On Inverting Onto Functions

Stephen A. Fenner[1] Lance Fortnow[2]

Ashish V. Naik[3] John D. Rogers[4]

November 6, 1995

[1]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

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

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

[4]School of Computer Science, Telecommunications, and Information Systems, Depaul University, Chicago, IL 60604. Email: jrogers@cs.depaul.edu. Work done while at the Computer Science Department, University of Chicago.

Abstract

We study the complexity of inverting many-one, honest, polynomial-time computable *onto* functions. Asserting that every polynomial-time computable, honest, onto function is invertible is equivalent to the following proposition that we call **Q**: 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.

We show that **Q** is equivalent to several well-studied propositions in complexity theory. For example, we show that **Q** is equivalent to the proposition that, for all NP machines M that accept SAT, there exists a polynomial-time algorithm g_M that transforms any accepting computation of M on input x into a satisfying assignment of x.

We compare **Q** with its following weaker version that we call **Q′**: 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. As a first step in comparing **Q** and **Q′**, we show that if every 0-1-valued total NPMV function has poly-time computable refinements, then for all $k \geq 0$, every k-valued total NPMV function has refinements in PF. We relate both **Q** and **Q′** to the question of whether the class NPMV_t has refinements in TFNP, a class of functions studied by Beame et al.

Finally, we study the relationship of **Q** and **Q′** with other complexity hypothesis. We show that **Q′** implies that $\mathrm{AM} \cap \mathrm{coAM} \subseteq \mathrm{BPP}$, and $\mathrm{NP} \cap \mathrm{coAM} \subseteq \mathrm{RP}$. Also, **Q′** and $\mathrm{NP} = \mathrm{UP}$ implies that the polynomial hierarchy collapses to $\mathrm{ZPP}^{\mathrm{NP}}$, and **Q** implies that every one-one paddable degree collapses to a one-one length-increasing degree.

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 $\mathrm{P} \neq \mathrm{UP}$. 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 $\mathrm{P} = \mathrm{NP}$. Thus, most researchers believe that poly-time computable, non-invertible functions exist: indeed, several results in public-key cryptography [ESY84] and pseudorandom generators [ILL89] are proven under this assumption.

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 $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$, and Borodin and Demers [BD76] showed that, if every many-to-one, poly-time computable onto function is poly-time invertible, then $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$. However, these consequences are still weaker than $\mathrm{P} = \mathrm{NP}$. Indeed, it is conceivable that every poly-time computable, honest, onto function is invertible in polynomial time, but $\mathrm{P} \neq \mathrm{NP}$. 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 study the complexity of inverting many-to-one *onto* functions. The hypothesis that all polynomial-time computable, honest, onto functions are invertible is equivalent to the following proposition that we call Proposition **Q**.

> **Proposition Q**: For all NP machines M such that $L(M) = \Sigma^*$, there exists a poly-time computable function f_M such that for all strings x, $f_M(x)$ outputs an accepting computation of M on x.

Propostion **Q** is equivalent to several other fundamental, yet seemingly unrelated propositions in complexity theory. For example, **Q** is equivalent to the following interesting 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. As a corollary, it follows that if **Q** 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 propositions to which **Q** is shown to be equivalent 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). The equivalence of **Q** to these much-studied complexity hypotheses illustrates the robustness of **Q**.

We also consider a weaker proposition and ask—can we efficiently compute a single bit of an inverse of an onto function? We call this proposition $\mathbf{Q}'$.

> **Proposition $\mathbf{Q}'$** For all NP machines M such that $L(M) = \Sigma^*$, there exists

a poly-time computable function f_M such that for all strings x, $f_M(x)$ outputs the first bit of some accepting computation of M on x.

In addition to the equivalence of $\mathbf{Q}'$ to various "one-bit" versions of $\mathbf{Q}$, $\mathbf{Q}'$ is equivalent to the following much studied proposition [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).

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 $\mathbf{Q}'$.

Are $\mathbf{Q}$ and $\mathbf{Q}'$ equivalent? In other words, 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 both of the hypotheses is equivalent to $\mathrm{P} = \mathrm{NP}$. However, since neither $\mathbf{Q}$ nor $\mathbf{Q}'$ are known to be equivalent to $\mathrm{P} = \mathrm{NP}$, 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 a "binary search technique with errors" 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 $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ [BD76, IN88]. Continuing this line of research, we show that $\mathbf{Q}'$ implies that $\mathrm{AM} \cap \mathrm{coAM} = \mathrm{BPP}$ and that $\mathrm{NP} \cap \mathrm{coAM} = \mathrm{RP}$. Thus, if $\mathbf{Q}'$ holds, then the graph isomorphism problem is in RP, which is not known to follow by the assumption that $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$. Next, we study how assuming that $\mathbf{Q}$ holds affects some well-studied open questions in complexity theory. The first question is whether $\mathrm{NP} = \mathrm{UP}$ implies that the polynomial hierarchy collapses. While neither hypothesis $\mathbf{Q}$ nor $\mathrm{NP} = \mathrm{UP}$ are by themselves known to imply to collapse of the polynomial hierarchy, we show that if both $\mathbf{Q}'$ and $\mathrm{NP} = \mathrm{UP}$ hold, then $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}} \subseteq \Sigma_2^{\mathrm{P}}$. 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. We conclude by listing the open questions in Section 5.

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]. 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.

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

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-$f(x)$ 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 iff $f \in$ NPMV and for all $x \in \Sigma^*$, $\|$set-$f(x)\| \leq k$.*

(e) *A function $f \in$ NPbV iff $f \in$ NP2V and for all x, set-$f(x) \subseteq \{0, 1\}$.*

We will be interested in subclasses of NPMV that are *total*, that is, functions f such that for all $x \in \Sigma^*$, $\|\text{set-f}(x)\| > 0$. 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]).

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 for all x, if $f^{-1}(x)$ is defined, then $g(x)$ outputs *some* value of $f^{-1}(x)$.

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

$$p_M(x) \mapsto y \quad y \text{ is an accepting computation of } M \text{ on } x.$$

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 show that **Q** and **Q′** are equivalent to several, seemingly unrelated complexity hypotheses.

Theorem 1 *The following are equivalent.*

1. *Proposition* **Q** *holds.*
2. *All polynomial-time computable* onto *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 $x \in SAT$,*
$$f_M(x, p_M(x)) \mapsto \textit{ a satisfying assignment of } x.$$
7. *For all $M, N \in \mathrm{NP}$ such that $L(M) \subseteq L(N)$, $\exists f_M \in_c \mathrm{PF}$ 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 See appendix. □

Suppose **Q** holds and $A, B \in \mathrm{NP}$ are such that $A \leq_m^{\mathrm{P}} B$ via a function f. It follows by Theorem 1, part (6) 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 2 *Proposition* **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.*

Next, we characterize $\mathbf{Q}'$.

Theorem 3 *The following are equivalent.*

1. *Proposition* $\mathbf{Q}'$ *holds.*
2. *For all polynomial-time computable onto 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* $\forall x$,
$$f_M(x, p_M(x)) \mapsto 1^{st} \textit{ bit of the satisfying assignment of } x.$$
6. $\forall M, N$ *such that* $L(M) \subseteq L(N)$, *there exists* $f_M \in \mathrm{PF}$ *such that for all strings* x,
$$f_M(x, p_M(x)) \mapsto 1^{st} \textit{ bit of } p_N(x).$$
7. *[FR94] All disjoint* coNP *sets are P-separable.*

Proof Fortnow and Rogers [FR94] showed that (7) is equivalent to $\mathbf{Q}'$. The rest of the proofs are analogous to the corresponding proofs in Theorem 1. □

Remark: In Theorem 3, we can replace "the first bit" in parts 2, 4, 5 and 6 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 4 *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 5 **(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 Suppose $\mathbf{Q}$ holds. Then, by Theorem 1, part 3 it follows that $\mathrm{NPMV}_t \subseteq \mathrm{PF}$. Thus trivially, $\mathrm{NPMV}_t \subseteq_c \mathrm{TFNP}$.

To prove (ii), let $\mathrm{NPMV}_t \subseteq_c \mathrm{TFNP}$. 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 $dom(g) = dom(f) = \Sigma^*$, and for all $x, y \in \Sigma^*$, if $g(x)$ maps to y, then $f(x)$ maps to y. Moreover, $graph(g) \in \mathrm{P}$. 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. $\square$

Hemaspaandra, Rothe and Wechsung [HRW95] define the complexity class $\mathrm{EASY}_\forall^\forall$ as the class of 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 6 $\mathbf{Q} \iff \mathrm{EASY}_\forall^\forall = \mathrm{P}$.

4 One bit versus many bits

In this section we ask the question, does $\mathbf{Q}$ hold if and only if $\mathbf{Q}'$ hold? Recall that this can be rephrased as,

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

Let us first consider an analogous question for *partial* multivalued functions—that is, whether $\mathrm{NPbV} \subseteq_c \mathrm{PF} \rightarrow \mathrm{NPMV} \subseteq_c \mathrm{PF}$. Selman [Sel94] showed by using the classical "self-reducibility pruning" argument that if $\mathrm{NPSV} \subseteq_c \mathrm{PF}$, then $\mathrm{NPMV} \subseteq_c \mathrm{PF}$. However, these techniques do not seem to carry over for classes of total functions.

The following theorem obtains a partial "collapse" result for total functions. The proof uses a *binary search procedure with multivalued oracles* that might be of independent interest.

Theorem 7 *For all* $k \geq 0$,

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

We need the following lemma to prove the main theorem.

Lemma 8 *If* $\mathbf{Q}'$ *holds, then* $\mathrm{NPSV}_t \subseteq_c \mathrm{PF}$.

Proof See appendix. $\square$

Proof of Theorem 7 We will show that if $\mathrm{NPbV}_t \subseteq_c \mathrm{PF}$, then $\mathrm{NPkV}_t \subseteq_c \mathrm{NP}(k-1)\mathrm{V}_t$. By induction, this implies that $\mathrm{NPkV}_t \subseteq \mathrm{NPSV}_t$. The theorem then follows by Lemma 8 and Theorem 3.

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 **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

$z \le y$, and outputs 1 if there is a value z of $f(x)$ with $z \ge y$ (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. □

5 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) $\mathrm{P} = \mathrm{NP} \to \mathbf{Q}$

Next, we consider one of the main open questions in structural complexity, namely, whether $\mathrm{NP} = \mathrm{UP}$ 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 See appendix. □

A set Z is *paddable* if there exists a function $g(\cdot, \cdot)$ 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, $x \in Z \iff g(x, y) \in Z$. Paddable sets play an important role in the study of the isomorphism conjecture [BH77]. A *1-1 paddable degree* consists of all sets 1-1 equivalent to some paddable set. We show that if $\mathbf{Q}$ holds then every 1-1 paddable degree collapses to a 1-1 length increasing degree.

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

Proof See appendix. □

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

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

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

Proof See appendix. □

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. *[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]
2. *[FR94] There exists an oracle A such that* $\mathrm{NP}^A \neq \mathrm{coNP}^A$ *and* $\mathbf{Q}^A$ *holds.*
3. *There exists an oracle B such that* $\mathrm{NP}^B = \mathrm{UP}^B$, $\mathbf{Q}^B$ *holds and* $\mathrm{NP}^B \neq \mathrm{coNP}^B$.
4. *[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.*
5. *[FFK92] There exists an oracle D such that* $\mathbf{Q}^D$ *fails and the isomorphism conjecture holds relative to D.*
6. *[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.

6 Open Questions

The following questions remain open.

1. Does $\mathbf{Q}$ imply that the polynomial hierarchy collapses? Is there an oracle relative to which $\mathbf{Q}$ holds and the polynomial hierarchy does not collapse to Σ_2^{P}?
2. Is there an oracle relative to which $\mathbf{Q}'$ holds but $\mathbf{Q}$ 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 $\mathbf{Q}$ and P=UP imply that the polynomial hierarchy collapses?
5. $\mathbf{Q}$ and the Isomorphism Conjecture: Is there an oracle relative to which Q holds and the Isomorphism conjecture holds?

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

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.

[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.

[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.

[ESY84] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information And Control*, 61(2):159–173, May 1984.

[FFK92] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proceedings of 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 30–39, 1992.

[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.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of 21st annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[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.

Appendix

Here we give proofs of some of our main theorems.

A Proofs of Theorems in Section 3

Theorem 1 *The following are equivalent.*

1. *Proposition* **Q** *holds.*
2. *All polynomial-time computable* onto *functions are invertible in* PF.
3. $\mathrm{NPMV}_t \subseteq_c \mathrm{PF}$.
4. *If $S \in P$ such that $S \subseteq SAT$, then 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 $L(M) \subseteq L(N)$, $\exists f_M \in_c \mathrm{PF}$ 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) $\iff$ (3): Let **Q** hold and let $f \in \mathrm{NPMV}_t$. Consider the following NP machine M that accepts Σ^*. On input x, M guesses a value y and accepts x if and only if $f(x) \mapsto y$. Since f is total, $L(M) = \Sigma^*$. Since hypothesis **Q** holds, for all x, some accepting path of M is computable in polynomial time. Hence $f \in_c \mathrm{PF}$. Conversely, 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 $g_M \in \mathrm{PF}$. Hence **Q** holds.

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

(3) $\iff$ (4): We simply observe that $\mathrm{NPMV}_t \subseteq_c \mathrm{PF} \iff [\mathrm{NPMV}_t \subseteq \mathrm{NPSV}_t$ and $\mathrm{NPSV}_t \subseteq_c \mathrm{PF}]$. The equivalence now follows by the relation $\mathrm{NPSV}_t = \mathrm{PF}^{\mathrm{NP} \cap \mathrm{coNP}}$ [Sel94, HNOS94].

(1) $\iff$ (6): Suppose **Q** holds and 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 a truth assignment of x and accept iff it is a satisfying assignment. It is easy to see that $L(M') = \Sigma^*$. Since **Q** holds, 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. Thus (1) implies (6).

To prove the converse, let $L(M) = \Sigma^*$. Let S be the range of Cook's reduction applied to M—that is, S is the set of boolean formulae obtained by encoding (see [Coo71]) the computations of M on strings in Σ^*. Let $f \in$ PF be the reduction implicit in Cook's theorem [Coo71] from M to SAT. Observe that $S \in$ P.

Define M' as follows. On input ϕ, accept immediately if $\phi \in S$. If $\phi \notin S$, then accept ϕ if and only if there exists a satisfying assignment to ϕ. It is easy to see that M' accepts SAT. It follows by the hypothesis that there exists a function $g_{M'}$ such that on input $\langle \phi, p_{M'}(\phi) \rangle$, $g_{M'}$ outputs a satisfying assignment of ϕ. And for all $\phi \in S$, $p_{M'}(\phi)$ is computable in polynomial time.

Now we can compute an accepting computation of M as follows. On input x, let $f(x) = \phi$ and let $g_{M'}(\phi, p_{M'}(\phi)) = w$. The string w is a satisfying assignment for ϕ. It follows by the encoding in Cook's theorem that an accepting computation of M can be recovered from w. Hence **Q** holds.

(6) $\iff$ (7): To show that (7) implies (6) is easy—simply let N be the NP machine that accepts SAT be guessing satisfying assignments. We will now show that (3) implies (7), which will imply that (6) implies (7). Let M and N be such that $L(M) \subseteq L(N)$. Define a function h_M as follows.

$$h_M(x, y) \mapsto \begin{cases} p_M(x) & \text{if } p_M(x) \mapsto y \\ x & \text{otherwise} \end{cases} .$$

It is easy to see that $h_M \in \mathrm{NPMV}_t$, since hence for all pairs $\langle x, y \rangle$, if $y = p_M(x)$, 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. Thus (7) holds.

(1) $\iff$ (8): One direction ((8) $\rightarrow$ (1)) is trivial. For the converse, it suffices to prove that (3) implies (8). Let $L \in$ P and let M be an NP machine that accepts L. Consider the following total function.

$$h_M(x) \mapsto \begin{cases} p_M(x) & \text{if } x \in L \\ x & \text{otherwise} \end{cases}$$

Clearly, $h_M \in \mathrm{NPMV}_t$, and by (3), h_M has a refinement g_M that can be computable in polynomial time. Thus, (8) holds.

(5) $\iff$ (8): Once again (8) $\rightarrow$ (5) is trivial. Now suppose that (5) holds. Let $S \in$ P and let M be an NP machine that accepts S. Let h be the poly-time computable Cook reduction from M to SAT. Consider the following range-set S',

$$S' = \{\phi \mid \exists x \in S, h(x) \mapsto \phi\}$$

It is easy to see that $S' \subseteq SAT$. It follows by definition of Cook reduction that the string x such that $h(x) \mapsto \phi$ is encoded in ϕ. Also, whether $x \in S$ can be determined in polynomial time. Thus, $S' \in$ P. Since (5) holds, there exists a poly-time procedure g that computes a satisfying assignment of all $\phi \in S'$. Thus, an accepting computation of M on $x \in S$ can be computed as follows. On input x, compute $g(h(x))$ to obtain a satisfying assignment of $g(x)$. It follows by the encoding in Cook reduction that given a satisfying assignment of $h(x)$, an accepting path of M on x can be computed in polynomial time. Thus, (8) holds. $\square$

B Proofs in Section 4

Lemma 8 *If* $\mathbf{Q}'$ *holds, then* $\mathrm{NPSV}_t \subseteq_c \mathrm{PF}$.

Proof Let $h \in \mathrm{NPSV}_t$. Assume, without loss of generality that there is a polynomial p such that for all strings x, the length of the output of $h(x)$ is exactly $p(|x|)$. We will denote the i^{th} bit of $h(x)$ by $h_i(x)$. Define a 0-1 valued function g as follows.

$$g(x,i) \mapsto \begin{cases} h_i(x) & \text{if } i \le p(|x|) \\ 0 & \text{otherwise} \end{cases}$$

Since g is 0-1 valued and $\mathbf{Q}'$ holds, there exists a refinement g' of g such that $g' \in \mathrm{PF}$. Given x, the value of $h(x)$ can be obtained simply by simulating $g'(x,1), g'(x,2), \ldots, g'(x,p(|x|))$ and then concatenating the output. Thus $h \in_c \mathrm{PF}$. □

C Proofs in Section 5

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

Proof It suffices to show that $\mathbf{Q}'$ and $\mathrm{NP} = \mathrm{UP}$ 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. For convenience, we will assume that for a boolean formula ϕ with variables $x_i, \ldots, x_k$, f_M outputs the value of x_i in the satisfying assignment.

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

It is easy to see that $b_1 \ldots b_n$ 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*, 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. □

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

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, the trivially, A reduces to B via a 1-1 length-increasing reduction. 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 $Im(x)$ 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 the function that maps x to such a string z. Clearly $h \in \mathrm{NPMV}_t$. Since $\mathbf{Q}$ holds, h has a refinement h' in PF. Hence h' is the 1-li reduction from A to B. $\square$

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. $\square$