of the output-multiplicity hierarchy imply a collapse of the polynomial hierarchy lower than Σ_2^{P}. With regard to this question, let us note that Hemaspaandra *et al.* [HNOS96] showed that $\mathrm{NPMV} \subseteq_c \mathrm{NPSV}$ implies the polynomial hierarchy collapses to $\mathrm{ZPP}^{\mathrm{NP}}$, and our techniques do not seem to obtain even this.

Another related open question is whether a conjecture raised by Even, Selman, and Yacobi [ESY84] holds relative to a random oracle. The conjecture states that every disjoint pair of Turing-complete sets in NP is separable by a set that is not Turing-hard for NP. It is known [ESY84, GS88, Sel94] that this conjecture implies (i) $\mathrm{NP} \neq co\text{-}\mathrm{NP}$, (ii) $\mathrm{NP} \neq \mathrm{UP}$, and $\mathrm{NPMV} \subseteq_c \mathrm{NPSV}$. It has been known that $\mathrm{NP} \neq co\text{-}\mathrm{NP}$ holds relative to random oracle [BG81] and this paper demonstrates that the second and third consequences hold relative to a random oracle.

Finally we raise the following technical question:

> Let $k \geq 1$. Does $\mathrm{NP}(k+1)\mathrm{V} \subseteq_c \mathrm{NP}k\mathrm{V}$ imply for all $m \geq k$, that $\mathrm{NP}m\mathrm{V} \subseteq_c \mathrm{NP}k\mathrm{V}$?

7 Acknowledgments

We would like to thank Lance Fortnow for his quick and useful comments on an early draft of the main proof of Section 3 and Ken Regan for helpful discussions on random oracles.

References

[BG81] C. Bennett and J. Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{Co-NP}^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, February 1981.

[BLS84] R. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM Journal on Computing*, 13(3):461–487, August 1984.

[Dud89] R. Dudley. *Real Analysis*. Wadsworth & Brooks/Cole, 1989.

[ESY84] S. Even, A. Selman, and J. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173.

[FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the Eighth Annual IEEE Conference on Structure in Complexity Theory*, pages 120–131, 1993.

[FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 213–223, 1996.

[FGH+96] S. Fenner, F. Green, S. Homer, A. Selman, T. Thierauf, and H. Vollmer. Complements of multivalued functions. In *Procedgins of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 260–269, 1996.

[FFK96] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. *SIAM Journal on Computing*, 25(1):191–206, 1996.

[FHOS97] S. Fenner, S. Homer, M. Ogihara, and A. Selman. On using oracles that compute values. *SIAM Journal on Computing*, 26(4):1043–1065, 1997.

[FR94] L. Fortnow and J. Rogers. Separability and one-way functions. In D. Du and X. Zhang, editors, *Proceedings of the Fifth International*

Symposium on Algorithms and Computation, volume 834 of Lecture Notes in Computer Science, pages 396–404, Springer, Berlin, 1994.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–355, April 1988.

[Has86] J. Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*. pages 6–20, 1986.

[HNOS96] L. Hemaspaandra, A. Naik, M. Ogihara, and A. Selman. Computing solutions uniquely collapses the polynomial hierarchy. *SIAM Journal on Computing*, 25(4):697–708, August 1996.

[JT95] B. Jenner and J. Torán. The complexity of obtaining solutions for problems in NP. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*. Springer, New York, 1995.

[KMR92] S. Kurtz, S. Mahaney, and J. Royer. Average dependence and random oracles. In *Proceedings of the Seventh Annual IEEE Conference on Structure in Complexity Theory*, pages 306–317, 1992.

[KMR95] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. *Journal of the Association for Computing Machinery*, 42:401–420, 1995.

[Ko83] K. Ko. On self-reducibility and weak P- selectivity. *Journal of Computer and System Sciences*, 26:209–211, 1983.

[Kur83] S. Kurtz. Three theorems about random oracles from computational complexity theory and recursive function theory. Unpublished manuscript, 1983.

[LS86] T. Long and A. Selman. Relativizing complexity classes with cparse oracles. *Journal of the Association for Computing Machinery*, 33(3):618–627, July 1986.

[LS93] L. Longpré and A. Selman. Hard promise problems and nonuniform complexity. *Theoretical Computer Science*, 115(3):277–290, 1993.

[Nai94] A. Naik. *The structural complexity of intractable search functions*. PhD thesis, State University of New York at Buffalo, Buffalo, NY, 1994.

[Ogi96] M. Ogihara. Functions computable with limited access to NP. *Information Processing Letters*, 58:35–38, 1996.

[Oxt80] John C. Oxtoby. *Measure and Category*. Springer-Verlag, New York, 2nd edition, 1980.

[Rog97] J. Rogers. The isomorphism conjecture holds and one-way functions exist relative to an oracle. *Journal of Computer and System Sciences*, 1997. To appear.

[Rud66] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 1966.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[Sel92] A. Selman. A survey of one-way functions in complexity theory. *Mathematical Systems Theory*, 25:203–221, 1992.

[Sel94] A. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.

[Sel96] A. Selman. Much ado about functions. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 198–212, 1996.

[Yao85] A. Yao. Separating the polynomial hierarchy by oracles. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.