Inverting Onto Functions

Stephen A. Fenner* Lance Fortnow† Ashish V. Naik‡ John D. Rogers§

Abstract

We look at the hypothesis that all honest onto polynomial-time computable functions have a polynomial-time computable inverse. We show this hypothesis equivalent to several other complexity conjectures including

- In polynomial time, one can find accepting paths of nondeterministic polynomial-time Turing machines that accept Σ^*.
- Every total multivalued nondeterministic function has a polynomial-time computable refinement.
- In polynomial time, one can compute satisfying assignments for any polynomial-time computable set of satisfiable formulae.
- In polynomial time, one can convert the accepting computations of any nondeterministic Turing machine that accepts SAT to satisfying assignments.

We compare these hypotheses with several other important complexity statements. We also examine the complexity of these statements where we only require a single bit instead of the entire inverse.

1 Introduction

Understanding the power of nondeterminism has been one of the primary goals of research in complexity theory in the past two decades. *One-way functions* are an important tool for studying nondeterministic functions. A polynomial-time computable function f is one-way if it is one-to-one, honest, and cannot be inverted in polynomial time. Grollmann and Selman [GS88] showed that one-way functions exist if and only if P $\neq$ UP. For many-to-one functions, it is easy to see that every polynomial-time computable many-to-one function is invertible if and only if P = NP.

Several of the results on noninvertibility of one-way functions do not restrict the functions to be *onto*, that is, the inverse of a one-way function could be a *partial* function. Grollmann and Selman showed that every one-to-one and *onto* function is invertible if and only if P = UP $\cap$ coUP, and Borodin and Demers [BD76] showed that, if every many-to-one, poly-time computable onto

*Department of Computer Science, University of Southern Maine, Portland, ME 04103. Research supported in part by NSF grant CCR 92-09833. Email: `fenner@usm.maine.edu`

†Computer Science Department, University of Chicago, Chicago, IL 60637. Research supported in part by NSF grant CCR 92-53582. Email: `fortnow@cs.uchicago.edu`

‡Computer Science Department, University of Chicago, Chicago, IL 60637. Research supported in part by NSF grant CCR 92-53582. Email: `naik@cs.uchicago.edu`

§School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, IL 60604. Email: `rogers@cs.depaul.edu`. Work done while at the Computer Science Department, University of Chicago.

function is poly-time invertible, then P = NP $\cap$ coNP. However, these consequences are still weaker than P = NP. Indeed, it is conceivable that every poly-time computable, honest, onto function is invertible in polynomial time, but P $\neq$ NP. However, other than the above results, not much is known about the consequences of assuming that every onto function is polynomial-time invertible.

In this paper we look at the hypothesis that all polynomial-time computable, honest, onto functions are polynomial-time invertible. We show that this proposition is equivalent to several other fundamental propositions in complexity theory. An interesting example is the following assertion: For all NP machines M that accept SAT, there is a polynomial-time procedure that translates an accepting computation of M into a satisfying assignment. Informally, this is equivalent to saying that there is essentially only one nondeterministic algorithm for accepting SAT. If this holds then every many-one reduction between two NP sets can be converted to a "witness-preserving" many-one reduction, which is equivalent to saying that Karp's notion of many-one completeness [Kar72] is equivalent to Levin's notion of "universal search problems" [Lev73]. Some other equivalent propositions are *tautology search* as studied by Impagliazzo and Naor [IN88] and the assertion that total functions in the function class NPMV have refinements in PF [Sel94] (formal definitions are given in Section 2). Because of the robust nature of these hypotheses, we use the notation $\mathbf{Q}$ to denote the property that any or all of the propositions hold.

We also consider a weaker proposition and ask—can we efficiently compute a single bit of an inverse of an onto function? This question is equivalent to the single bit version of all of the other $\mathbf{Q}$ hypotheses. These propositions are also equivalent to the following much studied hypothesis [GS88, FR94]: Every pair of disjoint coNP sets are p-separable (that is, for all disjoint pairs of coNP sets, there exists a p-time computable set that contains one of the two sets and is disjoint from the other one). We use the notation $\mathbf{Q}'$ to represent the property that any or all of these hypotheses are true.

Papadimitriou [Pap94] (see also [BCE$^+$95]) defined the function class TFNP to study the complexity of computing proofs that are always known to exist because of some combinatorial property. TFNP is the class of total functions whose graphs are polynomial-time computable. An interesting question is whether every total function in NPMV_t has a refinement in TFNP. We show that this question is intermediate between $\mathbf{Q}$ and $\mathbf{Q}'$.

Does hypothesis $\mathbf{Q}'$ imply hypothesis $\mathbf{Q}$? This is equivalent to the question: If all 0-1 *valued*, total NPMV functions are computable in poly-time, then is every total NPMV function poly-time computable? Without the totality constraint, the answer to this question is trivially in the affirmative, since either of the hypotheses implies that P = NP. However, since neither $\mathbf{Q}$ nor $\mathbf{Q}'$ are known to be equivalent to P = NP, the equivalence of $\mathbf{Q}$ and $\mathbf{Q}'$ seems to be a harder question. We make progress towards resolving this question in the affirmative and show that, if every 0-1 valued total NP function is computable in poly-time, then for all $k > 0$, every total NP function with at most k-many output values is computable in polynomial time (in symbols, for all $k \geq 0$, $\mathbf{Q}' \Rightarrow \mathrm{NPkV}_t \subseteq_c \mathrm{PF}$). To prove this, we use the technique of "binary search with multivalued oracles" that may be of independent interest.

Finally, we study the relationship of $\mathbf{Q}$ to other well-known complexity hypotheses. It is well-known that if $\mathbf{Q}$ holds, then P = NP $\cap$ coNP [BD76, IN88]. Continuing this line of research, we show that $\mathbf{Q}'$ implies that AM $\cap$ coAM = BPP and that NP $\cap$ coAM = RP. Thus, if $\mathbf{Q}'$ holds, then the graph isomorphism problem is in RP, which is not known to follow by the assumption that P = NP $\cap$ coNP. Next, we study how the assumption that $\mathbf{Q}$ holds affects some well-studied open questions in complexity theory. The first question is whether NP = UP implies that the polynomial hierarchy collapses. While neither hypothesis $\mathbf{Q}$ nor NP = UP are by themselves known to imply to collapse of the polynomial hierarchy, we show that if both $\mathbf{Q}'$ and NP = UP hold, then PH = $\mathrm{ZPP}^{\mathrm{NP}} \subseteq \Sigma_2^{\mathrm{P}}$. Next, we consider the question of whether every paddable 1-degree collapses to a paddable 1-length-increasing degree. We show that if $\mathbf{Q}$ holds, then indeed this is the case. Finally, we list some known relativization results to show that some of our results are optimal with respect to relativizable proof techniques.

In Section 2, we will give some preliminary definitions—in particular, we will define function complexity classes. In Section 3, we will prove the various characterizations of $\mathbf{Q}$ and in Section 4, we give our results about the relationship between $\mathbf{Q}$ and other complexity assertions. In Section 5, we look at the relationship between $\mathbf{Q}$ and $\mathbf{Q}'$. We conclude by listing open questions in Section 6.

2 Preliminaries

In this section, we will set down notation that will be used throughout the paper. All languages and functions are defined over strings in the alphabet $\Sigma = \{0,1\}$, the set of all strings is denoted by Σ^*. We will let SAT denote the set of all satisfiable boolean formulas. We assume that the reader is familiar with the definitions of standard language complexity classes such as $\mathrm{P}, \mathrm{NP}, \mathrm{UP}$, and AM [Bab85, BM88]. We will, however, formally define the various classes of nondeterministic functions that we will be looking at in great detail.

We will use the notation set down by Selman [Sel94] (see also [BLS84]) for defining partial, multivalued functions. A *transducer* is a nondeterministic Turing machine that, in addition to its usual input and work tapes, has a write-only output tape. The transducer T outputs a string y on input x if there exists an accepting path of T on input x that outputs y (we denote that by $T(x) \mapsto y$). Hence, a transducer could be *multivalued* and *partial*, since different accepting computations of the transducer may yield different outputs and since the transducer may not have any accepting computation on the input.

Given a multivalued function f and a string x, we use the following set.

$$set\text{-}f(x) = \{y \mid f(x) \mapsto y\}$$

Next, we define some useful function classes.

Definition 1

(a) PF *is the class of functions computable by a deterministic polynomial-time transducer.*

(b) NPMV *is the class of partial, multivalued functions f for which there is a nondeterministic polynomial-time machine N such that for every x, it holds that*

1. *$f(x)$ is defined if and only if $N(x)$ has at least one accepting computation path, and*
2. *for every y, $y \in set\text{-}f(x)$ if and only if there is an accepting computation path of $N(x)$ that outputs y.*

(c) NPSV *is the class of* single-valued *partial functions in* NPMV.

(d) *A function $f \in$ NPkV if $f \in$ NPMV and for all $x \in \Sigma^*$, $\|set\text{-}f(x)\| \leq k$.*

(e) *A function $f \in$ NPbV if for all x, $set\text{-}f(x) \subseteq \{0,1\}$.*

We will be interested in subclasses of NPMV that are *total*, that is, functions f such that for all $x \in \Sigma^*$, $\|set\text{-}f(x)\| > 0$. Given a function class $\mathcal{F}$, we will denote the set of all total functions in $\mathcal{F}$ by $\mathcal{F}_t$. For example, NPMV_t is the class of total functions in NPMV.

We also need the following technical notion of *refinement*. 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 $f(x)$. If f is a partial multivalued function and $\mathcal{G}$ is a class of partial multivalued functions, we write $f \in_c \mathcal{G}$ if $\mathcal{G}$ contains a refinement g of f, and if $\mathcal{F}$ and $\mathcal{G}$ are classes of partial multivalued functions, we write $\mathcal{F} \subseteq_c \mathcal{G}$ if for every $f \in \mathcal{F}$, $f \in_c \mathcal{G}$. This notion enables us to compare the complexity of two functions that output a different *number* of values (see [Sel94]).

Selman [Sel94] and Hemaspaandra *et al.* [HNOS94] have shown that $\mathrm{NPSV}_t = \mathrm{PF}^{\mathrm{NP}\cap\mathrm{coNP}}$. From this, we get the following useful proposition.

Proposition 1 $\mathrm{NPSV}_t \subseteq \mathrm{PF}$ *if and only if* $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.

We use the notion of refinement to define what it means to invert a many-to-one function. If $f \in \mathrm{PF}$ is an honest function and $\mathcal{F}$ is a function class, then we say that f is *invertible in* $\mathcal{F}$ if f^{-1} has a refinement in $\mathcal{F}$—that is, there exists a function $g \in \mathcal{F}$ such that and $dom(g) = dom(f^{-1})$ and for all x, if $g(x)$ outputs y, then $f(x) \mapsto y$.

If M is a nondeterministic polynomial-time Turing machine, then consider the following function $p_M \in \mathrm{NPMV}$. For all strings $x \in L(M)$, $p_M(x) \mapsto y$ if y is an accepting computation of M on x.

We will abuse notation to use $p_M(x)$ to denote *some* unspecified output value of p_M on input x.

3 Characterizations of Q and Q′

In this section we discuss two hypothesis that we will call **Q** and **Q′** and give several characterizations of each.

Theorem 2 *The following are equivalent.*

1. *For all* NP *machines* M *that accept* Σ^*, *there exists a polynomial-time computable function* g_M *such that for all* x, $g_M(x)$ *outputs an accepting computation of* M *on* x.

2. *All polynomial-time computable* onto *honest functions are invertible in* PF.

3. $\mathrm{NPMV}_t \subseteq_c \mathrm{PF}$.

4. *For all* $S \in P$ *such that* $S \subseteq SAT$, *there exists a poly-time computable* g *such that for all* $x \in S$, $g(x)$ *outputs a satisfying assignment of* x.

5. $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ *and* $\mathrm{NPMV}_t \subseteq_c \mathrm{NPSV}_t$.

6. *For all* $M \in \mathrm{NP}$ *such that* $L(M) = SAT$, $\exists f_M \in \mathrm{PF}$ *such that for all* $x \in SAT$,
$$f_M(x, p_M(x)) \mapsto \text{ a satisfying assignment of } x.$$

7. *For all* $M, N \in \mathrm{NP}$ *such that* $L(M) \subseteq L(N)$, $\exists f_M \in \mathrm{PF}$ *such that* $\forall x \in L(M)$, $f_M(x, p_M(x)) \mapsto p_N(x)$.

8. *For all* $L \in \mathrm{P}$ *and for all* NP *machines* M *that accept* L, $\exists f_M \in \mathrm{PF}$ *such that* $\forall x \in L$, $f_M(x) \mapsto p_M(x)$.

Proof

(1) ⇒ (3): Let $f \in \mathrm{NPMV}_t$. Consider the following NP machine M. On input x, M guesses a value y and accepts x if and only if $f(x) \mapsto y$. Since f is total, $L(M) = \Sigma^*$. By (1), for all x, some accepting path of M is computable in polynomial time. Hence $f \in_c \mathrm{PF}$.

(3) ⇒ (1): Let M be an NP machine accepting Σ^*. Consider the multivalued function, $f_M(x) \mapsto p_M(x)$. Since $L(M) = \Sigma^*$, $f_M \in \mathrm{NPMV}_t$ and thus f_M has a refinement $g_M \in \mathrm{PF}$.

(2) $\iff$ (3): The assertion in (2) is just a restatement of the assertion $\mathrm{NPMV}_t \subseteq_c \mathrm{PF}$.

(3) $\iff$ (5): We simply observe that $\mathrm{NPMV}_t \subseteq_c \mathrm{PF} \iff [\mathrm{NPMV}_t \subseteq_c \mathrm{NPSV}_t$ and $\mathrm{NPSV}_t \subseteq \mathrm{PF}]$ and apply Proposition 1.

(1) ⇒ (6): Suppose M is an NP machine that accepts SAT. Define an NP machine M' as follows. On input $\langle x, p \rangle$, if p is not an accepting computation of M on x, then accept. Else, if p is an accepting computation of M on x, then guess an assignment of x and accept iff it is a satisfying assignment. It is easy to see that $L(M') = \Sigma^*$. By (1), there exists $f \in \mathrm{PF}$ computes an accepting path of M' on input $\langle x, p \rangle$, and when $p = p_M(x)$, a satisfying assignment of x can be recovered from the output of f.

(6) ⇒ (1): Let $L(M) = \Sigma^*$. Let $h \in \mathrm{PF}$ denote the many-one reduction implied by Cook's theorem [Coo71] from M to SAT. Let S be the range of f, that is,

$$S = \{h(x) \mid x \in \Sigma^*\}.$$

Recalling the proof of Cook's theorem, observe that $h(x)$ is a boolean formula that encodes a nondeterministic computation of M on x, so given a satisfying assignment to $h(x)$, some accepting path of M can be computed in polynomial time. Moreover, it follows by the construction of h that x is encoded in $h(x)$. So $S \in \mathrm{P}$.

Now, define an NP machine N as follows. On input ϕ, N accepts immediately if $\phi \in S$. If $\phi \notin S$, then N accepts ϕ if and only if there exists a satisfying assignment to ϕ. It is easy to see that N accepts SAT. By (6), there exists a function g_N such that on input $\langle \phi, p_N(\phi) \rangle$, g_N outputs a satisfying assignment of ϕ.

Now we can compute an accepting computation of M as follows. On input x, let $h(x) = \phi$ and let $g_N(\phi, p_N(\phi))$ output w, a satisfying assignment for ϕ. Now compute an accepting path of M on x using w. Since for all $\phi \in S$, $p_N(\phi)$ is computable deterministically in polynomial time, the above procedure runs in polynomial time.

(7) ⇒ (6): Simply let N be the NP machine that accepts SAT by guessing satisfying assignments.

(3) $\Rightarrow$ (7): Let M and N be such that $L(M) \subseteq L(N)$. Define a function h_M as follows.

$$h_M(x, y) \mapsto \begin{cases} p_N(x) & \text{if } y \text{ is an accepting computation of } M(x) \\ x & \text{otherwise} \end{cases}$$

It is easy to see that $h_M \in \text{NPMV}_t$, since hence for all pairs $\langle x, y\rangle$, if $p_M(x) \mapsto y$, then there must exist a string $z = p_N(x)$, which will be output by h_M. By (3), h_M has a refinement g in PF.

(8) $\Rightarrow$ (1): Trivial.

(3) $\Rightarrow$ (8): Let $L \in$ P and let M be an NP machine that accepts L. Consider the following total function.

$$h_M(x) \mapsto \begin{cases} y & \text{if } x \in L \text{ and } y \text{ is an accepting computation of } M(x) \\ x & \text{otherwise} \end{cases}$$

Clearly, $h_M \in \text{NPMV}_t$, and by (3), h_M has a refinement g_M that can be computable in polynomial time.

(8) $\Rightarrow$ (4): Trivial.

(4) $\Rightarrow$ (8): Let $L \in$ P and let M be an NP machine that accepts L. Let h be the poly-time computable Cook reduction from M to SAT. Let $h(L)$ denote the range of h on strings in L.

$$h(L) = \{h(x) \mid x \in L\}$$

It is easy to see that $h(L) \subseteq SAT$ and $h(L) \in$ P. By (4), there exists a poly-time procedure g that computes a satisfying assignment for all $\phi \in h(L)$. Thus, an accepting computation of M on $x \in L$ can be computed as follows: On input x, compute $g(h(x))$ to obtain a satisfying assignment of $h(x)$. It follows by the encoding in Cook reduction that given a satisfying assignment of $h(x)$, some accepting path of M on x can be computed in polynomial time. □

Definition 2 *We let* **Q** *represent the hypothesis that any (and thus all) of the statements in Theorem 2 hold.*

Suppose **Q** holds and $A, B \in$ NP are such that $A \leq_m^{\mathrm{P}} B$ via a function f. It follows by characterization (6) in Theorem 2 that for all Turing machines M, N such that $L(M) = A$ and $L(N) = B$, there exists a polynomial-time computable function $g_{M,N}$ such that for all $x \in A$,

$$g_{M,N}(x, p_M(x)) \mapsto p_N(f(x)). \tag{1}$$

In their seminal papers on NP-completeness, Karp [Kar72] and Levin [Lev73] gave independent definitions of many-one reductions. The main difference between the Karp and Levin definitions of many-one reduction was that Levin insisted that in addition to instances in A mapping to instances in B, there must be a polynomial algorithm that maps every "witness" of strings in A to some "witness" of the mapped string in B. This is just a restatement of Equation 1, hence **Q** can be stated in another interesting way.

Corollary 3 *Proposition* $\mathbf{Q}$ *holds if and only if for all* $A, B \in \mathrm{NP}$, *every Karp reduction from* A *to* B *is also a Levin reduction.*

Theorem 2 looks at finding entire witnesses. What if we just need a single bit of a witness? This leads to a different set of equivalent propositions.

Theorem 4 *The following are equivalent.*

1. *For all* NP *machines accepting* Σ^* *there is a polynomial-time computable function* g_M *that computes the first bit of an accepting computation of* M.
2. *For all polynomial-time computable onto honest functions* f, *there exists a function* $g \in \mathrm{PF}$ *that computes the first bit of* f^{-1}.
3. $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$.
4. *For all* $S \in \mathrm{P}$ *such that* $S \subseteq SAT$, *there exists a poly-time procedure* f_M *such that for all* $x \in S$, $f_M(x) \mapsto$ *the first bit of a satisfying assignment of* x.
5. *For all* M *such that* $L(M) = SAT$, $\exists f_M \in \mathrm{PF}$ *such that* $\forall x$, $f_M(x, p_M(x)) \mapsto$ *the first bit of a satisfying assignment of* x.
6. $\forall M, N$ *such that* $L(M) \subseteq L(N)$, *there exists* $f_M \in \mathrm{PF}$ *such that for all strings* x, $f_M(x, p_M(x)) \mapsto$ *the first bit of* $p_N(x)$.
7. *[FR94] All disjoint* coNP *sets are P-separable.*

Proof The proof of the equivalence of the first six propositions are analogous to the corresponding proofs in Theorem 2.

Fortnow and Rogers [FR94] showed that (7) is equivalent to (1). □

Definition 3 *We let* $\mathbf{Q}'$ *represent the hypothesis that any (and thus all) of the statements in Theorem 4 hold.*

Remark: In Theorem 4, we can replace any of the occurrences of "the first bit" with any polynomial-time computable boolean function of the bits.

Beame at al. [BCE+95] study the class TFNP, which is the class of functions f in NPMV_t such that the set $graph(f) = \{\langle x, y\rangle \mid f(x) \mapsto y\}$ is in P. Does the graph of every function in NPMV_t belong to P? The following proposition shows that the answer is "no", unless P = NP.

Proposition 5 *If for all* $f \in \mathrm{NPMV}_t$, $graph(f) \in \mathrm{P}$, *then* $\mathrm{P} = \mathrm{NP}$.

Proof Consider the following 2-valued function f, which is clearly in NPMV_t. For all strings $x \in \Sigma^*$, $f(x)$ outputs the number 2, and for all strings $x \in SAT$, $f(x)$ outputs 1. (So if $x \in SAT$, then $f(x)$ outputs 1 and 2 on two different accepting paths.) By hypothesis, $graph(f) \in \mathrm{P}$. It is easy to see that $x \in SAT$ if and only if $\langle x, 1\rangle \in graph(f)$. □

Thus, it might be more meaningful to compare these classes using refinements. We ask whether every NPMV_t-function has a *refinement* whose graph is in P (in symbols, is $\mathrm{NPMV}_t \subseteq_c \mathrm{TFNP}$). We show that this hypothesis is intermediate in complexity between $\mathbf{Q}$ and $\mathbf{Q}'$.

Theorem 6 **(i)** *If* $\mathbf{Q}$ *holds, then* $\mathrm{NPMV}_t \subseteq_c \mathrm{TFNP}$.

(ii) *If* $\mathrm{NPMV}_t \subseteq_c \mathrm{TFNP}$*, then* $\mathbf{Q}'$ *holds.*

Proof

(i) We have $\mathrm{NPMV}_t \subseteq_c \mathrm{PF} \subseteq \mathrm{TFNP}$.

(ii) Let f be a function in NPbV_t. We want to show that f has a refinement in PF.

By hypothesis, there exists a function $g \in \mathrm{NPMV}_t$ such that g is a refinement of f and $graph(g) \in \mathrm{P}$. Let M be the polynomial-time TM that accepts $graph(g)$. Then, a polynomial-time refinement N of f can be described as follows. On input x, N simulates M on input $\langle x, 0\rangle$ and $\langle x, 1\rangle$. Since g is total, M must accept at least one of $\langle x, 0\rangle$ or $\langle x, 1\rangle$. If M accepts $\langle x, b\rangle$, for some $b \in \{0, 1\}$, then N outputs b. This implies that $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$, and hence $\mathbf{Q}'$ holds. □

Finally, using the fact that for all $M \in \mathrm{NP}$ such that $L(M) = \Sigma^*$, the accepting path of any given input can be verified in polynomial time, we get the following characterization of $\mathbf{Q}$.

Proposition 7 $\mathbf{Q}$ *holds if and only if* $\mathrm{TFNP} \subseteq_c \mathrm{PF}$

Hemaspaandra, Rothe and Wechsung [HRW95] define the complexity class $\mathrm{EASY}_\forall^\forall$ as the class of NP languages L such that for all NP machines M, if $L(M) = L$, then $p_M \in_c \mathrm{PF}$. It is easy to see that $\mathbf{Q}$ can be formulated as follows.

Proposition 8 $\mathbf{Q}$ *hold if and only if* $\mathrm{EASY}_\forall^\forall = \mathrm{P}$.

4 Relationships with other Complexity Hypotheses

In this section, we ask how propositions $\mathbf{Q}$ and $\mathbf{Q}'$ relate to other well-known complexity hypotheses. The following relationships are either well-known or easy to prove.

Proposition 9 **(i)** *[BD76, IN88] If* $\mathbf{Q}'$ *holds, then* $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.

(ii) *If* $\mathbf{Q}'$ *holds, then one-way permutations do not exist.*

(iii) *If* $\mathrm{P} = \mathrm{NP}$ *then* $\mathbf{Q}$ *holds.*

Next, we consider an interesting open question in structural complexity, namely, whether NP = UP implies that the polynomial hierarchy collapses. We show that if $\mathbf{Q}'$ holds, then the answer to this question is affirmative. This fact is interesting since it is not known whether $\mathbf{Q}'$ itself implies a collapse of the polynomial hierarchy.

Theorem 10 *If* $\mathbf{Q}'$ *holds and* $\mathrm{NP} = \mathrm{UP}$*, then* $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}} \subseteq \Sigma_2^{\mathrm{P}}$.

Proof It suffices to show that $\mathbf{Q}'$ and NP = UP implies that $\mathrm{NPMV} \subseteq_c \mathrm{NPSV}$, since by a result of Hemaspaandra *et al.* [HNOS94], if $\mathrm{NPMV} \subseteq_c \mathrm{NPSV}$, then $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$. Further, to prove that $\mathrm{NPMV} \subseteq_c \mathrm{NPSV}$, it suffices to show that there exists a *single-valued* nondeterministic transducer that computes a satisfying assignment of a given boolean formula [Sel94].

Let M be an UP machine accepting SAT. Since $\mathbf{Q}'$ holds, there exists a function $f_M \in \mathrm{PF}$ that computes the first bit of a satisfying assignment of ϕ, given ϕ and $p_M(\phi)$ as input. Let q be a polynomial that bounds the running time of M.

Now consider the following nondeterministic transducer T. On input $\phi(x_1, x_2, \ldots, x_n)$, guess n-pairs of strings: $(\langle y_1, b_1\rangle, \ldots, \langle y_n, b_n\rangle)$ such that $b_1, b_2, \ldots, b_n \in \{0,1\}$ and $y_1, \ldots, y_n \in \{0,1\}^{q(n)}$.

Now verify that $y_1 = p_M(\phi(x_1, \ldots, x_n))$, $b_1 = f_M(\phi, y_1)$, and for all $i, 2 \leq i \leq n$, $y_i = p_M(\phi(b_1, \ldots, b_{i-1}, x_i, \ldots, x_n))$ and $b_i = f_M(x, y_i)$. If all the above conditions hold, then output $b_1 \cdot b_2 \cdots b_n$.

It is easy to see that $b_1 \cdots b_n$ is a satisfying assignment of ϕ, since $b_n = f_M(\phi(b_1, \ldots, b_{n-1}, x_n))$. We need to show that $b_1 \ldots b_n$ is unique—that is, no two accepting computations of T output two different assignments. This follows from our following claim.

Claim 1 *For all i, $1 \leq i \leq n$, if $b_1, \ldots, b_{i-1}$ are unique, then b_i is unique.*

Proof If $b_1, \ldots, b_{i-1}$ are unique, then $\phi(b_1, \ldots, b_{i-1}, x_i, \ldots, x_n)$ is unique, and since M is a UP machine, $p_M(\phi(b_1, \ldots, b_{i-1}, x_i, \ldots, x_n))$ is unique too. Recall that $f_M \in \mathrm{PF}$, so the claim follows. $\square$

Thus T is an NPSV transducer that outputs unique satisfying assignments, and hence $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$. $\square$

A set Z is *paddable* if there exists a function $g(\cdot,\cdot) \in \mathrm{PF}$ that is one-to-one, length-increasing and p-time invertible in both arguments, and has the property that for all strings x and y, $x \in Z \iff g(x,y) \in Z$. A *1-1 paddable degree* consists of all sets that are 1-1 equivalent to some paddable set. A *length-increasing* degree is a set $\mathcal{C}$ of languages such that for all $A, B \in \mathcal{C}$, there exists a many-one reduction f from A to B and for all strings x, $|f(x)| > |x|$. Paddable sets play an important role in the study of the isomorphism conjecture [BH77]. SAT is known to be paddable, so the class of NP-complete sets form a paddable degree. Berman and Hartmanis [BH77] showed that if A and B are reducible to each other by 1-1 length-increasing and invertible reductions, then A and B are isomorphic. Thus, if every paddable degree collapses to a 1-1 length-increasing and invertible degree, then the isomorphism conjecture holds. Here we show that if $\mathbf{Q}$ holds, then a weaker form of the above implication is true..

Theorem 11 *If $\mathbf{Q}$ holds, then every 1-1 paddable degree is a 1-1 length increasing degree.*

Proof Let A and B be many-one equivalent and let $A \leq_m^{\mathrm{P}} B$ via a one-to-one function f. If B is paddable, then trivially, A reduces to B via a 1-1 length-increasing reduction [BH77]. Now assume that A is paddable. Let g be the padding function of A. We will show that A reduces to B via a one-to-one length-increasing reduction.

A one-to-one length-increasing reduction h' from A to B can be constructed as follows. Let x be an input string. Consider the set $pad(x) = \{g(x,y) \mid y \in \Sigma^{|x|+2}\}$. Now consider the set $Im(x) = \{f(w) \mid w \in pad(x)\}$. Since f is 1-1, it must map distinct strings in $pad(x)$ to distinct strings. Since g is 1-1 by definition, $\|Im(x)\| > 2^{|x|+1}$. Thus, by the pigeon-hole principle, for all $x \in \Sigma^*$, there exists a string $z \in Im(x)$ such that $|z| > |x|$.

Define h to be the NPMV function that maps x to z such that $z = f(w)$, $w \in pad(x)$, and $|z| > |x|$. It is easy to see that h is total. Since $\mathbf{Q}$ holds, h has a refinement h' in PF. Hence h' is the 1-li reduction from A to B. $\square$

We now extend Proposition 9, part (i) to probabilistic classes. It is interesting to note that none of the following collapses are known to be implied by the hypothesis $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.

Theorem 12 **(a)** $\mathbf{Q}' \rightarrow \mathrm{AM} \cap \mathrm{coAM} = \mathrm{BPP}$.

(b) $\mathbf{Q}' \rightarrow \mathrm{NP} \cap \mathrm{coAM} = \mathrm{RP}$.

Proof To prove (a), let $L \in \mathrm{AM} \cap \mathrm{coAM}$. It follows by a result of Furer *et al.* [FGM+89], that the $\mathrm{AM} \cap \mathrm{coAM}$ protocol for L can be converted to a protocol with "one-sided error", that is, for all strings x, the "correct" verifier will accept x for all random strings. Let V_1 and V_2 be the verifiers for the Arthur-Merlin systems for L and $\overline{L}$. Consider the following Turing machine M that accepts $\Sigma^* \times \Sigma^*$. On input $\langle x, r\rangle$, M guess a "response" from Merlin on input x and then nondeterministically simulates a computation of V_1 or V_2 on input x with the random string r. If either V_1 or V_2 accept, then accept $\langle x, r\rangle$. Clearly, M accepts $\Sigma^* \times \Sigma^*$, and since $\mathbf{Q}$ holds, there exists a polynomial-time computable function f_M that, on input x, outputs a computation of M. Hence, membership in L can be determined as follows. On input x, simulate $f_M(x, r)$ on a random string r. If the output of f_M is an accepting computation of V_1, then accept, else reject. It is easy to see that the above procedure will be correct with high probability. Hence $L \in \mathrm{BPP}$.

The proof of (b) is identical to the proof of (a)—now M also guess a witness for x if $x \in L$, hence the BPP algorithm described above is an RP algorithm. □

One interesting consequence of the Theorem 12 is that if $\mathbf{Q}$ holds, then the graph isomorphism problem is in RP since Goldreich, Micali and Wigderson [GMW91] showed that graph isomorphism is in coAM.

We end this section by listing the relativized results that are known about $\mathbf{Q}$ and $\mathbf{Q}'$.

Theorem 13 *The following relativized results are known.*

1. *[BG81, Ver93]* $\mathbf{Q}$ *and* $\mathbf{Q}'$ *fails for random oracles and generic oracles.*
2. *[FR94]* $\mathbf{Q}$ *holds relative to any sparse generic oracle with the* subset property *(any subset of the sparse generic set is also a sparse generic).*[1]
3. *[FR94] There exists an oracle A such that* $\mathrm{NP}^A \neq \mathrm{coNP}^A$ *and* $\mathbf{Q}^A$ *holds.*
4. *There exists an oracle B such that* $\mathrm{NP}^B = \mathrm{UP}^B$, $\mathbf{Q}^B$ *holds and* $\mathrm{NP}^B \neq \mathrm{coNP}^B$.
5. *[FR94, IN88, CS93] There exists an oracle C such that* $\mathrm{P}^C = \mathrm{NP}^C \cap \mathrm{coNP}^C$ *and* $\mathbf{Q}'^C$ *fails.*
6. *[FFK96] There exists an oracle D such that* $\mathbf{Q}^D$ *fails and the isomorphism conjecture holds relative to D.*
7. *[KMR89] There exists an oracle E such that* $\mathbf{Q}^E$ *fails and the isomorphism conjecture fails relative to E.*

Proof To prove (3), it is not hard to see that the oracle in (2) can be constructed so that NP = UP relative to the oracle. Hence the claim follows. □

In particular, the oracle in (3) implies that the collapse of the polynomial hierarchy in Theorem 10 is unlikely to be improved to NP = coNP. This also shows that the result of Hemaspaandra et al. [HNOS94] is optimal under relativizable proof techniques.

[1] See [FR94] for a discussion on sparse genericity.

5 One Bit vs. Many Bits

In this section we ask the question, does $\mathbf{Q}$ hold if and only if $\mathbf{Q}'$ hold? This question remains open even in relativized worlds.

We can rephrase the question as

Does $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$ imply that $\mathrm{NPMV}_t \subseteq_c \mathrm{PF}$?

Note that the answer to the analogous question for partial functions is trivial, since $\mathrm{NPbV} \subseteq_c \mathrm{PF}$ implies that $\mathrm{P} = \mathrm{NP}$. However, a collapse of $\mathrm{P} = \mathrm{NP}$ is not known to be implied the corresponding hypothesis about total functions.

The following theorem obtains a partial "collapse" result for total functions. The proof technique involves using *binary search with multivalued oracles*, which might be of independent interest.

Theorem 14 *For all $k \geq 0$,*

$$\mathrm{NPbV}_t \subseteq_c \mathrm{PF} \iff \mathrm{NPkV}_t \subseteq \mathrm{PF}.$$

Proof We will show that if $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$, then for all $k \geq 2$, $\mathrm{NPkV}_t \subseteq_c \mathrm{NP}(k-1)\mathrm{V}_t$. By induction, this implies that $\mathrm{NPkV}_t \subseteq \mathrm{NPSV}_t$. The theorem then follows by Theorem 4 and Propositions 9(i) and 1.

Let $f \in \mathrm{NPkV}_t$ for some constant $k \geq 2$. Suppose that for every input x we are given—as free advice—some value $c(x)$ which is guaranteed to be between the minimum and maximum outputs of $f(x)$, inclusive ($c(x)$ is otherwise arbitrary). We can then nondeterministically compute a refinement of f with at most $k-1$ values for every input x, as described by the algorithm $\mathbf{A}$ below. We then show that if $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$, then such a $c(x)$ can be computed in polynomial time, which then implies that $f \in_c \mathrm{NP}(k-1)\mathrm{V}_t$, which proves the theorem.

Begin A
Input: x. ($c(x)$ is also given as free advice.)
Guess an output y of $f(x)$
if $y = c(x)$, then output y and halt.
else begin
    $S := \{y\}$
    repeat
        Guess an output z of $f(x)$
            such that $z \notin S$
        $S := S \cup \{z\}$
    until S contains an element $\geq c(x)$.
    if $c(x)$ is the maximum element of S, then
        Output $c(x)$ and halt.
    else
        Output the minimum element of S
end
End A

We claim that procedure **A** outputs a refinement of f with at least one and at most $k-1$ values. First, note that all outputs of **A** are also outputs of $f(x)$. Second, note that **A** is total: if the repeat loop is entered, then by our assumption about $c(x)$ there must be at least two outputs of $f(x)$, and since at least one output is $\geq c(x)$, a value of z will always be found, and the loop will eventually terminate.

We now show that for all x, $\mathbf{A}(x)$ will output less than k strings. There are two cases:

1. If $c(x)$ is the maximum output of $f(x)$, then **A** will only output $c(x)$ on any accepting path, i.e., $\mathbf{A}(x)$ is 1-valued.

2. If $c(x)$ is less than some output of $f(x)$, then the maximum output of $f(x)$ is *never* output on any accepting path of **A**. This is because any accepting path will either output $c(x)$ or else the minimum of a set of at least two distinct outputs of $f(x)$. In this case, **A** outputs at most $k-1$ outputs of $f(x)$.

Now to complete the proof, assume that $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$. We show how to compute a value $c(x)$, lying between the extreme values of $f(x)$, via something akin to binary search. Let M be an NP machine that on input (x, y) outputs 0 if there is a value z of $f(x)$ with $z \leq y$, and outputs 1 if there is a value z of $f(x)$ with $z \geq y$ (the machine may output both values on different paths). M computes an NPbV_t function, so it has a refinement $Up(x, y)$ in PF. Note that if y is less (resp. greater) than all outputs of $f(x)$, then $Up(x, y) = 1$ (resp. $Up(x, y) = 0$). Fixing x, we perform "binary search" on the space of all y (up to an appropriate polynomial length bound), where for each probe y' in the middle of a range, we use $Up(x, y')$ to tell us where to continue searching—the upper half iff $Up(x, y') = 1$. By the aforementioned properties of Up, we will be steered into the range spanning the outputs of $f(x)$, and will converge on a value $c(x)$ satisfying our requirements. $\square$

6 Open Questions

The following questions remain open.

1. Does **Q** imply that the polynomial hierarchy collapses? Is there an oracle relative to which **Q** holds and the polynomial hierarchy does not collapse to Σ_2^{P}?

2. Is there an oracle relative to which $\mathbf{Q}'$ holds but **Q** fails?

3. For some non-constant function f, does $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$ imply that $\mathrm{NPfV}_t \subseteq_c \mathrm{PF}$?

4. Does **Q** and P=UP imply that the polynomial hierarchy collapses?

5. **Q** and the Isomorphism Conjecture: Is there an oracle relative to which **Q** holds and the Isomorphism conjecture holds?

7 Acknowledgments

The authors would like to thank Lane Hemaspaandra, Stuart Kurtz, and Alan Selman for their insightful comments on this work.

References

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of 17th Annual ACM Symposium on Theory of Computing*, pages 421–429, 1985.

[BCE+95] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. In *Proceedings of 27th ACM Symposium on Theory of Computing*, pages 303–314, 1995.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR76-284, Cornell University, Department of Computer Science, Upson Hall, Ithaca, NY 14853, 1976.

[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*, 10:96–113, 1981.

[BH77] L. Berman and H. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, 1977.

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

[BM88] L. Babai and S. Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of 13th Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[CS93] P. Crescenzi and R. Silvestri. Sperner's lemma and Robust Machines. In *Procs. of 8th Annual Structure in Complexity Theory*, pages 194–199, 1993.

[FFK96] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. *SIAM Journal on Computing*, 25(1):193–206, 1996.

[FGM+89] M. Furer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 429–442. JAI Press, Greenwich, 1989.

[FR94] L. Fortnow and J. Rogers. Separability and one-way functions. In D.Z. Du and X. S. Zhang, editors, *Proccedings of 5th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, pages 396–404. Springer Verlag, 1994.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *JACM*, 38(3):691–729, 1991.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17, 1988.

[HNOS94] L. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. Computing unique solutions collapses the polynomial hierarchy. In D.Z. Du and X. S. Zhang, editors, *Proccedings of 5th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, pages 56–64. Springer Verlag, 1994. To appear in SIAM J. of Comput.

[HRW95] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. Technical Report MATH/95/5, Friedrich-Schiller-Universität Jena, May 1995.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of 3rd Annual Conference on Structure in Complexity Theory*, pages 29–38, 1988.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proceedings of 21st Annual ACM Symposium on Theory of Comput.*, pages 157–166, 1989.

[Lev73] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973. English translation of original in *Problemy Peredaci Informacii.*

[Pap94] C. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, pages 498–532, 1994.

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

[Ver93] N. Vereshchagin. Relationships between NP-sets, Co-NP-sets and P-sets relative to random oracles. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 132–138. IEEE, New York, 1993.